IBM DB2 Alphablox

DB2 Alphablox 开发者指南

版本 8.4

S151-0144-01

IBM DB2 Alphablox

DB2 Alphablox 开发者指南

版本 8.4

S151-0144-01

注意:

在使用本资料及其支持的产品之前，请阅读第 253 页的『声明』中的信息。

第三版（2006 年 3 月）

本版本适用于 IBM DB2 Alphablox for Linux, UNIX and Windows（产品号 5724-L14）的 V8R4 及所有后续发行版和修订版，直到在新版本中另有声明为止。

目录

第 1 章 DB2 Alphablox 应用程序和底层的 Blox

DB2® Alphablox 使您能够创建定制业务分析应用程序 – 帮助最终用户将来自各种数据源的实时业务数据和事务可视化并进行分析的应用程序。DB2 Alphablox 应用程序通常通过简单易用的界面来合并业务逻辑和提供导向分析，而不是仅仅以查询工具和报告工具的方式来提供数据。

DB2 Alphablox 应用程序可以是任何包含 DB2 Alphablox 构建块（称为 Blox）的 Web 应用程序。此应用程序可以是一个 JSP 页面那么简单，也可以是与各种应用程序服务器和数据源进行通信的整个 Web 页面集合那么复杂。

Blox 是可重用的软件组件，您可以使用 JSP 标记或 Java™ 代码来将它们添加到 JSP 页面中以连接至数据源、执行数据转换和计算以及提供交互式的数据分析功能。

本节着重讲述 DB2 Alphablox 应用程序的公共关键特征。借助图形表示法和样本方案，本节演示了 DB2 Alphablox 中的功能和组件是如何实现这些特征的。

要了解有关 DB2 Alphablox 应用程序流和开发方法的详细信息，请参阅 第 9 页的第 2 章，『DB2 Alphablox 应用程序流』。

DB2 Alphablox 应用程序的主要特征

DB2 Alphablox 应用程序通常具有下列特征。每一项特征都可以通过将 DB2 Alphablox 中的功能进行各种组合来实现:

- 『实时数据访问和分析』
- 第 2 页的『交互式用户界面』
- 第 3 页的『个性化』
- 第 3 页的『共享和协作』
- 第 4 页的『实时规划』

实时数据访问和分析

DB2 Alphablox 应用程序可以对来自多个数据源（既包括关系数据源也包括多维数据源）的数据驱动数据分析。通过对数据库（用于 Microsoft® Analysis Services 的 MDX，用于 DB2 OLAP Server™ 和 Essbase 的“报告脚本”以及用于关系数据库的 JDBC）进行本机访问，DB2 Alphablox 揭示了数据库引擎的分析功能，如排名、派生的计算、排序、过滤、百分比、方差、标准偏差、相关性、趋势、统计功能以及其他复杂计算。

可以通过不同的方法向用户显示实时数据。如果用户需要将数据显示在网格和图表中，则您首先将 DataBlox 添加到应用程序中，然后指定要用于该 DataBlox 实例的数据源。您将立即能够访问数据库引擎中固有的所有分析功能。然后，添加 PresentBlox（它将嵌入一个 GridBlox 和一个 ChartBlox）以使用该 DataBlox 中的数据。现在，用户可以通过 Blox 用户界面与最新的数据进行交互以满足他们的数据分析需求。

例如，对于 CFO，她在登录时看到的第一个屏幕可能是一个管理显示板，该显示板包含月度收入陈述和有关市场利润排名的概述。该数据是实时的，并且，如果该 CFO 想要了解哪个客户正在购买哪种产品的话，她可以选择对该数据进行下寻。

要根据关系数据库来创建报告，您可以使用“关系报告”。“关系报告”的核心是 ReportBlox 组件，该组件将关系结果集显示为基于 DHTML 的报告。其他 Blox 组件支持“关系报告”中的数据访问、数据转换、计算和格式化。在这些 Blox 中，每个 Blox 都执行它的名称所指的特定任务。

关系报告可以是静态的或交互式的。如果您希望以交互方式显示报告，则用户可以使用“关系报告”的“报告编辑器”用户界面来快速地对数据进行排序、过滤或重新排序以设计他们自己的关系报告。

交互式用户界面

DB2 Alphablox 应用程序通常具有网格和图表，这些网格和图表能够以 DHTML 格式显示并可以使用受支持的 Web 浏览器访问。

在 DHTML 客户机中显示的网格和图表具有简单易用的用户界面，此用户界面使用户可以避免分析数据的复杂过程。当您添加 PresentBlox 时，它可以嵌套 GridBlox、ChartBlox、ToolbarBlox、PageBlox 和 DataLayoutBlox 以便向用户提供交互式的数据分析功能、建立书签功能、数据导出功能和视图定制功能。作为开发者，您可以对界面中的各种组件进行定制和个性化以满足设计需求。

DataLayoutBlox 显示为数据布局面板，它使用户能够以交互方式在各个轴之间移动维以及查看维。ToolbarBlox 显示为工具栏，它使用户能够通过单击鼠标来快速地访问频繁执行的数据分析任务。菜单栏提供可供用户使用的所有选项和操作。用户可以对视图建立书签、隐藏和显示网格或图表、对数据进行排序、以 PDF 或 Excel 格式导出数据以及浏览数据。PageBlox 显示为页面过滤器，它允许用户进行数据过滤以将过滤得到的数据显示在 GridBlox 和 ChartBlox 中。所有这些 Blox 都嵌套在 PresentBlox 中，从而简化了应用程序开发工作并节省了屏幕空间。

可以使用 DB2 Alphablox 标记库中提供的 JSP 标记来定制用户界面中的组件。例如，可以指定要使用的颜色、在工具栏中添加或除去按钮、对菜单栏添加或除去菜单或者添加或除去数据导航选项。还可以指定一组用于突出显示单元格的条件，这就是称为“单元格提醒”的功能（例如，当单元格的值低于指定的最小值时，将单元格显示为红色）。

此用户界面的一个主要优点是每当用户与数据进行交互时，它不会涉及页面刷新。用户不必等待下载整个页面，而下载整个页面将意味着更长时间的等待并可能失去原始上下文的位置。在门户网站环境中，因为刷新整个门户网站页面涉及到重新装入页面上的所有 portlet，从而可能花费相当多的时间，所以这是一个很大的优点。

DB2 Alphablox 主题

DB2 Alphablox 为 DHTML 客户机提供了若干个现成的主题，这些主题具有各种即时可用的关联样式表和 GIF 图像。您也可以通过复制现有的 DB2 Alphablox 主题并接着修改该主题使用的样式表和图像来创建自己的主题。

成员过滤器

有时，用户想查看更特定的数据，而不是一次一个级别地进行上寻和下寻。他们可能想查看维层次结构中的不同父代的特定成员的数据。例如，用户可能仅仅想对每个地区的州代表的数据进行比较。

“成员过滤器”界面允许他们浏览 Market 维并从 East 地区中选择 New York，从 West 地区中选择 California，从 Central 地区中选择 Illinois，并从 South 地区中选择 Texas。

当提供了数据导航选项时，可以从 GridBlox、DataLayoutBlox 和 PageBlox 中的右键单击菜单和弹出菜单中使用成员过滤器。“成员过滤器”对话框窗口中列示的维取决于用户单击鼠标右键时打开菜单的位置。

关系报告用户界面

当您使用 ReportBlox 及其支持 Blox 来创建交互式报告时，用户可以通过“报告编辑器”用户界面来对列进行排序、隐藏列、将列重新排序、创建中断组以及为每个中断组添加总结数据。

“报告编辑器”由三个上下文相关弹出菜单组成。所有这些菜单都是通过 DHTML 受支持的。报告和交互式菜单是使用特定的 CSS 样式类显示的。可以通过指定要使用的样式来定制颜色和字体。

个性化

由于每个用户都具有不同的数据和业务需求，所以通常需要将 DB2 Alphablox 应用程序个性化。例如，根据用户的不同，他们在登录时看到的第一个屏幕可能有所不同。您可能想控制数据导航以使 West 地区的用户无法查看 East 地区的数据。您也可能想允许用户指定他们的图表类型首选项或者用来突出显示网格数据的阈值。

定制属性

DB2 Alphablox 通过定制属性支持个性化。您可以定义定制用户属性并指定每个属性的有效值。然后，对于每个用户，可以对每个已定义的属性指定值。根据已登录的用户以及与该用户相关联的属性值的不同，可以动态地指定要显示的内容、数据应该具有的显示方式或者应该启用或禁用的数据导航功能。

共享和协作

建立书签、对数据添加注释和 PDF 转换是 DB2 Alphablox 提供的一些用来支持共享和协作的常用功能。

建立书签

DB2 Alphablox 的一项关键功能就是它的“书签”功能。通过用户界面，用户可以对数据视图建立书签并在以后检索具有最新数据的同一视图。书签可以是专用的，可以供特定组中的用户使用，也可以是所有能够访问服务器的用户公用的。

BookmarksBlox 提供了一个广泛的用于管理和处理书签的 API。例如，可以有规划地来更新所有书签以反映数据轮廓的更改。

对数据添加注释

为了支持协作分析，您可以利用 CommentsBlox 来支持单元格级别、页面级别或应用程序级别的注释。用户可以通过右键单击单元格并选择“添加注释”来将注释添加到 GridBlox 中的数据单元格中。具有注释的单元格的角部有一个注释指示器，因此用户可以快速地发现它们并选择查看它们。

以 PDF 格式导出

用户常常会想要保存他们的工作或共享他们的数据视图。DB2 Alphablox 具有一个“以 PDF 格式导出”选项，此选项使您能够将 Blox 中的数据以 PDF 格式保存。这解决了使用浏览器来打印或保存 Web 页面时的许多常见问题，如换页不正确、页面宽度对于显示图表来说不合适、不同的浏览器具有不同的打印效果以及必须通过电子邮件发送报告中所使用的所有 HTML 和图像。

“以 PDF 格式导出”选项使用户可以控制页面布局、页面方向、换页符、页眉和页脚文本以及图表大小。如果您选择该选项，则您也能够定制对话框以允许用户控制字体大小、颜色以及页眉和页脚的位置。

实时规划

分析应用程序既可以进行历史分析也可以预测将来的和前摄性的资源分配。可以通过使用 DB2 Alphablox 的数据回写功能来构建实时规划应用程序，如预算、销售预测和协作需求规划。

例如，可以将数据从数据源抽取到 GridBlox 中、允许区域经理在 GridBlox 中输入销售预测数据并且在提交该数据时将该数据回写到数据源中。通过与定制属性配合使用，应用程序可以根据用户所属的地区来动态地创建销售预测工作表。

底层的 Blox 组件

Blox 组件是 DB2 Alphablox 应用程序底层的关键组件。Blox 是可重用的软件组件，它使您能够连接至数据源、执行各种数据处理和表示任务以及构建动态的个性化数据分析应用程序。

一个 Blox 可以通过它的属性和相关联的方法提供若干项上述功能。这些属性和方法使您能够指定和控制 Blox 的外观和行为。还提供了用于处理用户事件（如上寻／下寻、旋转、更改页面过滤器或装入书签）的事件过滤器。

下表提供了每个 Blox 的简要描述。

Blox	描述
DataBlox	• 第 5 页的『DataBlox』：为所有数据表示 Blox 提供数据访问、检索和处理功能的关键 Blox。 • StoredProceduresBlox: 允许创建一个关系数据库连接和准备存储过程语句以供使用。

Blox	描述
用户界面 Blox	提供了 6 个使您能够以网格或图表格式显示数据的 Blox，这些 Blox 具有交互式用户界面，用户可以通过该界面来分析数据、更改页面过滤器、添加或装入书签以及指定网格或图表布局等等。这些 Blox 是: • 第 6 页的『GridBlox』 • 第 6 页的『ChartBlox』 • 第 6 页的『DataLayoutBlox』 • 第 7 页的『PageBlox』 • 第 7 页的『ToolbarBlox』 • 第 7 页的『PresentBlox』 在这些 Blox 中，每个 Blox 都具有用户界面组件和应用程序编程接口（API），该接口使您能够控制那些 Blox 的表示以及所允许的操作。许多 Blox 返回包含元信息的结果集。
分析基础结构 Blox	这些 Blox 提供用于构建分析基础结构的方法。 • RepositoryBlox: 为开发者提供一种在 DB2 Alphablox 存储库中保存和检索应用程序属性的方法。 • BookmarksBlox: 允许有规划地来创建和管理书签以及动态地设置书签属性。 • CommentsBlox: 为应用程序提供单元格注释以及一般的页面／应用程序注释功能。 • AdminBlox: 提供对通过 DB2 Alphablox 主页中管理页面设置的有关服务器、用户、组、角色、数据源和应用程序等信息的有规划的访问
业务逻辑 Blox	这些 Blox 组件允许您将业务逻辑添加到应用程序中: • MDBQueryBlox: 使您能够采用一种语言来构建 OLAP 查询，而不必考虑底层服务器的查询语言 • MemberSecurityBlox: 使您能够隐藏成员以使未经授权的用户无法查看它们 • TimeSchemaBlox: 支持动态时序，如显示“最近三个月”的数据
FormBlox	提供了一系列 FormBlox 来为您提供熟悉的 HTML 表单界面和处理状态管理。这些 FormBlox 具有数据感知（data-aware）特性，它们允许用户选择您提供的数据源、维、成员或其他选项以创建个性化的查询。
ContainerBlox	只能在 DHTML 方式下使用的 ContainerBlox 可以用来创建定制 Blox 组件以及在用户会话期间管理持久性。
ReportBlox	以交互式 HTML 格式显示的 ReportBlox 是用来根据关系数据源构建报告的核心 Blox。*Relational Reporting Developer's Guide* 一书提供了有关 ReportBlox 及其支持 Blox 的详细信息。

下列各节着重讲述 DataBlox 及每个用户界面 Blox 的作用以及它们如何一起工作以便为开发者和用户提供下列功能:

- 为开发者提供程序控制
- 为用户提供可视数据分析体验

DataBlox

DataBlox 是专门负责提供数据访问所需的功能的 Blox。它不具有图形用户界面。但它是所有为用户提供了用于与数据进行交互的图形用户界面的 Blox 的核心。它具有广泛的应用程序编程接口（API）。例如，您可以检测是否已成功地连接至所需的数据源或者当前数据库操作是否已完成等等。可以根据用户的身份防止该用户执行某些数据导航操作或查看结果集中的某些数据。下表显示了与 DataBlox 相关联的某些属性和方法以证明它的 API 的扩充性。

DataBlox 属性 / 方法的类别	描述	示例
数据源	与数据源相关的属性	aliasTable、catalog、query、schema、connectOnStartup、userName、password 和 dataSourceName
数据处理	与数据处理（如计算的成员、排序和钻取）相关的属性	calculatedMembers、columnSort、rowsort、hiddenMembers、keepOnly 和 parentFirst
数据外观	与数据外观（如是否应该在成员名中显示重复的数据、缺少的数据、全零数据或前缀以及它们的显示方式）相关	memberNameRemovePrefix、memberNameRemoveSuffix、suppressDuplicates、suppressMissing、suppressNoAccess 和 suppressZeros
回写	与数据更新相关	commitData()
结果集	与包含数据的结果集相关	clearResultSet() 和 getResultSet()
元数据	与当前 DataBlox 的底层数据源的 MetaData 对象相关	dimensionRoot 和 getMetaData()
MDB 结果集	与多维数据结果集中的轴、维、元组和单元格相关	((MDBResultSet) getResultSet())
RDB 结果集	与关系数据结果集中的列和行相关	((RDBResultSet) getResultSet())
MDB 元数据	与结果集的多维元数据相关	((MDBMetaData) getMetaData())
RDB 元数据	与结果集的关系元数据相关	((RDBMetaData) getMetaData())
事件过滤器	与服务器端事件过滤器相关	addFilter() 和 removeColumnSort()

GridBlox

GridBlox 以高级网格格式显示关系数据或多维数据，从而使用户能够对数据进行钻取、旋转、排序和探索。它具有一组广泛的属性及相关联的方法来允许您控制它的外观、数据格式化及其他方面。缺省情况下，一个独立的 GridBlox 具有:

- 一个菜单栏，它提供了可供该 GridBlox 及底层的 DataBlox 使用的所有选项和功能
- 一个 ToolbarBlox，它使用户能够通过单击按钮来快速地访问常用功能

ChartBlox

ChartBlox 以各种图表格式显示关系数据或多维数据，从而使用户能够更改图表外观和探索数据。ChartBlox 需要一个 DataBlox 以提供数据访问功能和数据处理功能。缺省情况下，一个独立的 ChartBlox 具有:

- 一个菜单栏，它提供了可供该 ChartBlox 及底层的 DataBlox 使用的所有选项和功能
- 一个 ToolbarBlox，它使用户能够通过单击按钮来快速地访问常用功能

DataLayoutBlox

DataLayoutBlox 显示可用的数据维以及它们当前所在的轴，从而使用户能够在轴之间移动维。DataLayoutBlox 嵌套在 PresentBlox 内。它不能嵌套在独立的 GridBlox 或独立的 ChartBlox 中。

当用户将一个维从一个轴移至另一个轴时，同一个嵌套 PresentBlox 内的 GridBlox 和 ChartBlox 中的数据都将自动地反映所作的更改。

PageBlox

PageBlox 显示位于页面轴上的维，从而有效地对图表或网格中的数据进行过滤并使用户能够更改数据过滤器。PageBlox 嵌套在 PresentBlox 中。它不能嵌套在独立的 GridBlox 或独立的 ChartBlox 中。当用户从 PageBlox 中的维中进行选择时，同一个嵌套 PresentBlox 内的 GridBlox 和 ChartBlox 中的数据将反映选择的过滤器。

ToolbarBlox

ToolbarBlox 显示了一些按钮，从而使用户能够访问各种 Blox 功能。ToolbarBlox 需要嵌套在 PresentBlox 或者独立的 GridBlox 或 ChartBlox 中。缺省情况下，ToolbarBlox 在这些用户界面 Blox 中处于打开状态，并且显示为两个工具栏。这些工具栏是可定制的。您可以对现有工具栏添加或除去按钮。您甚至可以添加或除去工具栏。

PresentBlox

PresentBlox 将所有上述 Blox 组合到单个 Blox 中以简化应用程序开发工作和节省屏幕空间。所有嵌套在 PresentBlox 中的 Blox 都相互进行交互。它们使用同一个数据源，因此，在 PageBlox 中进行的数据导航操作会同时影响显示在嵌套的 GridBlox 和 ChartBlox 中的数据。在适用的情况下，指定的数据选项将在它嵌套的所有 Blox 中反映出来。例如，如果您指定对成员名使用别名，则将在 GridBlox、ChartBlox 和 PageBlox 中使用别名。

DB2 Alphablox FastForward

DB2 Alphablox FastForward 是用于快速地开发、部署和共享定制分析视图的样本应用程序框架。FastForward 框架提供了现成的公共应用程序服务，这些服务包括安全性、协作、定制和个性化。应用程序管理员（通常是 OLAP 管理员）可以创建新版本的 FastForward 应用程序，通过选择报告模板和配置报告参数来发布报告，然后部署新的应用程序，而不必查看任何代码。JSP 开发者可以进一步修改或扩展应用程序框架，并且可以添加新的定制报告模板以供应用程序管理员配置和部署。有关更多信息，请参阅《*开发者指南*》中的『使用 DB2 Alphablox FastForward』主题。

第 2 章 DB2 Alphablox 应用程序流

本节描述 DB2 Alphablox 应用程序的文件结构、DB2 Alphablox 和应用程序服务器处理应用程序的方式以及应用程序开发者使用标准 Web 技术开发应用程序以获得期望的最终用户交互和程序控制的方式。

应用程序文件结构

由于 DB2 Alphablox 在 Java 2 Enterprise Edition（J2EE）Web 应用程序服务器环境中运行，因此本节描述了当您创建 DB2 Alphablox 应用程序时在该底层应用程序服务器中的文件结构。

应用程序上下文

当您从 DB2 Alphablox 主页中创建应用程序时，将要求您指定诸如应用程序上下文、显示名称、主 URL、缺省保存状态和写特权安全角色等信息。根据这组信息，DB2 Alphablox 在 DB2 Alphablox 存储库中创建应用程序定义，并创建应用程序目录结构。这将创建名为您指定的应用程序上下文的目录，并且通常将该目录称为应用程序“docroot”应用程序上下文或应用程序目录。

这个应用程序目录在物理上的所在位置取决于应用程序服务器。当 DB2 Alphablox 是使用 WebSphere® 所安装时，应用程序目录位于 WebSphere 的 `installedApps` 目录中。有关更多信息，请参阅*《管理员指南》*。

您创建的应用程序的所有文件都必须在这个应用程序目录结构中。通常，这些文件是 JSP 文件、HTML 文件、CSS 文件、JavaScript™ 文件与图像文件的组合。并且，通常按照 J2EE 规范的建议，将 Java 类或其他包含 servlet、bean 或其他实用程序类的 Java 归档文件放在 `WEB-INF` 目录、`classes` 目录和 `lib` 目录中的子目录中。

DB2 Alphablox 存储库

DB2 Alphablox 存储库是一个对象库，DB2 Alphablox 使用它来跟踪应用程序、用户、组、书签和其他此类信息。当使用 DB2 Alphablox 文件系统存储库时，与 DB2 Alphablox 存储库相关联的物理文件放在 `<db2alphablox_dir>/repository` 目录中，其中 `<db2alphablox_dir>` 是 DB2 Alphablox 的安装位置。

例如，当从 DB2 Alphablox 主页中创建名为“MyApp1”的应用程序时，将在 `<db2alphablox_dir>/repository/applications/` 目录下面为该应用程序创建名为“MyApp1”的文件夹。当您定义定制应用程序属性时，将更新应用程序属性描述符文件 `appprodesc.properties` 以存储信息。

同样，在添加用户时，将在 `<db2alphablox_dir>/repository/users/` 目录下面创建具有该用户名的文件夹。每个用户都具有相关联的用户属性文件，该文件存储诸如通过 DB2 Alphablox 主页定义的密码、电子邮件地址和组关联之类的信息。

通过使用 RepositoryBlox API，您可以获取、设置、保存或删除应用程序状态或者获取用户名和用户所属的组。

在 JavaServer Pages 中使用 Blox

在 J2EE 环境中，为了提供动态内容，要使用的关键技术是 JavaServer Pages（JSP）。JSP 技术允许将 HTML、JavaScript 和 Java 代码组合在一个物理文件中。

由于 Blox 通常是 Java bean，所以，要添加 Blox，请使用 JSP 标记来包括 bean，就象您通常使用 `<jsp:useBean>` 标记来包括任何 Java bean 一样。也可以利用 DB2 Alphablox 定制 JSP 标记来使用类似于 XML 的语法来添加 Blox。

请求处理

本节描述底层的应用程序服务器和 DB2 Alphablox 如果处理对 DB2 Alphablox 应用程序的 HTTP 请求。下列各节对此过程作了简要的概述。要获得更完整的描述，请参阅 JavaServer Pages 技术书籍。

以下描述基于具有应用程序上下文 MyApp1 的应用程序，该应用程序上下文具有下列文件:

- `welcome.html`: 应用程序登录页面。此页面包含指向 `intro.jsp` 和 `firstGrid.jsp` 的链接。
- `intro.jsp`: 一个 JSP 文件，它包含一些一般的 Java 和 JavaScript 代码。
- `firstGrid.jsp`: 一个包含 GridBlox 的 JSP 文件，它与前面『在 JavaServer Pages 中使用 Blox』一节中显示的那个 JSP 文件相似。

这些描述还假定应用程序服务器负责提供 Web 页面，而没有独立的 Web 服务器。

用户请求 1（http://myAppServer/MyApp1/welcome.html）

1. 用户“dave”通过浏览器访问 `http://myAppServer/MyApp1/welcome.html`。
2. 应用程序服务器转到 `MyApp1/` 并检查 `WEB-INF/` 目录中的应用程序部署描述符文件 `web.xml` 中定义的安全信息。
3. 根据定义的安全约束，应用程序服务器提出用户名和密码请求。
4. 在认证之后，将启动一个 J2EE 会话。应用程序服务器在响应中将一个 cookie 发送回给浏览器。发送的 cookie 包含会话标识。
5. 应用程序服务器解析 `welcome.html` 并将其发送回给浏览器。

用户请求 2（http://myAppServer/MyApp1/intro.jsp）

1. Dave 单击指向 `intro.jsp` 的链接。这将发送对 `http://myAppServer/MyApp1/intro.jsp` 的 HTTP 请求。
2. 应用程序服务器访问 cookie 和头信息以查找 J2EE 会话标识并验证安全性。
3. 应用程序服务器具有一个 JSP 引擎，后者编译并执行 JSP 文件。应用程序服务器首先检查此文件是否已被编译或者在上次编译后是否已更改。

   如果需要进行编译，则该引擎处理该文件并将其编译成 Java 类文件。它检查 JSP 文件中引用的类和包是否存在以及语法是否正确。
4. 应用程序服务器执行经过编译的文件并将响应发回给浏览器。

用户请求 3（http://myAppServer/MyApp1/firstGrid.jsp）

1. Dave 返回到 welcome.html 并单击指向 firstGrid.jsp 的链接。这将发送对 http://myAppServer/MyApp1/firstGrid.jsp 的 HTTP 请求。
2. 应用程序服务器访问 cookie 和头信息以查找 J2EE 会话标识并验证安全性。
3. 应用程序服务器首先检查此文件是否已被编译或者在上次编译后是否已更改。
4. 如果需要进行编译，则它的 JSP 引擎处理该文件并将其编译成 Java 类文件。应用程序服务器检查 <%@ ...%> 伪指令中引用的类、包和标记库描述符文件（TLD）是否存在、使用的 Java 方法和定制标记是否有效以及语法是否正确。然后，应用程序服务器执行 firstGrid.jsp。
5. 它将遇到以下 scriptlet 并对其进行处理。变量 banding 将得到 true 或 false 值:

   ```
   <% String banding =
   (Math.random() >= 0.5) ? "true" : "false"; %>
   ```

6. 然后，它将遇到它不熟悉的标记 <blox:grid...>。前缀 blox 与 taglib 伪指令 <%@ taglib uri="bloxtld" prefix="blox" %> 中指定的内容匹配。
7. 应用程序服务器转到 taglib 伪指令中定义的标记库。标记是“宏”，它们将被负责创建和初始化 bean 的实际 Java 代码替换。应用程序部署描述符文件 /MyApp1/WEB-INF/web.xml 将标记库描述符文件（TLD）的所在位置告知应用程序服务器:

   ```
   <taglib>
      <taglib-uri>bloxtld</taglib-uri>
      <taglib-location>/WEB-INF/tlds/blox.tld</taglib-location>
   </taglib>
   ```

8. DB2 Alphablox 现在开始工作。DB2 Alphablox 初始化 bean、用户会话、应用程序实例和同位体，并将结果发送回给应用程序服务器。下一节（『DB2 Alphablox 程序流』）讨论了有关如何处理 Blox 以及为其提供服务的详细信息。
9. 应用程序服务器继续处理 firstGrid.jsp 中的各行，直到它遇到末尾为止。
10. 应用程序服务器将结果发送回给浏览器。

应用程序服务器的角色

总的来说，应用程序服务器负责完成下列任务:

- 用户认证和安全性
- 处理和提供 HTML 文件
- 在它的 servlet/JSP 引擎的帮助下处理和编译 JSP 文件，然后将生成的整个响应提供给浏览器

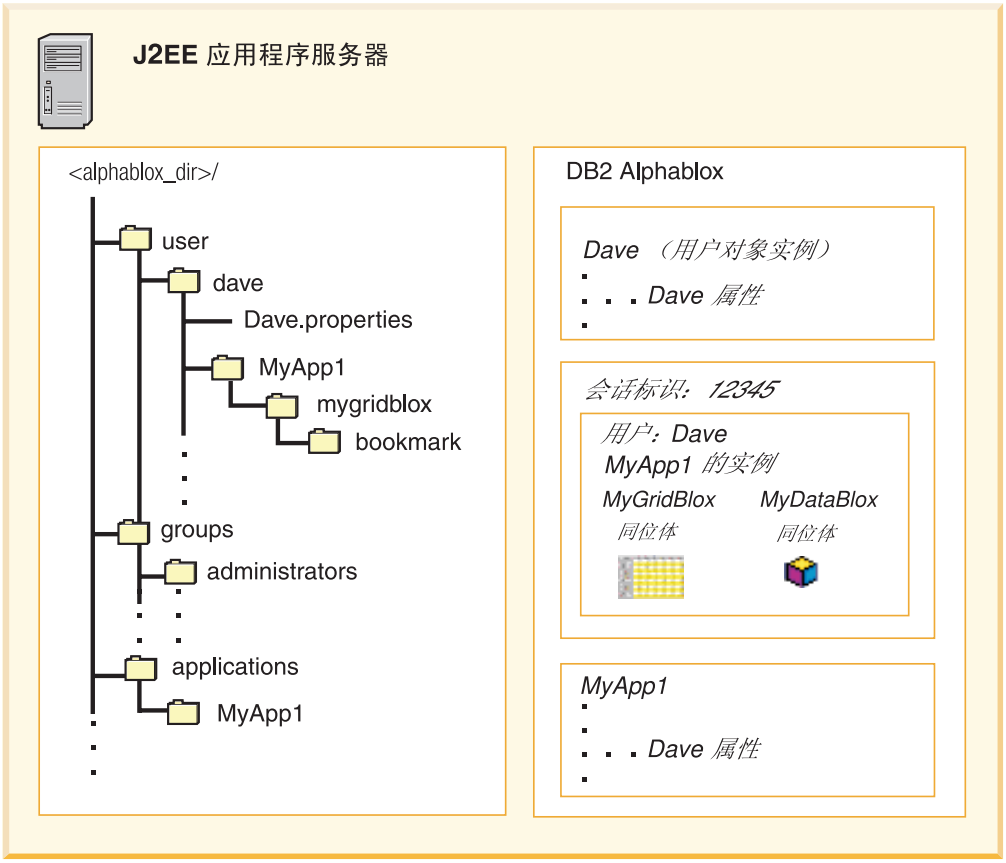
DB2 Alphablox 程序流

当应用程序服务器遇到诸如 <blox:grid> 之类的定制标记时，它将访问指定的标记库。根据应用程序的 web.xml 文件中指定的内容，应用程序服务器确定在将标记替换为 Java 代码时要使用的 Java 包:

在这个时候，DB2 Alphablox 执行下列任务:

1. DB2 Alphablox 从请求对象（J2EE 中的一个 API）中获取用户并检查是否已创建了 Dave 的用户对象。如果这是来自该用户的第一个请求，则 DB2 Alphablox 创建 DB2 Alphablox 用户。
2. DB2 Alphablox 从 DB2 Alphablox 存储库中装入用户概要文件并创建用户实例。

3. 接着，DB2 Alphablox 创建会话实例并指定响应头中包括的会话标识。将把该用户对象的实例添加到会话中。
4. 然后，DB2 Alphablox 创建应用程序实例。
5. 下一步，DB2 Alphablox 从请求对象中检索应用程序名并检查与该应用程序名相匹配的应用程序对象是否已存在。如果还不存在的话，DB2 Alphablox 就将创建应用程序对象并将该应用程序的实例添加到会话实例中。
6. 既然已经创建了用户、会话和应用程序的实例，所以 DB2 Alphablox 可以创建同位体了。
   a. `firstGrid.jsp` 具有一个标识为 `MyGridBlox` 的 GridBlox。DB2 Alphablox 检查 `MyGridBlox` 的网格同位体是否已存在。如果不存在，DB2 Alphablox 就创建一个。
   b. 网格同位体查找相关联的数据同位体。如果还不存在，DB2 Alphablox 就创建一个。
7. DB2 Alphablox 将显示的结果返回给应用程序服务器。应用程序服务器获取 DB2 Alphablox 发送的输出，将其合并到文件的其余部分中，然后再将结果发送回给浏览器。



如果同一个用户在同一个会话中进行了对同一个应用程序的另一个请求，则将重用现有的同位体。

DB2 Alphablox 的角色

总的来说，DB2 Alphablox 负责完成下列任务:

- 数据访问和处理
- 构建和部署交互式分析应用程序
- 数据视图的个性化（下列各节提供了更多详细信息）

建立书签、应用程序状态和 DB2 Alphablox 存储库

当用户“dave”对数据视图建立书签时，根据该书签是被保存为专用的、公用的还是对指定的组可视，将在存储库中该用户、应用程序或组的文件夹中创建一个文件夹，其名称为表示 Blox（通常是 PresentBlox、GridBlox 或 ChartBlox）实例的名称。

存储库中存储的关于每个书签的信息包括它的可视性（专用、公用还是由指定的组使用）、应用程序名、表示 Blox 的宽度和高度、该视图的数据源和数据查询、该书签的描述以及与该视图相关联的颜色方案和其他数据显示选项。

通过提供的 API，您可以有规划地来获取具有指定可视性的书签的名称。您可以保存、删除、重命名或复原书签以及检测书签的保存和装入事件。也可以有规划地来创建书签或更改与书签保存在一起的所有数据查询以反映数据轮廓更改。

如果您通过 DB2 Alphablox 管理页面上的应用程序定义页面指定自动保存应用程序状态，则存储库还将存储应用程序的状态。DB2 Alphablox 将把有关所有 Blox（如果有多个 PresentBlox 或多个独立的 Blox 的话）的信息保存在应用程序中，这包括查询结果集、网格和图表外观以及用户进行的其他更改。

应用程序开发和编程模型

*《开发者指南》*阐述应用程序开发环境的设置、JavaServer Pages、Blox 标记库以及基于任务的实现步骤和详细信息。在研究那些详细信息之前，了解下列概念非常有用。

Blox 组件

Blox 组件是基于 Java bean 构建的。我们提供了详尽的 API 以允许您使用 Java、scriptlet 或 Blox 定制 JSP 标记来访问和控制服务器上的 Blox 和 Java 对象。服务器端 API 允许您控制 Blox 的表示和行为、防止业务逻辑（通过浏览器的查看源文件选项或保存文件选项）显示给用户以及使开发小组中的页面设计者不必处理复杂的编程。

JSP 和定制标记

JSP 是 J2EE 中的一项关键技术，它使您能够在一个文件中组合静态内容和动态内容。缺省情况下，如果文件具有 `.jsp` 扩展名，则应用程序服务器将仅调用它的 JSP 引擎来处理请求并生成动态内容。HTML 页面将不会通过 servlet 编译和请求生成过程。应用程序服务器将把该页面作为静态 HTML 页面提供，并且浏览器将忽略 JSP 代码和 Java scriptlet。因此，为了使用 DB2 Alphablox 提供的功能，应该将 Blox 添加到 JSP 页面中。

由于 Blox 是基于 Java bean 构建的，所以，您可以认为它们与 Java bean 具有相同的属性。例如，它们具有属性以及属性的 setter 和 getter 方法。可以使用标准的 `<jsp:useBean>` 标记来添加 Blox，然后使用 `<jsp:getProperty>` 和 `<jsp:setProperty>` 标记来获取／设置 Blox 属性。但是，应该尽可能地使用 DB2 Alphablox 标记库。当

使用定制 Blox 标记时，将自动地把作用域设置为“会话”，并且 DB2 Alphablox 负责进行会话管理并自动清除未使用的／到期的资源。

要了解有关这两种方法以及 Blox 标记库的信息，请参阅第 39 页的『Blox display 标记』。

服务器端 API 与客户机端 API

由于 Blox 是 Java bean，所以，您可以访问服务器上的 Blox 及其同位体以获取信息并控制 Blox 的行为和外观。在将输出发送至客户机之前，处理是在服务器上进行的。这允许您使用服务器上的其他资源、重用组件以及减少不同浏览器或不同浏览器版本之间通常存在的差别和不一致性。通常，在服务器端，可以使用 JSP、Java scriptlet 或定制标记来完成下列工作:

- 创建 Blox 的实例
- 自动设置 Blox 的属性
- 获取 Blox 的属性

在某些情况下，您想让用户能够进行选择，如选择要查看的数据的地区或指定一些参数来显示网格，这时，您将需要调用一些 JavaScript 函数来将用户进行的选择传递到服务器。DHTML 客户机具有简单明了的客户机端 API，它允许您调用服务器上的 JSP 页面或服务器端 bean 并设置其属性。本书的随后内容详细叙述了客户机端 API。

第 3 章 开发环境

DB2 Alphablox 解决方案基于开放的万维网标准，因此，您能够选择许多开发工具来使用 DB2 Alphablox 组件构建分析应用程序。

如果您已经有了易于使用的 Web 开发工具，则您很可能会继续使用那些工具来使用 DB2 Alphablox 开发分析应用程序。在本节中，我们讨论了一些与开发者相关的重要问题，希望能够最大程度地为您提供帮助。

选择应用程序开发工具

设计 DB2 Alphablox 解决方案的意图是为了支持因特网的开放标准技术，这些技术包括 HTML、CSS、JavaScript 以及其他技术。由于不需要专门的开发环境，所以，DB2 Alphablox 允许您选择您最熟悉或最易于使用的工具。有经验的 Java 开发者可能已经能够熟练地使用 IBM® Rational® Application Developer、Rational Web Developer、带有 Web Tools Platform 插件的 Eclipse 或其他 IDE。如果您不熟悉 Java 或者经常创作 Web 页面，则您可能能够熟练使用您喜爱的 HTML 编辑器。某些读者更喜欢使用功能强大的文本编辑器，如 Visual SlickEdit 或 jEdit。如果您尚未找到理想的开发环境，则可以探索许多可用的选项，并且，您要知道，您选择的开发工具很可能可以用来开发基于 DB2 Alphablox 的分析应用程序。

Web 浏览器

只能使用 Microsoft Internet Explorer 来支持 DB2 Alphablox 应用程序（请参阅《*安装指南*》以了解特定的需求）。

作为开发者，您将在开发环境中反复进行编码并在 Web 浏览器中测试代码，并且，您应该知道，一些常见的问题会影响您的工作。首先，在开发期间，要了解一些一般的浏览器注意事项和问题。其次，为了提高开发效率，您会想将测试浏览器配置为具有最理想的状态以便在开发期间使用。下列各节对这些问题作了阐述。

一般注意事项

在应用程序开发期间，您可能会发现，为“开发方式”配置浏览器是有好处的。下列任务提供了一些步骤，这些步骤将 Microsoft Internet Explorer 配置为能够最大程度地提高开发者的工作效率。记住，您始终应该使用用户将要使用的 Web 浏览器和配置来测试最终应用程序及其行为。在绝大多数情况下，开发方式与最终用户方式之间的配置差别并不会影响最终的结果，但是，使用最终用户有可能使用的所有可能浏览器和配置来测试应用程序始终是最佳的做法。

使用 DHTML 方式

在应用程序开发期间，当您使用 DHTML 客户机时，您应该了解几个要点。在这里，作者并不想全面地讨论如何使用 DHTML 技术来进行编码，而是想说明您在开发 DB2 Alphablox 应用程序时应该了解的几种常见行为。

修改 Blox 标记

您在使用 DHTML 客户机时最早了解的一个情况是，对 Blox 标记进行的修改在当前浏览会话期间不会生效。但在修改 Blox 标记之后，为了能够看到所作的更改，必须关闭浏览器并启动新的浏览会话。这是预料之中的行为，此行为是由 DHTML 客户机使用服务器端代码的方式决定的。

您在开发应用程序过程中可能会犯的另一个常见错误是无意中创建了多个具有相同 id 属性的 Blox 组件。通常，这是由于您复制并粘贴代码（包括复制并粘贴 Blox 定义标记以便在同一个或另一个页面上创建另一个 Blox）但忘记更改新 Blox 的 id 属性而发生的。如果两个 Blox 具有相同的 id，则第一个装入到浏览器内存中的 Blox 确定了第二个 Blox 的外观 – 第二个 Blox 的属性设置将被忽略。

使用 Microsoft Internet Explorer 进行配置和开发

下列步骤适用于 Microsoft Internet Explorer V5 或更高版本。

1. 打开 Microsoft Internet Explorer 浏览器。
2. 单击“工具”菜单并从子菜单中选择“Internet 选项”以打开“Internet 选项”窗口。
3. 在“Internet 临时文件”部分中，单击“设置”按钮。
4. “检查所存网页的较新版本”的缺省设置是“自动”。将此设置更改为“每次访问此页时检查”。此选择将使打开的 Web 页面更有可能是您正在开发的页面的最新版本。
5. 通过单击“确定”按钮关闭此对话框，但不要关闭“Internet 选项”对话框窗口。
6. 现在，在“Internet 选项”中选择“高级”选项卡。您应该会看到一个较长的可滚动复选框列表。下列各节阐述这个较长的选项窗口的不同部分。下列设置是可选的，但建议您进行下列设置以提高对 Web 页面进行故障诊断的能力。

   JavaScript 错误通知

7. [可选] 为了帮助您识别 JavaScript 错误，建议在“高级”选项的“浏览”部分中选取“显示每个脚本错误的通知”。此选项的功能是弹出一个您不能忽略的对话框，该对话框指示已发生 JavaScript 错误。如果未启用此选项，则您必须对左下角状态窗口中显示的任何 JavaScript 提醒消息加以注意。

技巧: 当在 Microsoft Internet Explorer 中查看 DB2 Alphablox 应用程序页面时，浏览器可能不会按照您期望的方式来处理您尝试显示的页面。有时，浏览器可能会重新显示高速缓存的页面，而不是显示更新后的新页面。以上设置应该有助于防止这种情况的发生，但并不可靠。即使您单击浏览器的“重新装入”按钮，显示的页面也可能仍然是高速缓存的页面。如果您觉得这是一个问题时，可以使用其他几个选项。首先，可以使用“Ctrl-刷新”技术来对页面强制执行硬刷新（从服务器获取新副本，而不是获取高速缓存的副本）：按下 Ctrl 键不放，然后单击“刷新”按钮。第二个选项是关闭并重新打开浏览器。这将得到新的浏览器会话，并且是保证所显示的页面就是最新页面的最可靠方法。

Web 浏览器 – 已知 Mozilla 问题

重点描述了与 Microsoft Internet Explorer 不同的已知 Mozilla 问题。

最好是使用受您的组织支持的 Web 浏览器测试应用程序。下表重点描述了 Blox UI 组件的 Mozilla 和 Mozilla Firefox Web 浏览器的已知行为，该浏览器与受支持的 Microsoft Internet Explorer 浏览器不同。

表 1. 值得注意的 Mozilla 问题

问题	值得注意的 Mozilla 问题
分解器容器中的图表大小调整	图表保持相同大小，然后重新绘制。Firefox 维护图表的宽高比，这会改变容器大小。
编辑副本（Blox 模型 API 和 UI 功能部件）	不受支持。没有任何方法可用于复制到 Gecko 引擎中的剪贴板。
编辑字段选择和插入标记位置	不受支持。在 Gecko 引擎中使用的 window.getSelection() 方法不返回编辑字段中的文本。这是一个已知局限性，可以在将来的发行版中修正。
组合框自动完成突出显示	不受支持。尽管工作已完成，但没有任何方法可用于突出显示已完成的文本。
工具提示中的换行符	不受支持。没有任何方法可用于将换行符添加至工具提示。换行符字符显示唯一字符。在 Firefox 中，DHTML 视图代码将改为使用空格来替换它们。
弹出菜单、右拉菜单和下拉工具栏	已限制为框架。在 Firefox 中，弹出菜单和右拉菜单必须保持在框架内，并自动调整以允许容纳尽可能多的菜单。
拖放差别	在 Firefox 中，进行拖动时不会显示已更改的光标并且不能放下它们。不能拖动选择字段。用户不能在可拖动的编辑字段中使用光标来选择文本。
可调整大小的对话框	无边框。在 Firefox 中，除去了可调整大小的对话框的边框。主要问题是因为使用了百分比随对话框内容进行缩放的边界，这在 Firefox 中是不便于使用的 - 内容总是太多。
缩放静态文本组件的大小	将忽略静态组件上设置的大小。要解决此问题，您可以在组件的父代上设置大小来获取两个浏览器之间的兼容性。
键盘支持（加速键）	不受支持。由于不能在 `div` 元素中设置 `tabindex`，所以它在 Firefox 中不受支持。它将在 Mozilla Firefox 1.8 中受支持。
辅助功能选项	不受支持
从右到左（RTL）网格显示	不受支持
Grid.setFixedScrollbarPosition(boolean) 和 Grid.isFixedScrollbarPosition()	不受支持

Application Studio

Application Studio 提供了可以帮助您进行学习和开发的示例及其他工具。可以通过 DB2 Alphablox 主页上的“组装”选项卡来访问 Application Studio。样本代码的文件位于 Application Studio 目录下的以下位置中，您可以查看样本代码并在应用程序中重用那些代码:

```
<db2alphablox_dir>\system\ApplicationStudio\Examples
```

本指南引用的“Blox 样本程序”示例集演示了我们讨论的许多技术，该示例集在 Examples 目录中。

第 4 章 设计注意事项

对于任何应用程序开发工作，在进行设计和开发工作之前，您需要清晰地了解需求，然后在设计和开发工作完成后评估应用程序成功与否。本主题包括一些一般的需求收集指南，这些指南能够帮助您标识用户需求以及在开始进行设计和开发工作之前需要考虑的其他问题。

定义应用程序需求

应用程序的设计目标是确保应用程序能够提供适当的信息和功能以满足特定用户的特定需求。在收集需求时，您需要调查三方面的内容：数据、用户界面和应用程序逻辑。

- 『数据需求』
- 第 20 页的『用户界面需求』
- 第 21 页的『应用程序逻辑需求』

数据需求

DB2 Alphablox 支持的特定应用程序类型就是联机分析处理（OLAP）应用程序。与在事务数据源中生成和访问数据的联机事务处理（OLTP）应用程序比较，OLAP 应用程序访问数据源中的数据。这些数据源通常包含根据事务详细信息整理的数据，并且，这些数据存储在立方体系结构中。

应用程序设计过程的其中一个组成部分是标识必需的数据以及与数据访问相关的任何安全性问题。回答下列问题有助于定义应用程序数据需求。

1. 用户想要得到或需要什么信息？

   尽可能准确地回答此问题是找到该信息以及有效查询该信息的第一步。确定用户是否已能够访问必需的信息以及启用访问权时是否涉及安全性问题等十分重要。例如，地区销售经理可能能够查看所有地区的数据，而地区销售代表可能只能查看他们所在地区的数据。

2. 信息在什么位置？

   DB2 Alphablox 应用程序可以访问各种多维数据库和关系数据库中的数据。在开始开发应用程序之前，务必考虑将需要访问哪些数据源并验证 DB2 Alphablox 是否支持您的需求。（要了解有关特定数据源支持的详细信息，请参阅《*安装指南*》。）许多分析应用程序依赖于以维／成员层次结构组织信息的多维数据源。Blox 是专门为利用这种数据层次结构而设计的；它们的用户界面允许在层次结构中进行上寻和下寻，从而根据维和成员来过滤数据以及将一个或多个维移至另一个轴等等。

DB2 Alphablox 也支持以行列格式组织信息的关系数据源。关系数据的一个用途是“钻取到详细信息”，此功能使用户能够从包含所有信息的多维数据源移至在关系数据源中的底层详细信息。

DB2 Alphablox 的 DB2 Alphablox Cube Server 组件使管理员能够根据关系数据源中的信息来创建多维数据立方体。对于不需要功能全面的 OLAP 数据源的可伸缩性和开销的

应用程序来说，DB2 Alphablox Cube Server 特别有用。DB2 Alphablox Cube Server 包含维元数据，因此，用户可以执行诸如钻取、旋转和过滤之类的操作。有关如何将关系数据转换为多维数据的信息，请参阅《*DB2 Alphablox Cube Server 管理员指南*》。

注意，当数据显示在 DB2 Alphablox 应用程序中时，用户并不了解底层数据格式。但是，如果数据具有关系格式，则需要多维格式的用户操作（如钻取）将被禁用。

从 DB2 Alphablox 开始，您可以使用 ReportBlox 来根据关系数据源开发交互式报告，从而允许用户通过内置的"报告编辑器"用户界面来添加中断组、对列进行重新排序、进行数据排序、将列重命名以及编辑单元格和头样式。有关 ReportBlox 及其支持 Blox 的信息，请参阅 *Relational Reporting Developer's Guide*。

用户界面需求

用户界面对应用程序的使用来讲十分关键。应用程序应该包括内容表示、应用程序导航和用户帮助。虽然本文档不会全面探讨什么是有效用户界面和 Web 页面设计，但是，本节对以下方面提供了一些指导:

- 『用户组』
- 『内容表示』
- 第 21 页的『用户指示信息』
- 第 21 页的『用户导航』
- 第 21 页的『数据处理』
- 第 21 页的『保存和复原工作』

用户组

在定义用户组需求时，请记住下列注意事项:

- 用户的使用需求通常各不相同。解决这些差别的一种方法是将用户分到较大组中的小组中。例如，Financial 用户组可以包含 Analyst 组和 Administrative 组。应用程序的界面可以根据用户所属的组来动态地更改。
- 用户的数据存取需求也各不相同。通常，数据源本身的安全设施支持用户级别的和组级别的访问限制。要确保用户可以方便地访问数据，DB2 Alphablox 用户组应该遵守数据源上已实现的那些访问限制。

内容表示

DB2 Alphablox 允许您和用户控制内容表示（用户只能对内容表示进行程度较低的控制）。您在组织和表示信息方面有着相当大的自由度。应用程序页面的外观可以是管理显示板、内部网门户网站或可打印的报告。

您还可以选择数据表示。通过选择适当的 Blox 并对那些 Blox 设置属性值，您可以选择数据是显示在网格中、显示在图表中还是显示在网格／图表的组合中。由于 DB2 Alphablox 提供了许多不同的图表类型，所以，您可以进行试验以了解哪种数据表示最为有效。

在适当的时候，可以允许用户更改图表类型以及在网格表示与图表表示之间进行切换等等。例如，某些用户喜欢使用能够快速传达百分比和趋势的饼图，而其他用户可能希望使用能够提供数值并支持复杂分析的网格。您应该清晰地了解应用程序所面向的用户以便设计适当并且有效的内容表示。

用户指示信息

DB2 Alphablox 提供了联机帮助，联机帮助提供了全面的指示信息来阐述每个 Blox 的用法。访问 DB2 Alphablox 帮助页面的缺省方法是单击工具栏中的问号或者单击菜单栏中的“帮助”>“帮助...”菜单选项。但是，您可能会发现，在每个应用程序页面上提供应用程序级别的用户指示信息有助于缩短用户的学习过程。

如果此方法并不合适，或者如果用户指示信息相当详尽，则您可以编辑并扩充 DB2 Alphablox 联机帮助。要了解更多信息，请参阅第 235 页的第 24 章，『添加用户帮助』。

用户导航

最简单的应用程序是带有一个或多个 Blox 的单个 JSP 页面。但是，如果应用程序需要若干个与用户进行交互的 Blox，则可能会发生两种情况。如果多个 Blox 在一个页面上，则在显示某些 Blox 时，其他 Blox 可能只有通过滚动才能显示在浏览器窗口中。因此，请考虑在页面中提供能够快速地从一个区域移至另一区域的链接。

更常见的情况是，应用程序由若干个 JSP 页面组成。这种应用程序设计应该包括页面之间的向后和向前链接。应用程序的“主页”可以链接至该应用程序中的所有其他页面并提供一个适当的位置来让用户了解应用程序功能特性及增强功能。

数据处理

您既可以开发具有全面交互功能的分析和假设分析方案应用程序，也可以开发静态表示应用程序以便快速地管理快照。事实上，仅仅通过启用或禁用 Blox 交互功能和工具栏，您就可能可以开发一个能够满足多项用户需求的应用程序。成功的应用程序设计要求您了解目标用户与数据进行交互的方式。

保存和复原工作

执行复杂分析的用户通常想要在某些点保存他们的工作或者使特定数据视图可供其他用户使用。DB2 Alphablox 通过工具栏按钮支持这些需求。

应用程序逻辑需求

另一个主要的设计领域是应用程序逻辑。虽然 Blox 能够进行数据访问、表示和处理，但是，大部分 DB2 Alphablox 应用程序提供了逻辑以满足特定的用户需求，如:

- 提供预定义的查询列表以供用户选择
- 使用户能够通过一系列相关选择来动态地构造查询
- 根据用户登录来设置初始查询、传递格式和应用程序外观
- 根据用户输入的值来突出显示异常数据
- 根据用户操作切换内容表示
- 执行“假设分析”方案并且（可选）将结果回写到数据源

可以组合使用 JSP、HTML 表单、JavaScript 函数、Java 和 DB2 Alphablox 定制属性来实现应用程序逻辑。

定制属性

通过 DB2 Alphablox 管理页面，您可以定义应用于应用程序、数据源和用户的定制属性。在定义定制属性之后，可以通过 RepositoryBlox 服务器端 Java 代码来使用该属性。要了解更多信息，请参阅第 200 页的『创建定制用户属性』。

注: portlet 依靠门户网站基础结构来访问用户概要文件信息。请记住，门户网站用户概要文件和 DB2 Alphablox 用户信息将分别进行维护。

portlet 开发规划

在门户网站应用程序中使用 Blox 组件时，在规划阶段需要考虑以下设计要求。因为启用了 Blox 的 portlet 需要与其他 portlet 在同一页面上，所以您需要了解一些事项以便页面上的所有 portlet 能够正确地协同工作。

- Blox 组件需要 Internet Explorer v5.5 和更高版本。确保此要求与由门户网站服务器维护的其他 portlet 一致。检查《*安装指南*》中的特定浏览器要求。
- 应该对具有 Blox 组件的应用程序关闭高速缓存。Blox 组件的功能强大的交互式用户界面和现场数据支持要求与服务器通信。如果打开了高速缓存，则这些功能不起作用。因此，同一页面上的所有 portlet 都需要在非高速缓存环境中工作。
- 由于上述相同的原因，启用了 Blox 的 portlet 在离线方式下不会工作。
- 需要通过 DB2 Alphablox 管理页面对 DB2 Alphablox 定义由 Blox 组件使用的数据源。需要从门户网站管理中单独地设置具有 DB2 Alphablox 管理权限的用户。您可能需要与 DB2 Alphablox 管理员和数据库管理员一起工作以准备好您的数据源。

设计可访问的应用程序

对于有残疾的用户，为所有操作提供键盘等效键非常重要。对于视力有限的用户，您还需要考虑文本浏览器和屏幕朗读器的局限性。DB2 Alphablox 中的 UI 组件支持 Internet Explorer 的辅助功能选项，并具有内置键盘快捷键和加速键。

当您开发和定制分析应用程序时，以下是支持辅助功能选项所需要记住的事项:

- 为定制菜单选项创建加速键。对于显示在菜单栏上的定制菜单项，您应指定自己的加速键访问键。可以通过为 `<bloxui:menu>` 和 `<bloxui:menuItem>` 标记设置 `accesskey` 标记属性来达到此目的。有关详细信息，请参阅 *Developer's Reference* 一书中的 Blox UI Tag Reference 一节。
- 禁止或限制使用图表组件。非图形浏览器和屏幕朗读器不能对视力不好的用户显示图像。图表组件具有图形特性，因而不能使用键盘访问。建议您从 PresentBlox 中除去图表组件（`chartAvailable="false"`）或仅对视力有限的用户使用网格组件。
- 在提供的高对比度主题中显示 Blox。使用高对比度主题显示的 Blox 用户界面不仅减少了灰影颜色的使用，而且还保持了在浏览器中设置的字体大小显示首选项。要使用高对比度主题，请将 `theme` URL 属性设置为 `highcontrast`。例如: `http://server/application/file.jsp?theme=highcontrast`。
- 使视力不好的用户不必使用菜单栏或工具栏，即可轻松了解数据内容。`<bloxui:accessibility>` 标记用于增强有残疾的用户的用户经验。它提供了一个标记属性，以允许用户跳过菜单栏或工具栏直接获取数据。它还使您可以提供对以前打开但未正在查看的对话框的键盘访问。有关详细信息，请参阅 *Developer's Reference* 一书中的 Blox UI Tag Reference 一节。

设计可访问的应用程序通常涉及到通过用户概要文件进行个性化设置。达到此目的的一种方式是通过 DB2 Alphablox 中的定制用户属性。例如，您可以通过 DB2 Alphablox 管理页面指定定制用户属性 accessible。对于视力不好的用户，此属性将被赋予一个特定值，如“Yes”。然后，您可以使用 RepositoryBlox getUserProperty() 方法来访问该值并通过编程将图表组件隐藏在 PresentBlox 中。

```
<%@ taglib uri="bloxtld" prefix="blox"%>
<html>
<head>
<blox:header/>
</head>
<body>
<blox:data id="myData" dataSourceName="QCC-Essbase"
query="!" />
<blox:repository id="myRepository" />
<blox:present id="myPresent" visible="false">
<blox:data bloxRef="myData" />
</blox:present>
<%
String accessible = myRepository.getUserProperty("accessible");
if (accessible.equals("Yes")) {
myPresent.setChartAvailable(false);
} else {
myPresent.setChartAvailable(true);
}
%>
<blox:display bloxRef="myPresent"/>
</body>
</html>
```

有关一般的应用程序辅助功能选项问题的更多信息，请访问 IBM Accessibility Center。此站点提供了许多有助于您了解和开发可访问的应用程序的开发者资源和准则。

设计多个语言环境

显示 Blox 用户界面和联机帮助所使用的语言取决于浏览器中的语言设置。当收到请求时，DB2 Alphablox 将根据请求中的 Accept-Language 头来确定客户机语言环境。语言环境是一个两字母的语言代码，后面可以有选择地跟一个两字母的国家或地区代码，通过下划线（“_”）分开。DB2 Alphablox 检查浏览器中的语言集列表，如果第一种语言不能与受支持的语言环境完全匹配，则它将顺着列表往下找，直到找到受支持的语言环境为止。如果没有受支持的语言环境，则 DB2 Alphablox 使用列表中的第一种语言并检查相关语言。如果相关语言不受支持，或由于某种原因未在浏览器上设置该语言，则 DB2 Alphablox 将根据服务器的语言环境来确定语言环境。如果全部都不符合，则 DB2 Alphablox 将缺省使用英语。

例如，如果浏览器的语言设置将 fr_CA 作为第一种语言，并将 en 作为第二种语言，但因为 fr_CA 与受支持的语言环境不完全匹配，所以将检查列表上的第二种语言。en 是受支持的语言环境，因此 Blox 用户界面将以英语显示。如果 fr_CA 是列表中仅有的一种语言，则 DB2 Alphablox 将查找相关语言 fr。因为 fr 受支持，所以 Blox 用户界面将以法语显示。

浏览器中的语言设置

要指定语言首选项:

- 在 Internet Explorer 中，选择**工具 → Internet 选项...**，并单击**语言...**按钮。
- 在 Firefox 或 Mozilla 浏览器中，选择**工具 → 选项...**，并单击**语言...**按钮。

在打开的窗口中，用户可以添加语言或更改语言优先级。

如果 Internet Explorer 用户从列表中意外地删除了所有语言，则由于该浏览器中已知的 XMLHTTP 问题，用户可能会遇到问题。

Blox UI 的语言环境与用于格式化的语言环境

如先前所述，DB2 Alphablox 通过检查在浏览器中设置的语言环境列表来确定 Blox 用户界面所使用的语言。然而，因为数据格式化是按 Java 格式进行的，所以它不局限于 DB2 Alphablox 所支持的语言环境。当确定用户想要用于对数字进行格式化的设置时，DB2 Alphablox 始终使用语言环境列表中第一种语言。因此，显示 Blox 用户界面的语言环境可能与用于格式化的语言环境不同。

图表标注的字体映射

虽然图表用户界面支持多个语言环境，但如果服务器的语言环境与客户机的不同，则服务器在显示客户机的语言环境中的符号时可能会出现问题。如果这对于用户是潜在问题，则您可以为图表轴标和图注指定要映射的系统字体。
<alphablox_dir>/repository/servers 目录下的 chartfonts.xml 文件允许您将 Java 识别的 5 个逻辑字体名称（Serif、SansSerif、Dialog、DialogInput 和 Monospaced）映射为诸如 Courier、Arial 和 Times New Romans 等物理字体名称。

对于每个语言环境，请指定语言和每个系统字体的字体名称。以下示例指定中文的字体映射:

```
<locale language="ZH">
<Serif>MingLiu</Serif>
<SansSerif>SimHei</SansSerif>
<Monospaced>SimHei</Monospaced>
<Dialog>SimHei</Dialog>
<DialogInput>SimHei</DialogInput>
</locale>
```

对于每个语言环境，您还可以在必要时指定国家或地区以及变体。例如，对于巴西葡萄牙语:

```
<locale language="PT" country="BR">
...
</locale>
```

对于每个逻辑字体，您还可以通过使用 `style` 属性来指定样式。有效值为 `plain`、`bold`、`bolditalic` 和 `italic`。数据类型定义（DTD）文件 chartfonts.dtd 在 <alphablox_dir>/xml 目录中提供。

字体名称区分大小写。如果在客户机上未找到 chartfont.xml 中指定的字体，则使用基本 JVM 字体映射来显示图表。

设计问题和开发技巧

以下是一些您应了解的一般 Blox 行为、服务器行为和应用程序开发技巧:

- BloxContext 使用 `getLocales()` 方法从浏览器中查找请求的语言环境设置。
- 因为 Blox 有会话作用域，所以在建立会话之后更改语言设置将在启动新会话后才生效。用户必须打开新的浏览器窗口才能在 Blox 用户界面中看到所反映的新语言环境。
- DB2 Alphablox 提供了一组用于 Java ResourceBundle 类的包装程序标记。这些标记允许您的定制代码访问特定于语言环境的资源。在资源束相关标记中讨论了这些标

记。要获取工作示例，请参阅 FastForward 应用程序。此样本应用程序使用这些包装程序标记来访问其 WEB-INF/classes/fastforward/FastForwardBundle.properties 文件中的资源束。

- 在 JSP 代码中指定图表类型时，请始终使用英语字符串或图表类型的数字值。
- 书签名称和描述始终以保存它们时的语言显示。以法语保存的书签名称和描述将始终以法语显示，而与装入书签时浏览器的语言设置无关。
- 当根据客户机语言环境显示由 DB2 Alphablox 生成的错误消息时，来自数据库的异常和错误消息将使用与数据库服务器相同的语言环境。DB2 Alphablox 仅从数据库传递该信息。
- 服务器日志始终基于服务器语言环境。如果 DB2 Alphablox 安装在使用德语的机器上，则服务器日志将使用德语。
- DB2 Alphablox Cube Server 在服务器语言环境中运行，并且所有错误消息和日志消息都将基于服务器语言环境。如果服务器和立方体关系数据源在不同的语言环境中运行，则查询可能包含具有排序不一致的结果。这是因为在处理 MDX 查询期间，可能有一部分排序操作在 DB2 Alphablox Cube Server 中执行，而另一部分排序操作在关系数据源中执行。

设计双向语言

双向（也称为 BiDi）语言是从右到左进行阅读的语言，而数字则仍是从左到右进行阅读。缺省情况下，Web 浏览器解释 HTML 页面中的代码并从左到右地显示可视组件。对于诸如阿拉伯语和希伯来语的双向语言，根据浏览器或用户 Windows® 系统设置的不同，可视组件可能不会自动从右到左地进行显示。Internet Explorer 用户可以通过**查看 > 编码** 菜单选项来设置查看方向，但并非所有浏览器版本都提供了该选项。Web 页面设计器也可以通过将 `dir` 属性添加至 `<body>` 标记并将其值设置为 `rtl` 来指定方向。

显示 Blox 组件的语言和方向由浏览器的语言环境设置来确定。在 ChartBlox 中，从左到右的方向意味着 X 轴标显示在图表的左边。从右到左的方向将把 X 轴标放置到右边。在 GridBlox 中，从左到右的方向意味着行标题显示在数据单元格的左边。从右到左的方向将把行标题放置到右边。

不需要其他代码，应用程序就能支持双向语言。根据设计目标的不同，如果您需要确保从右到左地显示 Blox 组件，则使用以下技术可达到此目的。

在 HTML 代码中设置 dir 属性

因为 Blox 用户界面以动态 HTML格式显示，所以您可以利用浏览器的功能来解释 HTML 代码中指定的方向指令。可以在 `<body>` 标记（`<body dir="rtl">`） 或内部 `<div>` 标记 (`<div dir="rtl">`) 中设置此 `dir` 属性。此步骤指示浏览器从右到左显示 Blox 组件，但仍允许用户通过提供的**网格选项**和**图表选项**对话框来更改显示方向。在这两个对话框中，应用程序的用户仍可以将显示方向设置为从左到右。

如果您需要自动更改该方向，则可以通过用参数传递 JSP 来达到这一目的。然后根据该参数传递的值来设置输出方向。例如，如果您有一个 `test.jsp` 文件，则您可以用一个参数来调用它，如下所示:

```
http://myServer/myApp/test.jsp?dir=rtl
```

`test.jsp` 文件获取参数值并在 `<body>` 标记中设置方向:

```
<!--test.jsp-->
<% taglib uri="bloxtld" prefix="blox"%>
<%@ page contentType="text/html;charset=utf-8" %>
<blox:data id="dataBlox"
dataSourceName="QCC-Essbase"
useAliases="true"
visible="false"
query="!" />
<html>
<head>
<blox:header/>
</head>
<body dir="<%= request.getParameter( "dir" ) %>">
<blox:present id="myPresent" width="700" height="500" menubarVisible="true">
<blox:data bloxRef="dataBlox" />
</blox:present>
</body>
</html>
```

在 UI 组件中设置方向

Blox UI 模型中所有可视组件的 Component 基类都有一个 setDirection() 方法。缺省值为 DIRECTION_DEFAULT，它意味着将采用在 HTML 代码或浏览器中设置的方向。如果您将组件的方向设置为 DIRECTION_RTL，则其子组件的显示方向将始终是从右到左，而与语言、浏览器设置或 HTML 代码中指定的方向无关。用户仍可以通过提供的**网格选项**和**图表选项**对话框来更改显示方向。

第 5 章 使用 JavaServer Pages 和 Blox 标记库

JavaServer Pages 技术在 DB2 Alphablox 应用程序中的使用使开发者能够快速地创建并方便地维护基于 Web 的分析应用程序。除了利用 DHTML 技术（包括 HTML、JavaScript 和 CSS）进行开发以外，JSP 技术还添加了动态脚本编制元素，这些元素使您不必掌握 Java 就可以利用 Java 的强大功能。本主题阐述如何在 DB2 Alphablox 应用程序中使用 JavaServer Pages 技术。

JavaServer Pages 技术

JavaServer Pages（JSP）技术使开发者能够使用他们熟悉的 DHTML 技术（包括 HTML、CSS 和 JavaScript）以及动态脚本编制元素（这些元素使开发者能够使用 Java 和服务器端处理）来快速地创建并方便地维护基于 Web 的分析应用程序。这种技术还能够帮助开发者创建不易受不同浏览器特性这一缺陷影响的应用程序。简而言之，JSP 技术的主要优点是:

- 将内容生成与表示分开

  通过使用 JSP 技术，Web 页面开发者可以使用 HTML 或 XML 标记来设计和格式化 Web 应用程序页面。JSP 标记和 scriptlet 使 Web 页面开发者能够使用他们熟悉的标记语法和脚本编制功能来生成页面并将核心程序逻辑隐藏在定制标记库和 Java bean 中。高级 Java 开发者可以使用 Java 来创建这些可重用的组件，这些组件可供 Web 页面设计者和应用程序开发者使用。

- 着重于可重用的组件

  大部分 JSP 页面依赖于使用跨平台的可重用组件，如 Java bean 和 servlet。通过使用 JSP，Web 页面设计者和开发者可以更方便地使用 Java bean 和 servlet 组件来生成内容。例如，Blox 是与服务器对等项进行交互的 Java bean，但开发者可以使用简单的标记来定义这些 bean。

- 通过标记简化 Web 开发工作

  JSP 技术通过将众多功能封装在简单易用并且特定于 JSP 的 XML 标记中来允许生成动态内容。这些标准 JSP 标记用来与 JavaBeans™ 组件进行交互、设置和获取 bean 属性以及执行其他以别的方式难以编码或者非常耗时的功能。JSP 定制标记库的使用允许 DB2 Alphablox 和其他产品创建简单易用的标记，这些标记可供 Web 页面设计者和开发者使用，同时不必考虑不相关的复杂功能。

*《开发者指南》*一书假定读者基本熟悉 JavaServer Pages 技术，但是，即使未掌握此知识，您也仍然可以创建一些基本的 DB2 Alphablox 应用程序。本主题的余下部分着重讲述如何将 JSP 与 DB2 Alphablox 配合使用。

要了解更多有关 JavaServer Pages 技术的信息，Alphablox 建议您参阅下列书籍并访问下列 Web 站点:

建议您阅读的书籍

Bergsten, Hans. 2004. *JavaServer Pages*. Sebastapol, CA: O'Reilly & Associates.

这是一本出色的 JavaServer Pages 和应用程序开发指南，读者不必是核心开发者。面向 Web 页面设计者和开发者的第一部分讨论 JSP 概念以及 JSP 是如何融入 Web 应用程序开发的。随后的面向编程部分讨论如何创建 JSP 组件和定制 JSP 标记。

Fields, Duane K.; Kolb, Mark A.; and Bayern, Shawn. 2001. *Web Development with JavaServer Pages (2nd edition)*. Greenwich, CT: Manning Publications.

这是另一本出色的 JavaServer Pages 指南，此书既面向 Web 页面设计者又面向 Java 开发者。此书的内容包括有关使用 JSP 1.2 和 Servlet 2.3 规范的讨论以及常见 Web 应用程序任务的示例。

Falkner, Jason (editor). 2001. *Beginning JSP Web Development*. Birmingham, UK: Wrox Press.

这本 JavaServer Pages 介绍书籍假定读者不具备编程经验并且仅具备初步的 HTML 经验。在介绍如何构建基于 Web 的应用程序时，此书对相关的 JSP 和 Java 概念作了说明。从第三章开始，您创建简单的 Java bean。

Web 站点

JavaServer Pages（Sun） - http://java.sun.com/products/jsp/

Sun 发明了 JavaServer Pages 技术，这个站点提供了最新的信息，包括新闻、规范、软件和教程。在 Technical Resources 部分中，您可以获取快速语法参考卡和指南的 PDF 版本。

JavaBeans（Sun） - http://java.sun.com/products/javabeans/

包括 JavaBeans 规范、教程以及关于 JavaBeans 技术的最新信息。

Servlets（Sun） - http://java.sun.com/products/servlets/

这是 Sun 的一个站点，此站点提供有关 servlet 技术的最新信息，包括新闻、规范和教程。

JSP Insider - http://www.jspinsider.com/

这是一个出色的信息来源，它提供 JSP 文章、参考指南以及指向其他 JSP 资源的链接。*JSP Buzz* 新闻通讯提供新闻和文章，并且是您保持与 JSP 产品和功能部件同步的良好途径。

JGuru - http://www.jguru.com/

此站点提供许多与 JSP Web 应用程序开发相关的主题文章。还提供关于 JSP 和 servlet 的有用 FAQ。

将 JavaServer Pages 与 DB2 Alphablox 配合使用

借助 JavaServer Pages 技术和 DB2 Alphablox，您可以快速地创建并方便地维护复杂的分析应用程序。虽然 JSP 是基于服务器的技术，但是，它允许您合并标准的客户机端技术，包括 HTML、JavaScript 和级联样式表。这允许作为 DB2 Alphablox 开发者的您使用这些技术来构建灵活并且可扩展的应用程序。

分析应用程序通常既利用客户机端技术也利用服务器端技术，从而使用这两种技术的精粹来交付应用程序。*Developer's Reference* 一书对整个 Blox API（包括 Blox 客户机 API 和服务器端 Java API）作了详细描述。

在大多数情况下，Blox（包括表示 Blox）是使用 Blox 标记库在 JSP 文件中定义的。使用 JavaServer Pages 技术开发的 Blox 标记库包括简单易用的标记，这些标记可用于指定 Blox 及其属性。可以使用其他 Blox 标记来处理常见的开发者任务，包括应用程序调试和业务逻辑调试。虽然可以使用标准的 JSP 操作（包括 `jsp:useBean`、`jsp:setProperty` 和 `jsp:getProperty` 标记）来对 DB2 Alphablox 开发 JSP 应用程序，但是，Blox 标记库提供了几乎完全相同的功能，并且只要求您完成少量的工作。本主题余下部分着重讲述使用 Blox 标记库中的核心 Blox 标记来定义 Blox。本指南的随后内容将对其他 Blox 标记库（包括 Blox 表单标记库、Blox 逻辑标记库和 Blox UI 标记库）进行讨论。

使用 DB2 Alphablox 的服务器端编程

The DB2 Alphablox 服务器端编程模型（SSPM）着重于尽可能地在 Web 应用程序服务器上处理应用程序逻辑和业务逻辑。DB2 Alphablox 提供了一组丰富的服务器端功能，并且支持 JavaServer Pages 和 Java 编程语言。与功能强大的应用程序服务器（如 BEA WebLogic 和 IBM WebSphere）结合使用，Alphablox 能够为开发者提供功能强大并且与 J2EE 相符的环境。DB2 Alphablox 提供了 Blox Java API，后者为开发者提供了 Java、JavaServer Pages、JavaBeans 组件和 Java Servlet 技术的全部功能。

本指南着重讲授如何利用 Java 服务器端编程模型的强大功能来快速地交付分析应用程序（即使您只具备有限的 Java 经验甚至完全没有经验也可以完成这些任务）。通过面向任务，本指南将帮助您快速学习使用 DB2 Alphablox 的强大功能来满足即时的业务需求。

使用 Blox 标记库

DB2 Alphablox Blox 标记库是使用支持 JSP 定制标记的 JavaServer Pages 技术开发的，它包括简单易用的标记，这些标记可供 Web 页面设计者和 Java 开发者使用。

使用了核心 Blox 标记来定义公共用户界面 Blox，后者包括 ChartBlox、DataBlox、DataLayoutBlox、GridBlox、PageBlox、PresentBlox 和 ToolbarBlox。还提供了用来定义专门用于构建关系报告应用程序的 Blox 的 Blox 标记。其他 Blox 标记库可用于创建功能强大的表单元素（Blox 表单标记库）、扩展 Blox UI（Blox UI 标记库）、处理复杂的业务逻辑（Blox 逻辑标记库）和支持门户网站应用程序中的基于 URL 的客户端链接。要了解关系报告需求，请参阅 *Relational Reporting Developer's Guide* 一书中对 Blox 报告标记库（包括 ReportBlox 以及其他相关联的 Blox）的讨论。

在描述使用 Blox 标记库的优点之前，让我们看看下列代码示例，您应该能够自己体会到使用 Blox 标记的优点。现在，请不要为不理解 Blox 标记的工作方式细节感到担心 – 我们很快就会对那些细节进行说明。但是您现在应该把注意力放在代码示例的布局和可读性方面。

如果您已经了解如何使用标准 JSP 语法来定义 Java bean，则您应该能够毫无困难地理解以下代码示例。但是，如果您不熟悉此语法，您就可能会有许多问题，并且您会担

心在开始取得任何进展之前必须学习大量新知识。记住，本示例着重说明较为困难的 Blox 编码工作，也就是使用标准 JSP 语法来定义一个 PresentBlox:

```
<jsp:useBean id="myPresentBlox"
scope="session"
class="com.alphablox.blox.PresentBlox">
<%
BloxContext context = BloxContextFactory.getBloxContext(request, response);
myPresentBlox.init(context,"myPresentBlox");
myPresentBlox.setProperty("width","540");
myPresentBlox.setProperty("height","350");

DataBlox myDataBlox=myPresentBlox.getDataBlox();
myDataBlox.setProperty("dataSourceName","TBC");
myDataBlox.setProperty("query","<ROW(Market) <ICHILD Market
<COLUMN(Year) Year !");
  myDataBlox.connect();
%>
</jsp:useBean>
```

*《开发者指南》*一书不对此语法进行过多解释。即使您不熟悉此语法，也不必担心，Blox 标记库提供了一种更为方便的方法来构建 Blox 组件和应用程序。在某些情况下，标准 JSP 语法是进行解决方案编码的唯一办法，但是，在大多数情况下，您可以使用 Blox 标记。现在，将上一个代码示例与以下示例作比较，以下示例使用 Blox 标记来定义同一个 PresentBlox:

```
<blox:present id="myPresentBlox"
width="540"
height="350">
<blox:data
dataSourceName="TBC"
query="<ROW(Market) <ICHILD Market <COLUMN(Year) Year !"/>
</blox:present>
```

正如您看到的，代码更易于阅读和维护。以下是您应该对使用 JSP 定制标记方法感兴趣的主要原因:

- 更易于阅读

  在 PresentBlox 示例中，使用 Blox 标记库创建的代码要易于阅读得多 – 使用标记属性以简单的名／值对形式映射特性，可以对该名／值对进行格式化以方便阅读。

- 更易于编码

  由于减少了需要输入的内容，所以，使用 Blox 标记创建 Blox 的编码速度可以快得多。不需要添加许多附加的行来进行初始化和连接至数据源 – 这是由 Blox 标记自动处理的。

- 更易于维护

  由于更易于阅读和编码，所以 Blox 标记也应该更易于维护。包括初始化 Blox（Java bean）和连接至数据源在内的细节都自动地被处理。您把精力放在定义 Blox 及其属性上，标记库负责使其正常工作。并且，将 Java 代码与标记库封装到一起有一个优点，即，Alphablox 与您都可以更方便地管理将来的 Java 代码更新。

  因此，让我们开始学习如何使用这些 Blox 标记来定义分析应用程序将要使用的 Blox 吧。

访问 Blox 标记库

缺省情况下，Web 页面将忽略它不了解的标记。这意味着，如果将 Blox 标记放到页面中，而未告诉该页面在什么位置查找关于那些标记的信息，该页面就会忽略那些标记。

因此，在利用 Blox 标记之前，需要在 JSP 页面顶部添加 taglib 伪指令行。应该使用的 JSP taglib 伪指令如下所示:

```
<%@ taglib uri="bloxtld" prefix="blox"%>
```

这一行代码告诉 JSP 编译器您打算使用定制标记库，该库位于作为 bloxtld 指定的 URI（统一资源标识）中。这个 URI（bloxtld）是一个缩写的标注，它定义 DB2 Alphablox 可以为 Blox 定制标记库找到标记库描述符文件（blox.tld）的位置。blox.tld 文件位于您创建的每个应用程序中，它通常位于以下位置:

```
/webapps/<applicationName>/WEB-INF/tlds/blox.tld
```

blox.tld 文件确定了在 DB2 Alphablox 应用程序中支持的标记和标记属性，在您使用 DB2 Alphablox 主页创建的新应用程序中，将自动创建此文件。

注: 如果您渴望了解更多关于标记库描述符（.tld）文件的信息，请参阅上面『JavaServer Pages 技术』一节中列示的其中一个推荐的 JavaServer Pages 技术资源。要了解更多关于如何创建 DB2 Alphablox 应用程序的信息，请参阅*《管理员指南》*。

可以将 taglib 伪指令放在 JSP 页面中的任何位置，只要它在该页面使用 Blox 标记之前出现就可以了。但是，最好的做法是将 taglib 伪指令放在 JSP 页面顶部的 <html> 标记前面，如下所示:

```
<%@ taglib uri="bloxtld" prefix="blox" %>
<html>
<head>
...
```

同样，taglib 伪指令通知 Web 应用程序服务器您打算使用 Blox 标记并且要求该库可用。于是，JSP 引擎对 JSP 页面进行语法分析，在页面中查找任何以 taglib 伪指令中定义的 blox 前缀开头的标记，当找到这样的标记时，它将执行标记库中的 Java 代码 – 您不需要查看该代码。

使用 Blox header 标记

在页面顶部添加 taglib 伪指令之后，您需要在页面上包括的一个重要标记是 Blox header 标记（<blox:header>）。这个标记管理页面上 Blox 的显示，并使关键的外部 JavaScript 和级联样式表（CSS）文件可用。并且，它还添加几行用于管理文件高速缓存的代码。

应该将 Blox header 标记放在 JSP 页面的 <head> 部分中，但应该放在 taglib 伪指令后面:

```
<%@ taglib uri="bloxtld" prefix="blox" %>
<html>
<head>
   <blox:header/>
...
</head>
```

您至少需要添加缩写的 <blox:header> 标记，如以下示例所示。但是，根据特定应用程序的不同，可能需要在此标记内添加嵌套的标记以执行其他重要的 Blox 操作。本指南的随后内容将对这种用法进行说明。要了解有关该标记的语法和用法的详细信息，请参阅 *Developer's Reference* 一书中的 The <blox:header> Tag 一节。

注: 当使用 <jsp:include> 来包括一个文件，并且该文件在 JSP 页面上带有 Blox 时，需要在该页面顶部添加 <blox:header> 标记，以使该 Blox 在初始化阶段不会挂起。

注: 当开发使用框架集以及使用多个应用程序中的多个框架的应用程序时，可以使用 <blox:session> 标记来帮助正确地管理用户会话。有关此标记的进一步信息，请参阅 *Developer's Reference* 一书中的 The <blox:session> Tag 一节。

下一节（『定义 Blox』）说明了如何使用 Blox 标记库中的 Blox 标记来在应用程序页面上定义 Blox。

定义 Blox

下表列示了用户界面 Blox 及其 JSP 定制标记:

Blox 名
: Blox 标记

ChartBlox
: <blox:chart>

DataBlox
: <blox:data>

DataLayoutBlox
: <blox:dataLayout>

GridBlox
: <blox:grid>

PageBlox
: <blox:page>

PresentBlox
: <blox:present>

注: 可以在 *Developer's Reference* 一书中找到有关这些 Blox 的详细信息以及它们的标记、属性和用法的完整语法。

注: 在 blox 前缀、冒号和标记名之间没有空格。如果在冒号后面指定空格，则将产生 JSP 编译器错误。

注: 在《*开发者指南*》的有关 Blox 标记的讨论中，您将看到对缩写标记语法的引用。这是为了使您明白所讨论的是标记而不是 Blox 本身。例如，本指南经常引用 `<blox:grid>` 标记，而不是引用 Blox GridBlox 标记。

正如前面 第 1 页的第 1 章，『DB2 Alphablox 应用程序和底层的 Blox』 中讨论的那样，Blox 既可以是独立的也可以作为子代嵌套在其他父 Blox 中，这取决于所使用的 Blox 以及它们的特定用法。缺省情况下，独立的 Blox 包括嵌套的 Blox，后者被设置为具有缺省值。例如，PresentBlox 包括嵌套的 ChartBlox、DataBlox、DataLayoutBlox、GridBlox、PageBlox 和 ToolbarBlox。

下表列示了每个独立的表示 Blox 中嵌套的 Blox 组件:

独立的 Blox
: 嵌套的 Blox 组件

ChartBlox
: DataBlox 和 ToolbarBlox

DataBlox
: CommentsBlox [可选]

DataLayoutBlox
: DataBlox

GridBlox
: DataBlox 和 ToolbarBlox

PageBlox
: DataBlox

PresentBlox
: ChartBlox、DataBlox、DataLayoutBlox、GridBlox、PageBlox 和 ToolbarBlox

可以仅包括顶级父 Blox 标记，而不是必须包括嵌套的 Blox 的嵌套标记（上面列示的嵌套的 Blox 是以隐式方式包括的）。

例如，当使用开始标记和结束标记进行编码时，使用 `<blox:present>` 标记定义的最小 PresentBlox 将类似于:

```
<blox:present id="myPresentBlox"></blox:present>
```

否则，当使用独立方法时，它类似于:

```
<blox:present id="myPresentBlox"/>
```

在大多数情况下，建议使用缩写方法 - 仅当需要在标记之间添加内容（称为主体）时，开始标记和结束标记才是必需的。

注: 以上最小示例包括一个 `id` 属性，这是因为，要正确地显示 Blox，必须至少定义此属性。必须唯一地标识应用程序中的所有父 Blox，但嵌套的 Blox 不需要 `id` 属性。

如果包括了 PresentBlox 的所有可能隐式嵌套 Blox 标记，则上面那个 PresentBlox 将类似于:

```
<blox:present id="myPresentBlox">
   <blox:grid/>
   <blox:chart/>
```

```
   <blox:toolbar/>
   <blox:page/>
   <blox:dataLayout/>
   <blox:data/>
</blox:present>
```

由于嵌套的 Blox 是以隐式方式包括的，即使未显式地添加嵌套的 Blox 标记亦如此，所以，仅当需要包括必需的标记属性，或者要包括需要更改为非缺省设置的属性的标记属性时，才需要包括嵌套的 Blox 标记。

如果将 PresentBlox 标记放在页面上，而未定义任何嵌套的 Blox 或定义任何标记属性，则将在该页面上显示一个 PresentBlox，但该 Blox 不会执行任何有意义的工作 – 如果不对嵌套的 `<blox:data>` 标记定义数据源和查询，就不会检索任何数据。为了让 Blox 执行一些有用的工作，您通常需要添加标记属性或嵌套的属性标记。

现在，您应该知道如何让 Blox 显示在 JSP 页面上以及如何包括嵌套的 Blox 了。下一节说明如何对 Blox 添加简单的标记属性。

使用标记属性来设置 Blox 属性

在 JSP 页面上将 Blox 实例化时使用的初始 Blox 属性由 Blox 标记属性中定义的设置或者嵌套的属性标记中定义的设置确定。下一节『使用属性标记来设置 Blox 属性』将对属性标记进行说明。

Blox 既有公共属性也有独特属性。并且，就象出现在其他 Blox 中的隐式 Blox 一样，Blox 的所有属性都具有缺省值。虽然您可能从来不需要更改这些缺省值中的绝大多数，但是，当您需要更改它们的值时，可以使用标记属性来显示并更改这些属性中的绝大多数。*Developer's Reference* 一书对可以用来定制 Blox 的数百个标记属性作了详细描述，但是，作为示例，让我们看看两个经常修改的属性。

在没有为 PresentBlox 定义数据源和初始查询的情况下，它也能够正确地显示，并且将在网格和图表部分中显示"没有数据可用"消息。要检索数据以供显示，必须使用 `dataSourceName` 和 `query` 属性来定义两个 DataBlox 属性。正如前面提到的那样，要访问和修改嵌套的 Blox，需要在顶级 Blox 中添加嵌套的 Blox 标记，然后添加必需的标记属性或嵌套的属性标记。在以下 PresentBlox 示例中，添加了具有两个标记属性（`dataSourceName` 和 `query`）的嵌套的 DataBlox 标记:

```
<blox:present id="uniqueName">
<blox:data
dataSourceName="definedDataSource"
query="query"/>
</blox:present>
```

如果您想要更改 PresentBlox 的缺省图表类型设置，则必须添加嵌套的 ChartBlox 标记，从而使用 ChartBlox `chartType` 属性来定义备用图表类型。在以下示例中，PresentBlox 现在将显示折线图而不是缺省的条形图:

```
<blox:present id="uniqueName">
<blox:data
dataSourceName="definedDataSource"
query="query"/>
<blox:chart chartType="Line"/>
</blox:present>
```

要定制 Blox 以满足用户需求，您将使用标记属性来添加和修改许多 Blox 属性。除了标记属性以外，某些 Blox 还需要它们自己的特殊属性标记以定义属性。下两节讨论如何使用这些“嵌套的”属性标记来定义一些特定的 Blox 样式属性。

使用样式属性标记来设置 Blox 属性

虽然大多数 Blox 属性使用标记属性显示和定义，但是，其他属性（包括所有样式以及带下标属性（使用下标值以允许同一个 Blox 包含多个属性实例）相对较为复杂，并且大多包括也需要被定义的子属性。在这些属性中，其中一些属性要求使用它们自己的标记，而另一些属性则可以用作 Blox 的属性标记或标记属性。

下表列示了使用属性标记显示并定义的所有不带下标 Blox 属性（不能具有多个实例（每个实例都由下标值标识）的属性）：

属性	相关联的子属性或特性	适用于
titleStyle	前景字体	ChartBlox
footnoteStyle	前景字体	ChartBlox
labelStyle	前景字体	ChartBlox
axisTitleStyle	前景字体	ChartBlox

对于上面列示的每个样式属性，都有一个 Blox 属性标记定义了该属性及其相关联的子属性。标记属性用来设置样式属性的各个子属性。与嵌套的 Blox 定义标记相同，Blox 属性标记嵌套在它们的属性所应用于的 Blox 中。但是，与 Blox 定义标记不同，这些标记不是用来定义对象的，而是用来定义 Blox 的属性。

以下 GridBlox 标记示例包括若干个 GridBlox 标记属性，这些属性包括 id、bandingEnabled 和 defaultCellFormat。在 GridBlox 标记的主体中，您可以看到使用了一个嵌套的属性标记（<blox:titleStyle>）来定义网格中所有单元格的外观:

```
<blox:grid id="myGridBlox"
bandingEnabled="false"
defaultCellFormat="#,###.00"/>
  <blox:titleStyle
foreground="red"
font="Helvetica:10"/>
</blox:grid>
```

属性标记使开发者能够编码复杂的属性，而不必将所有这些子属性放在一个值字符串中。这有助于提高可读性和进行编码，从而有助于降低编码错误的可能性。并且，由于长长的值字符串不能包含换行符，所以它们不能象上述示例那样很好地被格式化。如果使用一个值字符串，则以上示例将变为:

```
<blox:grid id="myGridBlox"
bandingEnabled="false"
defaultCellFormat="#,###.00"
titleStyle="foreground=red,font=Helvetica:10;"/>
</blox:grid>
```

虽然采用一致的方法可以使调试更为容易，但是，使用标记属性还是嵌套的属性标记来定义样式是您的个人选择。并且，在样式标记属性中使用嵌套的引号是一个棘手问题。

在 *Developer's Reference* 一书中，有关每个样式属性的内容都对样式属性标记的用法作了进一步的说明。

使用属性标记来设置带下标 Blox 属性

带下标 Blox 属性也使用 Blox 属性标记进行定义，这些属性包括下标值以允许在一个 Blox 中使用这些属性的多个实例。与前面的样式属性标记不同，这些属性标记包括 index 属性以允许在一个 Blox 标记中处理同一个标记的多个实例。

在带下标属性标记与不带下标属性标记之间有两项重要的差别:

- 在父 Blox 标记中，可以有同一个带下标属性标记的多个实例。
- 除非在属性中显式地定义下标值，否则，带下标 Blox 属性标记在代码中的放置顺序将影响结果。

带下标属性标记具有一个公共 index 属性，当存在多个实例时，此属性用来定义解释顺序。index 属性允许:

- 对这些属性标记中定义的带下标属性编制脚本
- 指定解释顺序，因此，您不需要在嵌套的 Blox 中重新安排这些属性标记的顺序（虽然保持它们按顺序排列有助于更好地解释期望的行为）。

如果未定义 index 属性，则将自动地指定隐式下标值。第一个 index 属性将被指定值 1。如果您打算使用多个带下标属性标记并且将对这些标记编制脚本，则应该考虑对标记添加 index 属性并指定可以在代码中检查的值。这有助于确保对正确的标记编制脚本。

下表列示了所有带下标属性、它们的子属性以及它们所属的 Blox:

带下标属性	相关联的子属性或特性	适用于
cellAlert	index enabled condition value value2 description font foreground background apply format align valign link image image_align scope	GridBlox
cellFormat	index format scope	GridBlox
cellEditor	index scope	GridBlox
cellLink	index description image image_align link scope	GridBlox
generationStyle	index foreground background font align valign	GridBlox

正如前面提到的那样，在对带下标属性标记编制脚本时，该属性标记具有隐式的或定义的 index 属性。在以下示例中，GridBlox 具有两个 GridBlox cellAlert 标记，但这两个标记都未定义 index 属性:

```
<blox:grid id="myGridBlox">
  <blox:cellAlert
condition="GT"
value="50"
scope="{Scenario:Variance}"/>
```

```
  <blox:cellAlert
condition="GT"
value="50"
scope="{Scenario:Variance}"/>
</blox:grid>
```

要修改第二个 <blox:cellAlert> 标记，Java 方法可能如下所示:

```
myGridBlox.setCellAlert(2,"condition=GT,value=50, background=red,
scope={Scenario:Variable}")
```

尽管 GridBlox 标记未显式地指定 index 属性，但是第二个 cellAlert 属性的 index 属性自动设置为 2。虽然这样做也是可以的，但最好通过设置显式的 index 属性来定义单元格提醒，如下所示:

```
<blox:grid id="myGridBlox">
<blox:cellAlert index="1"
condition="GT"
value="50"
scope="{Scenario:Variance}"/>
<blox:cellAlert index="2"
condition="GT"
value="50"
scope="{Scenario:Variance}"/>
</blox:grid>
```

这样做，就可以更加容易地了解每个单元格提醒的下标值，当定义了大量单元格提醒时尤其如此。

控制 Blox 组件的可视性

公共 Blox 属性 visible 允许开发者控制 JSP 页面上 Blox 的显示。此属性可以应用于下列 Blox: ChartBlox、DataBlox、DataLayoutBlox、GridBlox、PageBlox、PresentBlox、RepositoryBlox 以及嵌套的 ToolbarBlox。缺省情况下，这些 Blox 的 visible 值是 true。可以将 visible 属性设置为 false，然后在完成了一些处理逻辑后使用 <blox:display> 标记来显示该 Blox。

注: 当使用 DHTML 客户机时，将在客户机页面上创建 Blox JavaScript 对象，从而允许该页面使用客户机 API 来与 DB2 Alphablox 进行通信。但是，如果 visible 属性设置为 false，则不会创建客户机端 JavaScript Blox 对象。

可以在 *Developer's Reference* 一书中找到关于 visible 标记属性的详细信息。

在显示之前处理逻辑

在更高级的分析应用程序中，您可能会发现，在 JSP 页面上显示 Blox 之前，需要在 scriptlet 中使用Java 方法来处理一些业务逻辑。在这些情况下，可以将 Blox 的 visible 属性设置为 false，然后使用 Blox display 标记（<blox:display>）来控制 Blox 的可视性。

在 JSP 页面上显示视图之前，可以将 Blox 的 visible 属性设置为 false 并指定处理逻辑代码，然后在完成处理后显示 Blox。

以下示例显示了将 visible 属性设置为 false 的 PresentBlox，接着显示了具有一些使用 Java 的处理逻辑的 scriptlet，最后显示了 <blox:display> 标记，该标记导致该 PresentBlox 显示在页面上:

```
<blox:present id="myPresentBlox"
visible="false"
...
/>
<%
your processing logic would go here
%>
<blox:display bloxRef="myPresentBlox"/>
```

在此示例中，如果未将 PresentBlox 的 visible 属性设置为 false，则 JSP 容器将在页面上显示两个 Blox，其中一个在处理逻辑完成前显示，另一个在处理逻辑完成后显示。并且，第一个 PresentBlox 不会显示所执行的逻辑的效果。

对于诸如 RepositoryBlox 和 DataBlox 之类的在页面上始终不可视的 Blox 来说，将 visible 属性设置为 false 不会有任何作用。由于这两个 Blox 永远不可视，所以 visible 属性将被忽略。

在多个页面上显示 Blox

当需要在一个页面上或者在框架集中的一个框架上创建 Blox，但需要在另一个页面中显示同一 Blox 时，使用 <blox:display> 标记就十分方便。此标记在两种常见情况下很有用: 当页面上有一个 Blox 并且您希望使用定制页面来进行打印或导出到 Microsoft Excel 时。例如，在将 Blox 视图导出到 Microsoft Excel 时，可以在一个带有“以 Excel 格式导出”按钮的页面上定义 Blox。当用户单击该按钮时，可以将一个页面装入到 Excel 中，该页面仅显示网格和图表，而不显示原始页面中不必要的文本或按钮。

并且，一旦在会话期间将 Blox 实例化，就可以使用 <blox:display> 标记来将它的当前视图提供给其他页面使用。

要了解关于 <blox:display> 标记的详细信息，请参阅 *Developer's Reference* 一书以及本指南第 141 页的『使用 <blox:display> 标记来创建定制打印页面』。

Blox 实用程序标记

某些 Blox 标记不是用来定义页面上显示的 Blox，而是用来提供对其他功能的访问。以下是关于 Blox 实用程序标记的简要描述，您可以在 General Blox Reference Information 一书中找到详细信息。

Blox header 标记

先前在 第 31 页的『使用 Blox header 标记』 中对 Blox header 标记（<blox:header>）作了描述。

Blox context 标记

<blox:bloxContext> 标记类似于 <blox:header> 标记，它们都根据实际请求和响应类型（HTTP 或基于 portlet）创建适当的 BloxRequest、BloxResponse 和 BloxContext 对象。然而，它不输出任何主题或 JavaScript 代码进行显示。使用此标记的一个示例是，当您在某个 JSP 页面上不具有 Blox，但需要在包含其他具有 Blox 的 JSP 页面的 portlet

上访问 Blox 上下文信息。因为 <blox:bloxContext> 标记和 <blox:header> 标记都尝试声明相同的变量，所以它们不能在 JSP 页面中共存。

Blox debug 标记

可以将另一个特殊标记 Blox debug 标记（<blox:debug>）添加到 JSP 页面中，以便将有用的调试信息发送至系统控制台。

可以在中“故障诊断”一节找到有关 Blox debug 标记的更多用法信息。要了解有关使用系统控制台的信息，请参阅《*管理员指南*》。

Blox display 标记

如果想在完成一些处理逻辑后显示 Blox 或者在与最初创建 Blox 的不同页面上显示 Blox，则先前讨论的 <blox:display> 标记就会特别有用。可以在第 137 页的第 15 章，『显示数据』以及 *Developer's Reference* 中的 The <blox:display> Tag 主题中找到有关使用 <blox:display> 标记的详细信息。

资源束标记

资源束使您可以开发可以本地化并翻译为不同语言的应用程序。DB2 Alphablox 为 Java ResourceBundle 类提供了包装程序标记，以允许您从应用程序的资源束中访问特定于语言环境的资源。这些包装程序标记如下:

- <blox:resourceBundle>
- <blox:message>
- <blox:messageArg>

可以在 *Developer's Reference* 一书中的 Resource Bundle Related Tags 一节中找到关于这些标记及其用法的详细信息。

使用标准 JSP 语法

可以使用标准的 JSP 语法（而不是使用 Blox 定制标记）来定义 Blox。正如先前内容所讨论的那样，Blox 标记几乎始终能够提供最佳的 Blox 定义方法。但是，在某些情况下，您可能会发现唯一的可选方法是使用标准 JSP 语法。

如果您以前未曾使用标记库而只使用标准 JSP 语法来进行编码，则可能希望使用您所熟悉的语法，而不想使用 Blox 标记。但是，基于本主题的前部分描述的原因，在您决定使用标准 JSP 语法之前，您应该尝试使用 Blox 定制标记。

注: 如果在同一个页面上既使用标准 JSP 语法又使用 Blox 标记，则需要在页面顶部添加下面这两行内容:

```
<%@ taglib uri="bloxtld" prefix="blox" %>
<%@ page import="com.alphablox.blox.*" %>
```

注: Blox 标记通过将 scope 设置为 session 来定义 Blox（Java bean）。如果使用标准 JSP 语法，则几乎总是应该使用具有会话作用域的 bean（在本实例中，它是一个 Blox）。useBean 语法的缺省值将 scope 设置为 page。以下是 useBean 的一个示例:

```
<jsp:useBean id="regionsPresentBlox"
class="com.alphablox.blox.PresentBlox"
scope="session">
```

后续步骤

在本节中，您学习了如何使用核心 Blox 标记来定义表示 Blox 及其属性。您还学习了有关 Blox 实用程序标记（包括 <blox:display>、<blox:debug> 和 <blox:header>）的内容。要了解有关这些标记的语法和用法的特定内容，请参阅 *Developer's Reference* 一书以及本指南。

在本《*开发者指南*》的余下部分中，我们将向您介绍常用任务以及一些潜在的解决方案。在您为用户开发复杂的分析应用程序的过程中，《*开发者指南*》、*Developer's Reference* 以及一些优秀的 JavaServer Pages 参考资料应该能够为您提供非常大的帮助。

第 6 章 Blox 表单标记库

Blox 表单标记库包括 FormBlox 和其他标记，这些标记可以生成具备内置增强功能的 HTML 表单元素。某些标记将自动地为数据源、维和维成员生成选择列表。其他标记可以用来管理单选按钮和复选框或者用来创建用于导航和其他用途的树形控件。并且，当使用这些标记时，在会话期间将对状态持久性进行处理。因此，您不需要编写附加的 Java 或 JavaScript 代码来在用户浏览会话期间管理选项持久性。复选框将保持它们的被选中状态，最后被选中的单选按钮仍被选中，树形菜单将保持它们的状态，即使用户在同一会话期间离开该页面并且稍后返回亦如此。

使用 Blox 表单标记库

FormBlox 及相关标记在 `bloxform.tld` 文件中定义。当您创建新的 DB2 Alphablox 应用程序时，此文件将自动包括在以下目录中:

<application_dir>/WEB-INF/tlds/bloxform.tld

注: 如果找不到该 TLD 文件，或者意外地删除了该文件，则可以在以下目录中找到这个 TLD 文件的当前版本的副本:

```
<db2alphablox_dir>/bin/
```

FormBlox 及相关标记全都在 `bloxform.tld` 文件中定义。当您创建新的 DB2 Alphablox 应用程序时，此文件将自动包括在以下目录中:

```
/<applicationContext>/WEB-INF/tlds/bloxform.tld
```

要使用 Blox 表单标记，必须在 JSP 页面顶部包括以下 taglib 伪指令，这样才能识别标记库。

```
<%@ taglib uri="bloxformtld" prefix="bloxform" %>
```

FormBlox 组件概述

下面简要描述了 FormBlox 组件，并描述了一些简单的用法示例。FormBlox 组件是使用您作为开发者有权访问的那些低级别 Blox UI 组件构建的，但是，已经为您完成了构建这些简单易用的 Blox 组件的工作。

可以在 *Developer's Reference* 一书的 Blox Form Tags Reference 一节中以及在 *Blox API Javadoc* 文档中找到关于 FormBlox API 以及其他与表单相关的标记的详细信息（包括语法、用法和示例）。

FormBlox 组件类别

可以将 Blox 表单标记库的 FormBlox 组件分为四类: 表单控件、元数据选择列表、时间模式选择列表以及树形控件。以下是这些组件组的简要概述。

基本表单控件

这组 FormBlox 组件创建基本的 HTML 表单控件，包括复选框、文本字段或文本区域、单选按钮以及选择列表。在很大程度上，这些 Blox 组件与标准 HTML 元素提供的功能类似，但前者还具有在会话期间保持状态的优点。当用户事件（如选取复选框或从选择列表中选择项）发生时，将把更改了的值发送至适当的对象，而不必刷新页面。

FormBlox 组件
: **描述**

CheckBoxFormBlox
: 使用 HTML `<input type="checkbox">` 标记来创建复选框（独立的复选框或成组的复选框），以便选取或取消选取项。

EditFormBlox
: 使用 HTML `<input type="text">` 或 `<textarea>` 标记来创建文本字段或文本区域。

RadioButtonFormBlox
: 使用 `<input type="radio">` 标记来创建单选按钮表单控件。

SelectFormBlox
: 使用 HTML `<select>` 和 `<option>` 标记来创建下拉选择列表和滚动选择列表。

元数据选择列表

这组 FormBlox 组件创建专用的 HTML 选择列表，这些选择列表根据数据源元数据来生成选择列表选项。这些选择列表包括数据源选择列表、多维数据库选择列表、立方体选择列表、维选择列表和成员选择列表。由于这些列表是动态生成的，因此，您不需要担心要记住添加或除去列表选项。

与其他基本 HTML 选择列表不同，这些元数据选择列表将在用户会话期间保持状态。当用户事件（如选取复选框或从选择列表中选择项）发生时，将把更改了的值发送至适当的对象，而不必刷新页面。

FormBlox 组件
: **描述**

DataSourceSelectFormBlox
: 为 DB2 Alphablox 数据源创建动态生成的 HTML 选择列表。

CubeSelectFormBlox
: 创建动态填充的 HTML 选择列表，该选择列表包含指定的 DB2 Alphablox 数据源中的可用立方体。

DimensionSelectFormBlox
: 为指定的立方体中的可用维创建动态填充的 HTML 选择列表。

MemberSelectFormBlox
: 创建动态填充的 HTML 选择列表，该选择列表包含所选维中的可用成员。

时间模式选择列表

与时间模式相关的 Blox 表单标记允许开发者动态地为用户生成选择列表，以便选择常用的业务时间段和时间单位。这些选择列表可用来驱动用户所看到的分析视图。以下是 TimePeriodSelectFormBlox 和 TimeUnitSelectFormBlox 这两个主要的时间模式标记的概述。

Blox 组件

描述

TimePeriodSelectFormBlox

创建用于提供常用业务时间段的选择列表，这些时间段包括本周、本月、本月迄今为止、本季度迄今为止、本年迄今为止、前季度、前两个季度、前四个季度、前两个月、前三个月、前六个月、去年以及前两年。可以包括定制时间段。

TimeUnitSelectFormBlox

创建用于提供时间单位选项的 HTML 选择列表，这些时间单位选项包括天、星期、月、季度和年。

树形控件

此类别包括一个项，即 TreeFormBlox。TreeFormBlox 创建由文件夹和项组成的树形控件。这些文件夹和项（当被选取时）可以具有相关联的操作。TreeFormBlox 可用来创建分层选择列表和导航菜单。它还可以使用 HTML 表单的 POST 方法。可以在树形控件内拖放项和文件夹（如果允许这样做的话）。

Blox 组件

描述

TreeFormBlox

创建 DHTML 树形控件，此控件可用于创建分层选择列表以及导航菜单。

获取和设置 Blox 和 JavaBeans 组件属性

Blox 表单标记库包括两个嵌套的标记，即 `<blox:getChangedProperty>` 和 `<blox:setChangedProperty>`。`<blox:getChangedProperty>` 对于在两个 FormBlox 之间传递值（即将它们链接到一起）来说非常有用。可以在两个 FormBlox 之间、在 FormBlox 与另一个 Blox 组件（例如 DataBlox）之间或者在 FormBlox 与其他定制 JavaBeans 组件之间使用 `<blox:setChangedProperty>`。`<blox:setChangedProperty>` 还包括 `callAfterChange` 属性，此属性可以在更改发生后调用服务器端 Java 方法。在对意外的行为进行调试时，`debugEnabled` 布尔属性非常有用。

以下是这两个嵌套的 FormBlox 标记的总结:

Blox 组件

描述

`<bloxform:getChangedProperty>`

嵌套在一个 FormBlox 中以指定另一个 FormBlox 作为目标。目标 FormBlox 中的指定属性值将被传递到具有此标记的 FormBlox。

`<bloxform:setChangedProperty>`

嵌套在 FormBlox 中，以指定任何其他 JavaBeans 组件（包括其他 FormBlox、Blox 或定制 bean）作为目标。拥有此标记的 FormBlox 中的指定属性值将被传递到目标 bean。

可以在 *Developer's Reference* 一书的 Blox Form Tags Reference 一节中以及在 *Blox API Javadoc* 文档中找到关于 `<bloxform:getChangedProperty>` 和 `<bloxform:setChangedProperty>` 标记以及其他与表单相关的标记的详细信息（包括语法、用法和示例）。

FormBlox 事件模型

如果您认为有必要的话，可以为 FormBlox 组件编写自己的事件处理程序。每当控件更改时，简单的 FormBlox 事件模型就会提供该控件在更改前以及更改后的值。FormBlox 组件并不是全面的开发解决方案，它只是提供了一个简单的事件模型，可用来处理在大多数情况下都会发生的基本事件。如果需要创建更复杂的组件以便能够满足更复杂的需求，您也可以使用 Blox UI 模型组件（它们支持更为丰富的事件模型）来构建自己的事件模型。

使用 FormBlox 标记的示例

“Blox 样本程序”以及别处的许多示例都使用了 FormBlox 组件。并且，在本指南中，提供了许多使用 FormBlox 的示例。FormBlox 组件对于开发者来说是一个相当优秀的资源，它能够处理许多涉及动态列表生成编码和状态管理编码的繁重任务。下面着重列举了一些使用各种 FormBlox 组件的示例。

使用 DataSourceSelectFormBlox 的特别分析

在“Blox 样本程序”的“使用 FormBlox 和逻辑 Blox”部分中，有一个示例使用 DataSourceSelectFormBlox 来创建一个简单视图以供用户选择数据源，然后使用具有全面交互功能的 PresentBlox 来使用指定的立方体进行分析。可以在第 100 页的『使用 DataSourceSelectFormBlox 来设置不同的数据源』的“连接至数据”主题中找到对此示例中使用的代码的逐步描述。

查询构建器

查询构建器位于 DB2 Alphablox 管理页面中的“组装”选项卡下面的“工作台”部分中，这个界面用来帮助开发者生成多维查询语句以便与 DB2 OLAP Server、Hyperion Essbase 和 Microsoft Analysis Services 配合使用。在后台，如果您查看查询构建器的源代码，您会找到 FormBlox 组件（包括 DataSourceSelectFormBlox 和 CubeSelectFormBlox）的使用实例。

使用 FormBlox 来指定报告选项

在“Blox 样本程序”的“与数据进行交互”部分下面的“使用 HTML 表单元素”示例中，您可以找到另一个优秀的 FormBlox 组件使用示例。在此示例中，使用了 RadioButtonFormBlox 和 CheckboxSelectFormBlox 来选择报告选项。

使用 TreeFormBlox 的导航菜单

当您打开“Blox 样本程序”应用程序时，在左框架中打开的导航菜单使用 TreeFormBlox。请参阅“Blox 样本程序”应用程序中的 `navigation.jsp` 文件以获取使用这个 FormBlox 创建的大型导航菜单的示例。

FastForward 应用程序中的报告模板

Alphablox FastForward 应用程序广泛地使用 FormBlox 组件来构建报告模板以及将其用于导航菜单。要查阅 Alphablox FastForward 应用程序中使用的代码，请创建该应用程序的新副本。第 237 页的第 25 章，『使用 DB2 Alphablox FastForward』 也描述了使用 FormBlox 组件的代码示例。

第 7 章 Blox 逻辑标记库

Blox 逻辑标记库包括更多简单易用的标记，这些标记可以用来处理时间段选择、在用户不需要了解如何创建 Essbase 报告脚本（用于与 DB2 OLAP Server 和 Essbase 配合使用）或 MDX 语句（用于与 Microsoft Analysis Services 配合使用）的情况下处理多维数据库查询以及处理成员安全性。

使用 Blox 逻辑标记库

Blox 逻辑标记库中的标记在 `bloxlogic.tld` 文件中定义。当您创建新的 DB2 Alphablox 应用程序时，此文件将自动包括在应用程序的以下目录中:

<application_dir>/WEB-INF/tlds/bloxlogic.tld

注: 如果找不到该 TLD 文件，或者意外地删除了该文件，则可以在以下目录中找到这个 TLD 文件的当前版本的副本:

```
<alphabloxDirectory>/bin/
```

要在页面上访问 Blox 逻辑标记库，需要包括以下 JSP taglib 伪指令:

```
<%@ taglib uri="bloxlogictld" prefix="bloxlogic" %>
```

Blox 逻辑标记库组件

Blox 逻辑标记库中的主要 Blox 逻辑组件包括 MDBQueryBlox、MemberSecurityBlox 和 TimeSchemaBlox，概述如下。

可以在 *Developer's Reference* 一书的 Business Logic Blox and TimeSchema DTD Reference 一节以及在 *Blox API Javadoc* 文档中找到关于 Blox 逻辑标记库组件的详细信息（包括语法、用法和示例）。

逻辑 Blox
: **描述**

MDBQueryBlox
: MDBQueryBlox 是多维数据查询的对象表示。它允许您处理 MDB 查询，而不必使用与数据源相关联的查询语言。通过使用 `<bloxlogic:mdbQuery>` 标记或它的 API，可以处理各个查询部分，如轴的元组的变化部分。在 MDBQueryBlox 中进行更改之后（通过调用它的 `changed()` 方法），它的源 DataBlox 将被重新执行的数据查询自动更新。

MemberSecurityBlox
: MemberSecurityBlox 提供在给定维上用户有权访问的成员列表。它通过根据指定的 MemberSecurityFilter 对 DataBlox 执行 suppressNoAccess 来构造该列表。要设置 MemberSecurityFilter，请使用 `addMember()` 或 `setMember()` 方法来指定维以及该维中的成员。

TimeSchemaBlox
: 根据时间模式的定义来为给定的数据源创建一个时间表。通过使用 TimeSchema

数据类型定义（DTD），可以通过指定下列内容来定义如何构造 Time 维: Time 维的名称、代级别（Year、Quarter、Month 和 Week）、立方体中时间段的起始时间、是应该应用正常日历时间还是应该应用星期时间以及年份的长度是否特殊（如 48 个星期的年份）等。

使用 MDBQueryBlox 组件来选择产品

通过简单易用的标记，可以使用 MDBQueryBlox 来处理多维查询，而不必编写任何特定于 DB2 OLAP Server、Hyperion Essbase 或 Microsoft Analysis Services 的逻辑。

在“Blox 样本程序”中的“使用逻辑 Blox”下面，有一个将 MDBQueryBlox 与 PresentBlox 配合使用以允许用户从选择列表中选择产品的示例（该选择列表是使用 MemberSelectFormBlox 创建的）。这里，我们快速浏览一遍创建类似页面的主要步骤:

1. 为要在页面上使用的 Blox 标记库添加 JSP `taglib` 伪指令。

```
<%@ taglib uri="bloxtld" prefix="blox" %>
<%@ taglib uri="bloxlogictld" prefix="bloxlogic" %>
<%@ taglib uri="bloxformtld" prefix="bloxform" %>
```

2. 定义将要使用的 DataBlox，将 `visible` 属性设置为 false 并通过将 `useAliases` 设置为 `true` 来允许使用别名成员名。

```
<blox:data id="dataBlox"
visible="false"
dataSourceName="QCC-Essbase"
useAliases="true" />
```

3. 指定要在列轴、行轴和页面轴上使用的元组列表。

```
<!-- Column Time tuples -->
<bloxlogic:tupleList id="timeTuples">
  <bloxlogic:dimension>All Time Periods</bloxlogic:dimension>
  <bloxlogic:tuple>
    <bloxlogic:member>Qtr 1 01</bloxlogic:member>
      </bloxlogic:tuple>
  <bloxlogic:tuple>
    <bloxlogic:member>Qtr 2 01</bloxlogic:member>
      </bloxlogic:tuple>
    </bloxlogic:tupleList>
<!-- Column Measures tuples -->
<bloxlogic:tupleList id="measuresTuples">
  <bloxlogic:dimension>Measures</bloxlogic:dimension>
  <bloxlogic:tuple>
    <bloxlogic:member>Sales</bloxlogic:member>
      </bloxlogic:tuple>
  <bloxlogic:tuple>
    <bloxlogic:member>Sales % of All Locations</bloxlogic:member>
      </bloxlogic:tuple>
    </bloxlogic:tupleList>
<!-- Page tuples -->
<bloxlogic:tupleList id="pageTuples">
  <bloxlogic:dimension>Scenario</bloxlogic:dimension>
  <bloxlogic:dimension>All Products</bloxlogic:dimension>
  <bloxlogic:tuple>
    <bloxlogic:member>Actual</bloxlogic:member>
    <bloxlogic:member>All Products</bloxlogic:member>
      </bloxlogic:tuple>
</bloxlogic:tupleList>
```

4. 添加一个 MemberSelectFormBlox，以便用户能够从 Product 维中选择产品。

```
<!-- MemberSelect FormBlox for the Product dimension. On change event,
     MemberSelect FormBlox will change the pageTuples.
-->
<bloxform:memberSelect id="selector"
visible="false"
dataBloxRef="dataBlox"
dimensionName="All Products"
rootMemberName="100"
selectedMemberName="100">
   <bloxform:setChangedProperty
formProperty="selectedMembers"
targetRef="pageTuples"
targetProperty="listFromMetadataMembers"
callAfterChange="changed"/>
</bloxform:memberSelect>
```

5. 添加一个 MDBQueryBlox。

```
<!-- The MDBQuery creates a query from the 2 column tuples,
the page tuple and the row query fragment
-->
<bloxlogic:mdbQuery id="query"
dataBloxRef="dataBlox">
<bloxlogic:axis type="columns">
<bloxlogic:crossJoin>
<bloxlogic:tupleList tuplesRef="timeTuples" />
<bloxlogic:tupleList tuplesRef="measuresTuples" />
</bloxlogic:crossJoin>
</bloxlogic:axis>
<bloxlogic:axis type="rows"
queryFragment='<ROW ("All Locations") <ICHILD "All Locations"' />
<bloxlogic:axis type="pages">
<bloxlogic:tupleList tuplesRef="pageTuples" />
</bloxlogic:axis>
</bloxlogic:mdbQuery>
```

6. 添加 PresentBlox，它引用先前指定的 DataBlox。

```
<blox:present id="presentBlox"
visible="false">
<blox:data
bloxRef="dataBlox" />
</blox:present>
```

7. 添加页面的余下部分以显示该 Blox 并安排视图布局。

```
<html>
<head>
   <blox:header />
</head>
<body>
<table width="100%" height="400">
  <tr>
<td align="center" height="10">Product: <blox:display
bloxRef="selector" /></td>
  </tr>
  <tr>
    <td>
<blox:display bloxRef="presentBlox" width="100%" height="100%" />
    </td>
  </tr>
</table>
 </body>
</html>
```

请参阅“Blox 样本程序”的“使用 Blox 逻辑标记”中的内容以获取完整代码以及一个使用 MDBQueryBlox 和 MemberSelectFormBlox 的有效示例。

使用 MemberSecurityBlox 来列示立方体成员

MemberSecurityBlox 标记允许根据访问许可权来列示维中的成员。它使用 DataBlox suppressNoAccess 属性来过滤成员，并且可以采用多个根成员。它还可以用来指定多对 dimension:members 以进行过滤。

以下是 MemberSecurityBlox 的使用方式示例:

1. 在文件顶部添加 JSP page 伪指令以指定需要访问的 Java 类。

```
<%@ page import="com.alphablox.blox.logic.MemberSecurityFilter" %>
```

2. 为将要使用的 Blox 标记库添加 JSP taglib 伪指令。

```
<%@ taglib uri="bloxtld" prefix="blox"%>
<%@ taglib uri="bloxformtld" prefix="bloxform"%>
<%@ taglib uri="bloxlogictld" prefix="bloxlogic"%>
```

3. 记住将 <blox:header> 标记添加到页面的 head 部分中。

```
<head>
   <blox:header />
</head>
```

4. 将 DataBlox 添加到页面中。

```
<blox:data id="myDataBlox"
query="" dataSourceName="QCC-MSAS" />
```

5. 添加 MemberSecurityBlox 标记。

```
<bloxlogic:memberSecurity id="memberSecurityMsas"
dataBloxRef="myDataBlox"
cubeName="QCC"
dimensionName="[Products].[Category]">
  <bloxlogic:memberSecurityFilter
dimensionName="[Measures]"
memberName="[Measures].[Sales]" />
  <bloxlogic:memberSecurityFilter
dimensionName="[Measures]"
memberName="[Measures].[COGS]" />
</bloxlogic:memberSecurity>
```

6. 添加 SelectFormBlox。

```
<bloxform:select id="members"
visible="false"
multiple="true"
size="5" >
<%
  members.setItems(memberSecurityMsas.getDisplayMemberNames());
%>
</bloxform:select>
```

7. 在页面中要显示选择列表的位置添加 <blox:display> 标记。

```
<body>
<blox:display bloxRef="members" />
</body>
```

TimeSchemaBlox 组件

TimeSchemaBlox 根据时间模式的定义来为给定的数据源创建一个时间表。通过使用 TimeSchema 数据类型定义（DTD），可以通过指定下列内容来定义 Time 维的结构: Time 维的名称、代级别（Year、Quarter、Month 和 Week）、立方体中时间段的起始时间、是应该应用正常日历时间还是应该应用星期时间以及年份的长度是否特殊（如 48 个星期的年份）等。

`<bloxlogic:timeSchema>` 标记创建一个 TimeSchemaBlox，TimePeriodSelectFormBlox、TimeUnitSelectFormBlox 或 MDBQueryBlox 可以引用该 TimeSchemaBlox 来创建时间段选择列表或处理数据查询。

应该将包含 TimeSchema 定义的 XML 文件命名为 `timeschema.xml` 并存储在应用程序的 `WEB-INF` 目录中。用来定义 TimeSchema XML 的数据类型定义（DTD）是在 TimeSchema XML DTD 中描述的。

可以在 *Developer's Reference* 一书的 Business Logic Blox and TimeSchema DTD Reference 一节中找到关于 TimeSchemaBlox 和 TimeSchema XML DTD 的语法和用法详细信息。

以下代码段显示了由 TimePeriodSelectFormBlox 使用的 TimeSchemaBlox。缺省情况下，TimePeriodSelectFormBlox 向用户显示可以从中进行选择的时间段列表。进行选择后，当 `changed()` 方法被调用时，histTuples 的 `listFromMetadataTuples` 属性将相应地更改。

```
<blox:data id="dataBlox"
dataSourceName="QCC-MSAS"/>
<bloxlogic:timeSchema id="timeSchema"
name="MSAS"
dataBloxRef="dataBlox" />
<bloxlogic:tupleList id="histTuples">
    <bloxlogic:dimension
list="<%=timeSchema.getDimensions()%>">
    </bloxlogic:dimension>
    </bloxlogic:tupleList>
<bloxform:timePeriodSelect id="historySelector"
timeSchemaBloxRef="timeSchema"
selectedSeriesString="SEQUENCE(QUARTER,-1,1)(QUARTER)"
visible="false">
   <bloxform:setChangedProperty
formProperty="tuples"
targetRef="histTuples"
targetProperty="listFromMetadataTuples"
callAfterChange="changed"/>
</bloxform:timePeriodSelect>
```

第 8 章 Blox Portlet 标记库

Blox Portlet 标记库提供一些标记，这些标记使您可以在 Blox 或 UI 组件中定义 portlet 链接或操作链接。这些链接允许您使用 Portlet API 进行 portlet 至 portlet 的消息传递。本节介绍了 Blox Portlet 标记库并显示了如何使用它将基于 ClientLink 的链接或操作链接连接到 portlet 中的顶级 Blox 或 UI 组件。

注: 顶级 Blox 是嵌套其他 Blox 的最外层 Blox。例如，如果您有一个嵌套了 GridBlox 和 ChartBlox 的 PresentBlox，则该 PresentBlox 是顶级 Blox。

Blox portlet 标记概述

您经常想要在一个 portlet 中单击链接以在另一个 portlet 中触发更新。Blox UI 模型的 `ClientLink` 对象使您可以将链接连接到特定的 Blox UI 组件。当单击该组件时，浏览器将处理这个基于 URL 的链接。`ClientLink` 对象导致视图层使用它自己的逻辑来处理用户的单击，而不是将该操作发回给服务器。除了要装入的 URL 之外，`ClientLink` 还允许您指定要装入新页面的目标窗口和浏览器窗口功能部件字符串（例如，`features="scrollbars=yes,width=300,height=300"`）。然而，在门户网站环境中，该链接将仅在第一次使用时起作用。在链接触发页面重新装入之后，由于门户网站服务器处理每个请求的方式而使得该 portlet 链接将过时。以后单击该链接将不会提交实际操作。Blox Portlet 标记库使您可以添加 PortletLinkDefinition 或 ActionLinkDefinition 来提供以下功能:

- 如果添加了 PortletLinkDefinition，则使用它来创建 PortletLink 对象。然后使用 PortletLink 对象来定义实际链接以调用具有指定的参数值的 URI。每次刷新页面时，portlet 将重新编码该 URL，以防止它过时。
- 如果添加了 ActionLinkDefinition，则可以使用它来创建 PortletLink。或者，通过将操作名称传递到 `BloxResponse.getActionURL()`，可以使用它来获取此链接的 portlet URI。

当将 PortletLinkDefinition 或 ActionLinkDefinition 与 Blox 组合在一起时，该定义将分配给 Blox，而 PortletLink 用来生成 ClientLink 以在 Blox UI 模型中使用。可以在任何数据表示 Blox、FormBlox、ReportBlox 或任何具有单击事件概念的 Blox UI 模型组件内使用 Blox Portlet 标记。虽然 PortletLinkDefinition 和 ActionLinkDefinition 都可以用来创建 PortletLink，但是 ActionLinkDefinition 也允许您为此链接定义设置操作名称以创建 PortletURI。然而，在已使用此定义生成 PortletLink 之后，不能将操作名称传递到 `BloxResponse.getActionURL()`。

有关基本代码结构和如何将 Blox 添加至 portlet JSP 的开发技巧的详细信息，请参阅『入门』一节中的 portlet 教程。

使用 Blox Portlet 标记库

在 bloxform.tld 文件中定义 Blox Portlet 标记库中的标记。对于 portlet 项目，您应使用 portlet 开发工具来正确地设置结构和部署描述符文件。有关如何创建项目以便正确地引用和指定所有需要的 Alphablox 标记库和 servlet 映射的信息，请参阅『入门』一节中有关使用 Rational Application Developer 来构建 portlet 的教程。有关如何将 Blox 组件添加到 portlet 的详细信息，请参阅『入门』一节中的 portlet 教程。

注: 如果找不到该 TLD 文件，或者意外地删除了该文件，则可以在以下目录中找到这个 TLD 文件的当前版本的副本:

```
<alphabloxDirectory>/bin/
```

通过将 <bloxportlet:actionLinkDefinition> 或 <bloxportlet:portletLinkDefinition> 标记嵌套在 Blox 标记中，将把 Blox Portlet 标记添加到 Blox 或 Blox UI 组件中。这样就会将链接定义连接到该组件。通过使用嵌套的 <bloxportlet:parameter> 标记来指定参数及其值:

```
<%@ page contentType="text/html"%>
<%@ taglib uri="bloxtld" prefix="blox"%>
<%@ taglib uri="bloxportlettld" prefix="bloxportlet" %>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
<%
String bloxName = portletResponse.encodeNamespace("buttonContainer");
%>
<head>
   <blox:header />
</head>
<blox:container id="myButtonContainer" bloxName="<%= bloxName %>"
width="40" height="20">
<bloxportlet:actionLinkDefinition action="showData">
<bloxportlet:parameter name="a" />
<bloxportlet:parameter name="b" value="2" />
<bloxportlet:parameter name="c" />
</bloxportlet:actionLinkDefinition>
<%
BloxModel model = myButtonContainer.getBloxModel();
model.clear();
Button myButton = new Button("button1", "Show Data");
model.add(myButton);
model.changed();
%>
</blox:container>
```

然后可以从指定的操作访问 PortletLink 并在 scriptlet 中设置链接信息或参数值:

```
<%
// programmatically set the parameter values for the named Portlet
PortletLink plink = myButtonContainer.getPortletLink("showData");
plink.setParameterValue("a","1");
plink.setParameterValue("c","xyz");
myButton.setClientLink(plink.getClientLink());
%>
```

PortletLink 用来生成要在 PortletLinkDefinition 中调用的标记。某些 PortletLink 方法包含:

- getClientLink(): 返回一个 ClientLink 来表示要在 Blox UI 模型内使用的 PortletLink
- getLinkHRef(): 返回一个字符串来表示要在 HTML 锚点标记内使用的 HREF
- setParameterValue(): 设置此链接内参数的值

有关 ClientLink 的更多信息，请参阅 第 83 页的『DHTML 客户机 API 框架』。

Blox Portlet 标记库示例

本节包括一些示例，它们演示了在 Button（Blox UI 组件）和 ReportBlox 内 Blox portlet 标记的用法。每个示例使用添加操作链接或 portlet 链接的基本方法:

1. 在 Blox 或 UI 组件内添加 <bloxportlet:actionLinkDefinition> 标记以连接此链接定义，并使用 action 属性来为操作指定名称。
2. 使用嵌套的 <bloxportlet:parameter> 标记来指定参数的名称及其值。

然后可以获取指定操作的 PortletLink 并在 scriptlet 中设置链接信息或参数值。有关这些 API 的详细信息，请参阅 Javadoc 文档中的 com.alphablox.blox.portlet 包。要获取有关如何将 portlet 链接添加至 GridBlox 和 TreeFormBlox 的更多示例，请参阅 *Developer's Reference* 一书中 Blox Portlet Tag Library Reference 主题。

将链接添加至按钮

以下示例使用 3 个参数为按钮定义了一个名为"showData"的操作。PortletLink 的 ClientLink 与该按键相关联，因此，当单击该按钮时，将会对此 PortletLink 的 3 个参数中的两个参数设置值。要在另一个 portlet 中使用此信息，还需要其他代码。此示例仅演示如何设置链接。

```
<%@ page contentType="text/html"%>
<%@ taglib uri="bloxtld" prefix="blox"%>
<%@ taglib uri="bloxportlettld" prefix="bloxportlet" %>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
<%
String bloxName = portletResponse.encodeNamespace("buttonContainer");
%>
<head>
   <blox:header />
</head>
<blox:container id="myButtonContainer" bloxName="<%= bloxName %>"
width="40" height="20">
<bloxportlet:actionLinkDefinition action="showData">
<bloxportlet:parameter name="a" />
<bloxportlet:parameter name="b" value="2" />
<bloxportlet:parameter name="c" />
</bloxportlet:actionLinkDefinition>
<%
BloxModel model = myButtonContainer.getBloxModel();
model.clear();
Button myButton = new Button("button1", "Show Data");
model.add(myButton);
model.changed();
// programmatically set the parameter values for the named Portlet
PortletLink plink = myButtonContainer.getPortletLink("showData");
plink.setParameterValue("a","1");
plink.setParameterValue("c","xyz");
myButton.setClientLink(plink.getClientLink());
%>
</blox:container>
```

将链接添加至 ReportBlox 组件

以下示例为 ReportBlox 定义一个名为“selectProductCode”的操作。此链接将连接至 Product 列。当单击报告中的产品名时，参数“code”的值将设置为产品的代码。要在另一个 portlet 中使用此信息，还需要其他代码。此示例仅演示如何设置链接。

```
<%@ page contentType="text/html" %>
<%@ taglib uri="bloxtld" prefix="blox"%>
<%@ taglib uri="bloxreporttld" prefix="bloxreport"%>
<%@ taglib uri="bloxportlettld" prefix="bloxportlet" %>
<%@ taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
<head>
   <blox:header/>
<link rel="stylesheet" href="/AlphabloxServer/theme/report.css">
</head>
<%
String reportName = portletResponse.encodeNamespace("myReportBlox");
%>
<bloxreport:report id="report" bloxName="<%= reportName %>" interactive="false">
<bloxreport:cannedData />
<bloxreport:filter expression="Sales < 100" />
<bloxreport:group members="Area" />
<bloxreport:sort member="Week_Ending" />
<bloxportlet:actionLinkDefinition action="selectProductCode">
<bloxportlet:parameter name="code" />
</bloxportlet:actionLinkDefinition>
<%
PortletLink link = report.getPortletLink("selectProductCode");
link.setParameterValue("code", "<value member=\"code\"/>");
String href = link.getLinkHRef();
String productLink = "<a href=\""+ href + "\"><value/></a>";
%>
<bloxreport:text>
<bloxreport:data columnName="Product" text="<%= productLink %>" />
</bloxreport:text>
</bloxreport:report>
```

下面显示了处理此操作的方式，并创建了一个 actionPerformed() 方法以通过 Portlet API 来利用此信息:

```
<%
public void actionPerformed(ActionEvent event) throws PortletException {
String actionString = event.getActionString();
PortletRequest request = event.getRequest();
if (actionString.equals("selectProductCode")) {
String productCode = request.getParameter("productCode");
// ... use the value of the parameter accordingly ...
}
}
%>
```

第 9 章 Blox UI 标记库

DHTML 客户机 UI 模型提供了一个标记库以使您能够方便地进行常用的 UI 处理。标记包含在 bloxui.tld 库中，并且，可以在 *Developer's Reference* 一书的 Blox UI Tags Reference 一节中找到每个标记及其属性的完整列表。

DB2 Alphablox 提供了用于处理 Blox 的标记库，该标记库称为 Blox 标记库。Blox UI 标记库是对 Blox 标记库的补充。开发者应该使用 Blox 标记库来设置数据属性、执行一般的 UI 处理（如调整图表／网格方向、使菜单栏可视和调整分割窗格）以及访问一般的 DB2 Alphablox 功能（如单元格提醒和计算的成员）。如果您正在使用 DHTML 客户机并且需要对用户界面进行 Blox 库无法提供的更高级别控制，Blox UI 标记库可以向您提供所需的功能。

Blox UI 标记库类别

Blox UI 标记分为四类:

类别 **描述**

组件定制
用于在组件级别定制 UI。示例: 定制菜单和工具栏。

定制布局
提供对网格布局的控制，例如，添加空行或空列，或者以蝶形布局生成网格。

分析 用来将分析功能合并到应用程序中。

实用程序
用于简化操作处理的便捷标记。开发者可以使用实用程序标记来拦截用户选择或者在网格更改时执行操作。

Blox UI 标记示例

本节提供诸多 Blox UI 标记的几个示例，目的是让您感受一下这些简单易用的标记的强大功能。

Blox UI 组件定制

以下示例显示了一个组件定制标记。这些组件定制标记可以用来对用户界面添加和除去（或者启用和禁用）菜单。

例如，您可以使用下列 Blox 和 Blox UI 标记来禁用“工具”菜单并除去“书签”菜单:

```
<blox:grid id="testGridBlox"
width="600"
height="700"
bandingEnabled="true"
rowIndentation="None"
commentsEnabled="false">
<blox:data bloxRef="dataBlox" />
<bloxui:menu name="toolsMenu" disabled="true" />
<bloxui:menu name="bookmarkMenu" visible="false" />
</blox:grid>
```

定制布局标记

下一个示例显示使用定制布局标记来将网格的显示更改为以蝶形布局显示，这将把行头移至网格中间，如下图所示:

```
<blox:present id="bfpresent"
visible="true"
width="600"
height="400"
chartAvailable="false">
<blox:grid bandingEnabled="true" />
<blox:data bloxRef="bfdata" />
  <bloxui:butterflyLayout
scope="{ Scenario:Budget }"
showOnLayoutMenu="true"
addSeparatorColumns="false" />
</blox:present>
```

通过使用此标记的属性，开发者可以指定行头的位置以及是否应该在头与数据之间引入分隔列。

分析标记

开发者还可以使用分析标记来将分析功能直接合并到应用程序中。例如，组装人员可能想将“bottom N”计算合并到他们的应用程序中:

可以使用下列 PresentBlox 标记和 Blox UI bottomN 标记来实现此视图:

```
<blox:present id="tbnpresent"
width="600"
height="500"
chartAvailable="false">
<blox:grid bandingEnabled="true" />
<blox:data bloxRef="tbndata" />
<bloxui:bottomN
prompt="true"
showRank="true"
number="7"/>
</blox:present>
```

Blox UI 分析标记还包括一个使开发者能够将计算合并到应用程序中的一般标记。

实用程序标记

最后，Blox UI 标记库提供了标记来大大简化用户输入处理。以下代码样本使用实用程序标记来拦截用户对“旋转”菜单项的单击以显示对话框。

```
<blox:grid id="testActionFilter"
width="80%"
height="500"
bandingEnabled="true">
<blox:toolbar visible="true" />
<blox:data bloxRef="dataBlox" />
<bloxui:actionFilter
className="<%= MyActionFilter.class.getName() %>"
componentName="dataPivot" />
</blox:grid>
<%!
  public static class MyActionFilter implements IActionFilter
{
    public void actionFilter( DataViewBlox blox,
                        Component component ) throws Exception
```

```
{
MessageBox.message( component, "Action Filter",
"Item clicked!" );
}
}
%>
```

在此示例中，首先在 JSP 文件中定义 `actionFilter` 类，然后将其与网格和“旋转”菜单项相关联。在 *Developer's Reference* 一书中，Blox UI Tags Reference 的 Utility Tags 一节对事件过滤器作了进一步的讨论。

更多示例

请查看“Blox 样本程序”以获取与这些以及其他 Blox UI 标记库标记相关的示例（提供了源代码）。

第 10 章 DHTML 客户机 UI 可扩展性

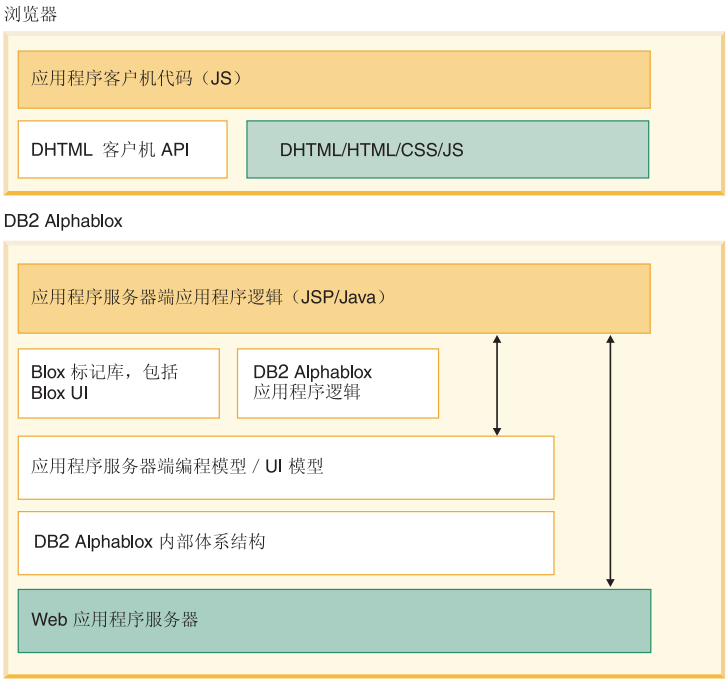
这个高级主题面向能够轻松自如地使用服务器端 Java API 的开发者。这里深入描述的 Blox UI 模型提供了丰富并且功能强大的 API，该 API 可以完成使用简单 Blox UI 标记无法完成的用户界面定制工作。下一个主题（第 83 页的第 11 章，『DHTML 客户机 API』）讨论 DHTML 客户机 API，它对于应用程序开发者而言是客户机端编程模型与服务器端编程模型之间的纽带。

Blox UI 模型

DHTML 用户界面模型是一个程序 API，它允许应用程序开发者检查和控制最终用户的用户界面。此 API 提供了对用户界面所有各方面（包括显示的内容以及用户输入处理方式）的直接控制。UI 模型 API 也提供了一定程度的用户界面控制，此控制类似于开发者使用数据 API 对元数据和结果集数据进行的控制。数据 API 和用户界面 API 一起为开发者提供了对应用程序所有各方面的总体控制和定制。

如下图所示，DB2 Alphablox 框架包括:

- DB2 Alphablox 内部体系结构
- DB2 Alphablox 服务器端编程模型，它包括 UI 模型
- DB2 Alphablox 应用程序逻辑
- Blox 标记库，包括 Blox UI 标记库
- DHTML 客户机 API



Blox UI 模型 API 是 DB2 Alphablox 服务器端编程模型的关键部分，所有表示 Blox（包括 PresentBlox、GridBlox、ChartBlox、DataLayoutBlox 和 PageBlox）都支持此 API。服务器端 `getBloxModel()` 方法允许应用程序开发者访问特定 Blox 实例的模型。

当我们提到 Blox UI 模型时，我们实际上指的是三个不同的用户界面概念：组件、控制器和事件。这些组件概述如下：

概念 **描述**

组件 构成用户界面的各个控件和容器，如按钮、列表框、编辑字段、网格和图表。组件存在于容器的层次结构中，以便为用户界面提供结构。产生的模型就是向用户显示的用户界面的逻辑表示。

控制器 用来处理来自组件的事件，从而将一般组件行为转换成应用程序定义的行为。例如，作为对用户选取复选框的响应，可以显示一个图表。在这种情况下，控制器将复选框选取操作解释成图表显示信号。所有应用程序逻辑都应该在与模型组件（即容器）相关的一个或多个控制器中。

事件 将用户界面、底层应用程序逻辑和模型本身的状态更改传达给模型的组件和控制器。每个组件和控制器都具有一组它能识别和理解的预定义事件。识别的事件通常会导致修改存储在本地的组件状态。应用程序代码应该根据用户操作来使用事件触发应用程序逻辑。例如，当浏览器中的复选框状态更改时，将生成 `ClickEvent` 并将它发送至服务器。当应用程序的定制控制器接收到 `ClickEvent` 时，它可以执行相关联的应用程序行为，如显示图表。

对于 UI 模型，您应该记住下列要点：

- 模型是客户机上的用户界面对象状态的服务器端表示。这允许服务器端 Java 代码设置组件、处理组件之间的交互以及处理用户输入，而不必实际地编写任何客户机端代码。模型本身并没有输入按钮；但它允许服务器端代码控制浏览器中向用户显示的这种按钮的状态。例如，这允许服务器端代码确定用户选择了哪些网格单元格。
- 直接对 UI 模型进行编程是可选的。当应用程序开发者想要添加新功能或修改现有功能时，他们只需要与模型进行交互。

Blox UI 模型的用途

每个表示 Blox 都已提供了丰富的功能和属性，这些功能和属性可以更改 Blox 的外观和行为。这些功能和属性包括：

1. **Blox 标记属性**（`bandingEnabled` 和 `chartFirst` 等）
2. **Blox 属性**（在 JSP scriptlet 中，通过方法调用来处理这些属性）
3. **Blox UI 修饰符标记**，如蝶形布局和压缩头等

如果内置行为正是应用程序开发者所期望的，则只需要对 Blox 标记添加属性或者调用服务器端 Java 方法。

在某些情况下，应用程序开发者需要使用未作为内置属性提供的新功能或者希望对内置功能的实现进行调整。在引入 UI 模型之前，唯一的求助方法是请求更改并等待它在新产品发行版中出现。

通过使用 UI 模型，应用程序开发者可以更改内置功能的行为以及对 UI 添加新功能。下面是这些定制工作的一些示例：

- 添加新的工具栏以提供定制计算操作

- 更改网格的外观
- 将复选框或其他控件添加到网格单元格中并提供逻辑以处理用户的选择
- 根据用户单击的用户界面组件来修改右键单击菜单

UI 模型并未代替易于实现的内置产品功能，但它为应用程序开发者提供了根据需要进行几乎无限定制的能力。

Blox UI 组件概述

模型中的每个组件都与用户界面上的特定控件相对应。这些控件包括按钮、网格、树、复选框、列表框、图表、菜单和工具栏等。更改 UI 模型中的组件状态将影响用户界面中的控件状态。同样，当用户与用户界面中的控件进行交互时，将更新 UI 模型以反映控件状态。

因为模型保持所有组件的状态，所以，即使用户刷新页面，基于服务器的模型都将保持组件的状态，并且将使用刷新页面时的状态。基于服务器的应用程序代码在任何时候都可以检查用户界面中使用的任何组件的状态，并且不需要单独地存储或管理状态信息。例如，如果将 Checkbox 组件添加到 UI 模型中，则 `Checkbox.isChecked()` 方法将提供有关选取状态的最新信息。

在 UI 模型中，组件按层次结构排列，层次结构提供了格式化控制以及对基本组件集进行集中管理的方法。这个层次结构是使用一个或多个 ComponentContainer 实现的，而 ComponentContainer 又包含 Component 以及其他 ComponentContainer。

对于一个简单的对话框，得到的层次结构可能类似于以下结构:

对话框 组件容器 组件容器 检查框 单选按钮 单选按钮 组件容器 按钮 按钮

在 Blox 的生存期内，任何时候都可以更改、修改、添加或删除 UI 模型组件。这允许用户界面在用户与界面进行交互（选择选项和功能部件）时进行更改。在将初始页面传递给浏览器之后，在更改组件时，开发者必须调用 `changed()` 方法，如下所示:

- 如果直接或间接地修改了组件（即，更改了它的样式），则应该对该组件本身调用 `changed()`
- 如果添加或删除了组件，则应该对父容器调用 `changed()`。

在将初始页面传递给用户之前，对组件调用 `changed()` 方法是没有任何效果的（无论是正面效果还是负面效果）。

组件

模型中的每个组件都与用户界面上的特定控件相对应（隐藏的组件除外）。这些控件包括按钮、网格、树、复选框、列表框、图表、菜单和工具栏等。在 UI 模型中，`Component` 是所有可视组件和容器的基类。因此，每个可视组件都是从 Component 派生的。Component 类提供可视组件参与 UI 模型框架所需的一组基本属性和行为。同样，所有不可视模型对象（如样式、布局和图表轴定义）都并非源于 `Component` 基类。

组件名称

可以对所有组件指定名称。在组件的生存期内，名称是可以更改的，但可能性最大的情况是，名称被设置一次，然后在组件的生存期内保持不变。名称有两项用途:

1. 名称用来标识组件及其在模型中的角色。例如，每个菜单项都具有用来标识其特定功能的名称。如果组件没有名称，就很难标识组件的用途 – 在模型内移动组件时尤其如此。具有名称的组件可以在模型内四处移动并且仍能够正常地工作。
2. 组件的名称用作它的“操作代码”。生成操作事件（例如，`ClickEvent`）的组件使用组件的名称（如果有的话）来将操作映射到方法。稍后，我们将在『控制器』一节中全面地描述此用途。

处理不唯一的组件名称

名称不必是唯一的，在缺省 Blox 模型中，一些名称被不同的组件所共享。当多个组件映射到同一个操作时，这非常方便，并且这还允许在不同模型之间自由地移动组件。

由于名称不保证是唯一的，所以按名称搜索组件的代码应该准备好处理多个结果。要降低基于名称的搜索返回多个结果的机会，应该从组件层次结构中尽可能深的位置启动搜索。例如，如果正在查找名为“sort”的工具栏按钮，则应该在每个工具栏中启动搜索，而不是在模型顶部启动搜索。

以下示例显示如何查找具有同一名称的所有组件以便对它们执行一些操作（在本实例中，要执行的操作是将它们隐藏起来）。

```
ArrayList components =
myBlox.getBloxModel().searchForAllComponents("componentName");

for (int i=0; i < components.size(); i++) {
  Component component = (Component)components.get(i);
  component.setVisible(false);
}
```

最后，没有明确的“规则”要求名称一定不是唯一的。定制模型代码可以很方便地遵守它所添加的组件的唯一命名约定，而不必担心组件搜索操作返回多个结果。

内置名称

Blox 模型添加的所有组件都具有一组标准名称。您应该避免对不相关的函数使用相同的名称。*Blox API Javadoc* 文档中的 `com.Alphablox.blox.uimodel.ModelConstants` 类文件以及 *Developer's Reference* 一书的 Blox UI Tags Reference 一节定义了所有主要 Blox 组件的标准名称。在代码中，始终应该使用已定义的常量，而不是使用硬编码的字符串。

组件标题

标题用来向用户描述组件。例如，添加到 CheckBox 的标题将用来告诉用户他们选取这个框的原因。每个组件使用标题的方式都略有不同，但是目的相同。下面提供了每个组件及其标题使用方式的列表。

组件类型
: **“title”的使用方式**

Static 显示的值

ComponentContainer
: 顶级容器的标题，否则被忽略

Checkbox
: 显示在 CheckBox 后面

RadioButton
: 显示在 RadioButton 后面

Edit 被忽略

GroupBox
GroupBox 的标题

ListBox 和 DropDownList
被忽略

Image 和 StaticImage
被忽略

Toolbar 和 Menu bar
在菜单中用来引用 Toolbar，否则被忽略

Menu 和 MenuItem
菜单标注

Button
按钮标注

Spacer
被忽略

容器

每个组件都必须位于容器中才能显示。可以按层次结构安排容器以便对一组组件进行封装以及进行布局控制。ComponentContainer 提供诸如在容器内搜索组件以及在容器的层次结构内任何位置搜索组件之类的服务。

ComponentContainer 是 Component 的后代，后者允许将容器添加到其他容器中以及共享基本 Component 功能。例如，容器可以具有名称、UID、边框和背景色。

容器内的组件是按照组件被添加到容器的顺序显示的。容器的布局定义了应该如何解释此顺序。

布局

ComponentContainer 布局限于只能指定容器中的组件显示方向。对容器指定 `VerticalLayout` 将使组件垂直地堆叠。对容器指定 `HorizontalLayout` 将使组件从左到右地显示。

```
// Show the components in the container vertically stacked
ComponentContainer.setLayout(new VerticalLayout());
// Show the components in the container left to right
ComponentContainer.setLayout(new HorizontalLayout());
```

复合组件

模型提供了许多核心用户界面组件。但是，在许多情况下，您可能希望创建更高层的组件，那些组件包含协调工作的特定数目的核心组件。于是，可以将这些“复合组件”视为任何其他组件，并且可以根据需要将它们添加到 UI 中。

创建复合组件时，您只需要扩展 `ComponentContainer` 类并添加所需的用户界面组件。

例如:

```
Class MyComponent extends ComponentContainer {
    public MyComponent() {
add(new Static("label:"));
    add(new ListBox());
}
// Deal with events and add custom behaviors
}
```

然后，可以象添加任何核心组件类那样方便地将以上 MyComponent 类添加到任何 ComponentContainer 中:

```
myContainer.add(new MyComponent());
```

通过创建复合组件来创建具有内置行为的可重用定制组件，那些定制组件的使用与核心组件一样方便。

使用 ContainerBlox

为了在页面上显示任何 UI 模型组件，必须将它们放在 Blox 框架内。ContainerBlox 实际上是一个空的 Blox 框架，它不具有预定义的 DB2 Alphablox 应用程序逻辑。它在页面上为开发者提供了一个区域，以便他们使用 UI 模型的用户界面组件来创建定制用户界面。由于 ContainerBlox 不具有预定义的行为，所以开发者需要手工添加所有必需的组件，包括菜单、工具栏和网格等。

当应用程序需要任何表示 Blox 都未提供的定制用户界面时，应用程序开发者将使用 ContainerBlox。例如，使用 ContainerBlox 来在页面上放置 UI 模型树组件以帮助用户进行导航。在这个示例中，由于树操作百分之百由应用程序定义，所以不需要继承任何现有的 Blox 行为。

ContainerBlox 可以象任何其他 Blox 一样与 UI 模型配合使用，例如:

```
<blox:container id="myComponent" >
<%
BloxModel model = myComponent.getBloxModel();
// Add user interface components and handlers
// Keep in mind that the model is empty
%>
</blox:container>
```

此外，可以创建 ContainerBlox 的子类以便根据模型来创建独立的定制 Blox 组件。

```
class MyComponent extends ContainerBlox
{
   public MyComponent( )
{
          BloxModel model = myComponent.getBloxModel();
// Add user interface components and handlers
}
}
```

然后，可以在 <jsp:useBean> 标记中使用上面这个类，例如:

```
<jsp:useBean id="myBlox" class="MyComponent" scope="session" />
```

控制器

控制器提供应用程序逻辑以定义一个或多个组件的行为。UI 模型的基本控制器类还提供了许多服务（这些服务使您能够更方便地处理事件），并且提供了方便您覆盖标准控制器行为的框架。任何组件或组件派生的类都可以具有相关的控制器，但是，并非所有组件都必须具有控制器，在大多数情况下，这是正常的。

当组件接收到事件时，该事件将被分派到与该组件相关的控制器。如果没有控制器可用，或者相关的控制器指示应该继续传递该事件（通过从事件处理程序中返回 `false` 来指示这一点），则将把该事件发送至该组件的父代。此过程将不断重复，直到该事件被处理或者组件没有父代为止，这是因为，对于顶级容器，事件只会被忽略。

由于控制器通常提供了应用程序逻辑来将许多组件的状态转换为单个操作，所以，那些控制器很有可能与容器相关，而不是与各个组件相关。有关此问题的一个基本示例是对话框，在对话框中，许多组件由与该对话框相关的控制器控制。

尽管容器更有可能具有相关的控制器，但是，没有任何情况导致组件不能具有专用的控制器。通常，组件在下列情况下具有自己的控制器:

- 组件是对执行特定任务的用户界面的添加项。应用程序开发者添加的大部分菜单项和工具栏按钮都属于此类，并且，对已添加的组件添加控制器也是有意义的。
- 组件是容器的具有控制器的部件，但是，组件可以执行一些能够影响其状态的特殊处理。例如，编辑控件可以具有专门用于验证用户输入的控制器。

Controller 基类

所有控制器类都必须继承 `Controller` 类。这个基类提供了许多服务，包括:

- 将接收到的所有事件转换为方法调用。控制器接收到的每个事件将会导致调用格式为 `public boolean handle`*`EventType`*`(`*`EventType`* `event) throws Exception` 的方法。应该将 `EventType` 替换为实际的事件类，如 `SelectionChangedEvent`。如果该方法不存在，控制器就将忽略该事件。
- 根据用户单击的组件的名称来将 `ClickEvent`（这是主要的用户操作事件）转换为方法调用。例如，名为“myButton”的按钮组件的 `ClickEvent` 将导致调用方法 `public void actionMyButton(ModelEvent event) throws Exception` `public void actionMyButton(ModelEvent event) throws Exception`。如果该方法不存在，控制器就将忽略该事件并将其继续传递给下一个感兴趣的控制器（如果有的话）。
- 当与该控制器相关联的组件创建的对话框被关闭时，调用 `closedDialogName()` 方法。
- 提供一个基础结构，这个基础结构允许定制代码添加事件处理程序以覆盖控制器的内置事件行为。

“隐式”控制器

许多核心组件具有不能被覆盖的隐式控制器。这些隐式控制器处理内部状态更改，从而在相关的控制器检查组件的状态之前组件能够反映正确的状态。例如，当复选框被选取时，如果 CheckBox 组件在任何控制器接收到 `ClickEvent` 之前接收到 `ClickEvent`，则 CheckBox 组件将立即选取自己。

当控制器接收到事件时，控制器可以安全地从组件获取状态信息，并且能够肯定该组件在服务器上的模型准确地反映了客户机上的控件状态。

Blox UI 模型事件

事件用来在浏览器与 UI 模型之间传达组件状态更改和操作。在 UI 模型中，还使用事件来将模型更改和属性更改信息通知控制器。

有关 UI 模型事件的要点如下所示:

- 大部分事件传达详细的用户界面操作。例如，按钮单击、菜单单击和滚动等。
- 应用程序开发者编写的代码可以拦截事件。例如，代码可以拦截用户对“下寻”菜单项的单击。
- UI 模型中的事件在涉及用户界面操作方面与 JavaScript 事件类似
- 事件被分派给生成该事件的组件，然后被分派给该组件的父代
- 事件还用来在模型内部的控制器与组件之间传递信息
- 这些内部事件允许代码拦截模型内部的关键操作以及用户界面操作。例如，当网格创建单元格时，它将生成一个事件以允许代码定制该单元格。

事件是按特定顺序分派的，该顺序使所有相关组件和控制器都能够参与事件处理，如下所示:

1. 模型对象 – 生成事件（如编辑字段内容更改）的特定模型组件。
2. 模型对象控制器 – 如果该组件具有控制器，则该控制器将接收事件。
3. 模型对象的父代 – 重复以上步骤 1 和 2，直到到达顶级容器为止。
4. 模型控制器 – 模型的顶级控制器将是事件的最后一站。如果该控制器未处理该事件，则将废弃该事件。

在所有情况下，事件处理程序都可以执行下列其中一项操作:

- 忽略该事件并且不执行任何将会导致继续分派事件的操作。
- 吸收该事件并且执行一些操作（可选），这些操作可以包括生成其他事件。不对该事件进行进一步的分派。
- 修改该事件并允许继续分派它。
- 对该事件作出反应，但允许继续分派它。
- 使用 `getComponent()` 来访问生成该事件的组件
- 访问特定于该事件的任何其他属性

下列示例描述了如何拦截按钮组件发出的事件。对于每个示例，组件是名为 `MyButton` 的按钮，并且它被添加到名为 `blox` 的 Blox 中。当用户按下按钮时，所有按钮都将生成 `ClickEvent`。

将专用控制器添加至组件

在本示例中，我们对按钮本身添加一个控制器，该控制器将处理按钮单击。这样，在将单个的组件添加至模型时就可以很方便地添加行为，但会导致难以在多个组件之间协调行为。当组件尚未包含控制器，或者当您希望替换预先存在的控制器时，您将使用这种方法来处理事件。

```
<blox:grid ... >
<%
BloxModel model = blox.getBloxModel();
Button button = new Button("MyButton");
button.setController(new Controller() {
```

```
    public boolean actionMyButton(ModelEvent event) throws Exception
{
// Do something
}
});
  model.add(button);
%>
</blox:grid>
```

对预先存在的控制器添加侦听器

此示例对 Blox 模型控制器添加按钮处理程序。这里，我们对模型层次结构中具有较高级别的控制器添加侦听器。该事件将被发送至生成该事件的组件（即 Button 组件），然后在组件层次结构中向上传递。

```
<blox:grid .... >
<%
  BloxModel model = blox.getBloxModel();
model.add( new Button("MyButton"));
  model.getController().addEventHandler(new IEventHandler() {
    public boolean handleClickEvent(ClickEvent event)
    throws Exception {
if ("MyButton".equals( event.getComponent().getName())){
// Do something
return true;
}
return false;
}
});
%>
</blox:grid>
```

注意，上述所有示例都是在 Blox 标记内添加事件处理程序。这一点很重要，其原因在于您不想在页面每次刷新时都添加处理程序。仅当最初创建 Blox 时才执行 Blox 标记内的所有代码，而不会在每次刷新页面时执行那些代码。在将 <jsp:useBean> 与会话作用域配合使用时，也有类似的约定。

模型分派器

组件一旦与 Blox 模型相关，就可以使用该组件来获取模型分派器。分派器是模型中的顶级容器提供的服务点，它提供许多与模型相关的服务。分派器提供了下列服务（请参阅 BloxModel 和 IModelDispatcher 接口）：

- 显示对话框 – 导致使用 showDialog() 来在客户机上显示对话框
- 关闭对话框 – 导致使用 closeDialog() 来在客户机上关闭对话框
- 使用 getTopLevelContainer() 来获取对顶级容器的引用
- 分派事件 – 使用 dispatchEvent() 来在模型内分派事件
- 显示浏览器窗口 – 导致浏览器使用 showBrowserWindow() 来显示新窗口并访问所提供的 URL
- 使用 sendClientCommand() 来将 JavaScript 命令发送给客户机
- 显示右键单击菜单 – 导致浏览器立即使用 setAttachedRightClickMenu() 来在指定的位置显示右键单击菜单
- 控制忙状态 – 允许代码使浏览器 UI 处于忙状态，直到使用 setBusy() 解除此状态为止

分派器的最常见用法是显示对话框，如下所示:

```
component.getDispatcher().showDialog(myDialog);
```

与模型不相关的组件没有模型分派器。

对话框

对话框用来收集用户输入，以便设置选项或者弄清用户意向。UI 模型使应用程序开发者能够快速地构造对话框并向用户显示该对话框。Dialog 是一个容器，它通过添加两项特殊功能来扩展基本的 ComponentContainer 模型对象:

1. 对话框位于浏览器上的一个单独的窗口中，该窗口可调整大小并且可移动。
2. （可选）对话框可以让用户界面其余部分停止接受输入，直到该对话框关闭为止。

除此之外，Dialog 的工作方式与其他 ComponentContainer 非常相似。大部分（如果不是全部的话）Dialog 还将需要一个 Controller 来解释用户的选择并执行操作。

为了使对话框能够引起用户注意，可以使用 Dialog.setModal(boolean) 方法来设置对话框的模态属性。模态对话框禁止与用户界面的其他部分进行交互，直到模态对话框关闭为止。非模态对话框不禁止用户界面交互，它最适合用于具有“应用”功能的对话框。可以同时显示多个对话框并让最后一个显示的模态对话框控制用户界面。

如果用户刷新浏览器页面，则对话框将重新显示。

创建简单的对话框

通过使用 Blox UI 模型，可以创建对话框以捕获用户输入。要创建对话框，请执行下列基本步骤:

1. 根据 UI 模型资源文件创建对话框。
2. 创建控制器（它扩展 DialogController）以处理用户与该对话框的所有交互。在大多数情况下，控制器只对“确定”、“取消”和“应用”这些用户操作起作用。
3. 使该控制器与 Dialog 对象相关。
4. 指示模型分派器显示该对话框。

在以下示例中，JSP 页面添加一个标注为“My Menu Choice”的定制菜单项，此菜单项将被添加到 PresentBlox 菜单栏的“数据”下面。当用户单击“My Menu Choice”时，将显示 XML 资源文件中定义的对话框。在这个简单的示例中，对话框询问用户一个问题并提供了“确定”或“取消”响应。下面显示了这两个文件的代码，JSP 页面显示 PresentBlox 和定制菜单项，XML 资源文件用来定义对话框窗口。JSP 页面期望在应用程序的根目录中找到 XML 资源文件。

JSP 页面（customDialog.jsp）

以下 JSP 文件将在页面上显示 PresentBlox 并在“数据”菜单底部提供定制的“My Menu Choice”菜单选项。如果您想了解所发生的情况，请阅读 JSP 文件中的注释。

```
<%@ page import="com.alphablox.blox.*,
  com.alphablox.blox.uimodel.*,
  com.alphablox.blox.uimodel.core.*,
  com.alphablox.blox.uimodel.core.event.*,
  com.alphablox.blox.uimodel.core.Component.*,
  com.alphablox.blox.uimodel.core.grid.*,
```

```
  com.alphablox.blox.uimodel.tags.IActionFilter,
  com.alphablox.blox.uimodel.tags.internal.ActionFilterAdapter,
  com.alphablox.blox.data.*,
com.alphablox.blox.data.mdb.*" %>
<%@ page import="java.io.*,
  java.io.File.*,
java.util.*" %>
<%@ taglib uri="bloxtld" prefix="blox"%>
<%@ taglib uri="bloxuitld" prefix="bloxui"%>
<%!
  // class needs to be static in order to be used by the
  // bloxui:actionFilter tag
  public static class MyActionFilter implements IActionFilter {
    String dialogPath;
    public MyActionFilter(PageContext pageContext) {
      File ctxPath =
new File(pageContext.getServletContext().getRealPath(""));
      dialogPath = ctxPath.getAbsolutePath() + File.separator +
"MyDialog.xml";
}
    // handle the action for the MyMenuChoice component
    public void actionFilter(DataViewBlox blox, Component component)
    throws Exception {
System.out.println("actionMyMenuChoice() was called");
      try {
        Dialog dialog = Dialog.createFromResource(dialogPath);
        DialogController dialogController = new MyDialogController(dialog);
        // Attach the controller to the dialog
        dialog.setController(dialogController);
        // Get component from the event so we can get model dispatcher
        // The dispatcher is used to send the dialog to the client
        component.getDispatcher().showDialog(dialog);
}
      catch(Exception e) {
System.out.println("actionMyMenuChoice() exception" + e.getMessage());
        throw e;
}
}
}
  public static class MyDialogController extends DialogController {
    public MyDialogController(Dialog dialog) {
      super(dialog); // must be the first thing in this constructor
System.out.println("MyDialogController () was called");
}
    public void actionOk() {
      // Take some action
System.out.println("actionOk() was called");
      // Invoke default OK handler after taking the action
      super.actionOk();
}
}
%>
<blox:data id="analyticsDataBlox"
dataSourceName="QCC-Essbase"
query="!">
   </blox:data>
<blox:present id="analyticsBlox"
visible="false"
width="95%"
height="45%"
splitPane="false"
>
<blox:data bloxRef="analyticsDataBlox"/>
<bloxui:menu name="dataMenu">
<bloxui:menuItem separator="true" />
<bloxui:menuItem name="MyMenuChoice"
title="My Menu Choice" />
```

```
  </bloxui:menu>
<bloxui:actionFilter
filter="<%= new MyActionFilter(pageContext) %>"
componentName="MyMenuChoice" />
</blox:present>
<html>
<head>
   <blox:header/>
</head>
<body>
<blox:display bloxRef="analyticsBlox" />
 </body>
</html>
```

XML 资源文件（MyDialog.xml）

在以上示例中，我们使用资源文件创建了对话框。另外，我们还可以通过创建和添加构成对话框的各个组件来创建对话框。资源文件由对话框中组件的可进行语言本地化的 XML 表示组成，这是一种简单的对话框、菜单栏和工具栏创建方法。

本示例的对话框资源 MyDialog 类似于:

```
<?xml version="1.0" ?>
<Dialog name="MyDialog"
title="My Dialog"
cache="false"
modal="true"
height="150"
width="500"
layout="vertical">
<Static title="Do you really want to do this?" />
<ComponentContainer layout="horizontal" alignment="center">
<Button name="ok" title="OK" />
<Button name="cancel" title="Cancel" />
</ComponentContainer>
<Spacer />
</Dialog>
```

可以将任何核心模型组件作为容器子代添加到资源文件中。要了解有关资源文件的更多信息，请参阅 *Developer's Reference* 一书的 XML Resource Files Reference 一节。

MessageBox

MessageBox 是具有简单 API 的模态对话框，它使开发者能够快速方便地向用户显示基于文本的消息。MessageBox 能够以确定、是／否、是／否／取消和确定／取消响应形式收集用户的简单输入。MessageBox 在本质上始终是模态的，所以，用户必须进行响应才能继续与其他用户界面部分进行交互。

如果应用程序逻辑需要将某些情况通知用户，则可以向用户发送 MessageBox 并可以忽略用户的响应。

```
MessageBox.message(myBlox.getBloxModel().getModelDispatcher(),
"Attention User",
"Some situation has occurred you should know about");
```

当应用程序逻辑对用户的响应感兴趣时，它可以使用 MessageBox 类提供的回调机制来了解用户的意愿。在这种情况下，MessageBox 对 IMessageCallback 接口调用一个方法来将用户响应传递回给应用程序代码。任何类都可以实现此接口以便当用户对 MessageBox 作出响应时接收通知。

下面是一个显示 MessageBox 并提供响应处理程序的类示例:

```
class MyClass implements IMessageCallback
{
  public void ask(IModelDispatcher dispatcher) {
MessageBox.message(dispatcher, "MessageBox Title",
"MessageBox message",
                     MessageBox.MESSAGE_OKCANCEL,this);
}
    public boolean action(MessageBox messageBox,String action){
        // handle the user response
}
}
// To invoke the above MessageBox
MyClass mine = new MyClass();
mine.ask(myBlox.getBloxModel().getModelDispatcher());
```

DHTML 客户机应用程序逻辑和流

UI 模型控制着跨多层分布的用户界面 – DB2 Alphablox 位于一层，浏览器位于另一层。这与 HTML 的分层性质类似，但有一些关键的差别。

UI 模型在不进行页面刷新的情况下更新 DHTML 用户界面。当用户与 UI 进行交互时，将对页面进行更改，但不刷新整个页面或框架集。

UI 模型将用户界面的表示放在服务器上，服务器负责保持 UI 状态并为 UI 提供基于服务器的程序化接口。这表示两个层需要互相同步，在服务器上进行的更改需要在客户机上有所反映，反之亦然。

让用户界面跨多个层分布并基于 HTTP 协议将影响服务器端代码的编写方式以及它处理用户操作的方式。服务器端代码不能等待用户响应，这是因为那些响应是在不同的线程中发生的，并且每个线程都是有限的服务器资源。无论是通过消息循环、回调还是处理程序来实现，大部分工作与其他用户界面的操作方式并没有多大区别。在 UI 模型中，使用控制器来处理用户操作。

代码通常具有以下结构:

线程 1（假定用户单击了用户界面上的按钮）

1. 拦截 ClickEvent
2. 创建 Dialog 对象
3. 使用所需的组件填充对话框
4. 使一个控制器与该对话框相关以处理对话框事件
5. 使用分派器来显示该对话框（此时，该对话框尚未真正显示，而是进行排队以便尽快地显示）
6. 从 ClickEvent 处理程序返回

经过一段时间 ...

线程 2（假定用户在线程 1 创建的对话框上单击了“确定”）

1. 在对话框控制器中拦截“确定”按钮的 ClickEvent
2. 对该对话框中的组件状态执行操作
3. 导致该对话框关闭（同样，该对话框将进行排队以便尽快地关闭）
4. 从 ClickEvent 返回

本示例使用显示对话框并从对话框收集输入的实例，但是，一般结构对于所有模型组件来说都是相同的。用户操作的处理与模型的创建总是在不同线程中进行的。

DHTML 客户机是基于主题的

UI 模型由主题驱动。DHTML 客户机使用的所有 CSS 类名都将被公布，并且想要创建自己的定制主题的应用程序开发者可以覆盖这些类名。请记住，由于 UI 模型的图表对象是图像，所以它不使用主题的 CSS 设置（请参阅第 73 页的『制图』）。

由于 CSS 文件是有组织的，所以您可以很容易地找到和更改公共的可视属性（如颜色、字体、字体大小和背景图像）。所有这些属性都位于每个主题目录的 *themeName*_dhtml.css 文件（例如，coleman_dhtml.css）中。要获取 CSS 类名及其函数的列表，请参阅 CSS 主题文档（第 142 页的『CSS 主题』）。

主题的布局将刚好在每个 Blox 显示前应用于该 Blox。如果您编写了大幅修改 UI 模型布局的定制代码，则应该对该 Blox 关闭主题布局应用。如果未关闭主题布局应用，则很可能会撤销您所作的更改而采用主题的缺省布局。

setApplyThemeLayout(boolean) 方法控制着主题布局对 Blox 模型的应用。将此方法设置为 false 将使服务器停止应用布局。

样式

所有 UI 模型组件都允许开发者应用样式以覆盖由控制主题提供的缺省样式。可以应用于组件的样式包括前景色和背景色、字体、边框以及其他文本属性（如下划线、粗体和斜体）。

UI 模型样式与控制主题的 CSS 类中定义的样式一起发挥作用。在对组件应用样式时，将仅对该组件应用样式中明确设置的属性。未设置的属性将继续继承由主题提供的值。这使开发者在设置组件前景或背景时不必关心所有其他属性（如字体和边框）。

由于图表基于不受主题的 CSS 样式影响的图像文件，所以，图表组件使用略微不同的样式机制。要了解有关特定于图表的样式的描述，请参阅『图表』一节。

要对 UI 模型组件应用样式，请创建 Style 对象并在组件中并设置它，如下所示:

```
// Make the text in a component bold
Style myStyle = new Style();
myStyle.setBold(true);
myComponent.setStyle(myStyle);
```

另外，可以使用类似于 CSS 的缩写表示法来创建 Style 对象。请注意，缩写表示法是一组有限的 CSS，它仅支持 Blox 模型 Style 对象明确支持的那些 CSS 样式。

```
// Make the text in a component bold
Style myStyle = new Style("font-weight:bold;");
myComponent.setStyle(myStyle);
```

除了 Style 对象以外，开发者也可以指定特定的 CSS 类以便对每个 UI 模型组件使用该 CSS 类。当要应用不受 Blox 模型 Style 对象支持的 CSS 样式时，应该使用这个 CSS 类。指定的类可以位于 CSS 文件中，也可以使用 <Style> 元素来在 HTML 页面中指定该类。

```
// Set the CSS class to be used by tha component

myComponent.setThemeClass("myCssClass");
```

使用此方法时，由于还会覆盖为某些组件指定的缺省 DHTML 主题类，所以您务必非常小心。而且，一些模型组件会根据组件状态修改主题类名。例如，将 Disabled 或 Enabled 追加到 CSS 类名后面。

设置多个主题类

如果要扩展对组件使用的 CSS 类，则可以添加由空格分隔的多个主题类名。例如，要将 CSS 类添加到已定义了类的组件中，请执行下列操作:

```
button.setThemeClass("myThemeClass "+ button.getThemeClass());
```

制图

制图是 UI 模型的一个特别有趣的功能。组装人员可以使用图表组件来创建图表或修改现有图表。本节的目的是阐述图表 UI 模型的一般结构并提供几个常用任务示例，以使组装人员可以开始了解制图可扩展性。

与制图有关的关键术语

DataSeries

数据系列是数据值的列表，其中，每个数据值通常都表示一个成员。例如，如果有一个 Sales 对 Region（East、West、North 和 South）的网格，则数据系列包含 4 个数据值（East、West、North 和 South 各 1 个数据值），并且该数据系列将名为“Sales”。于是，当在图表上绘制数据时，将绘制一条带有 4 个点的线。不同的图表接受不同类型的数据系列对象。SingleValueDataSeries 是一种由条形图、折线图、面积图和饼图使用的数据系列，在此数据系列中，图表上的每个点都仅仅是一个值。BarDataSeries 和 LineDataSeries 两者都扩展 SingleValueDataSeries。另外，还有多值数据系列（例如，ScatterDataSeries 和 BubbleDataSeries）。例如，ScatterDataSeries 的每个数据点都有 X 和 Y 值。某些图表类型只支持 1 个数据系列。目前，PieChart 只支持一个饼图，并且每个饼图只有 1 个数据系列。大部分图表类型支持多个数据系列。设想一个带有 3 条线的折线图。每条线都表示一个数据系列（例如，这可能是 Sales、COGS 和 Inventory），并且每条线有 4 个点（East、West、North 和 South）。每个数据系列都是针对 1 个或多个轴绘制的。轴可以是 OrdinalAxis 或 NumericAxis。请参阅条形图代码样本以了解如何设置您自己的 BarChart。

OrdinalAxis

OrdinalAxis 实际上是具有字符串标注的轴。这个轴包含数据组的标注。例如，如果正在绘制包含值 3、5、4 和 6 的数据系列，并且这些值来自 East、West、South 和 North 的 Sales 数值，则 OrdinalAxis 将包含标注“East”、“West”、“South”和“North”。这个轴被称为 Ordinal（序数）的原因是数据系列的顺序与标注的顺序匹配。每个标注实际上都是数据点的存储区。

NumericAxis

NumericAxis 是图表上用来绘制（数据系列所包含的）实际数据值的轴。例如，如果有一个包含值 3、5、4 和 6 的数据系列，则 NumericAxis 的范围为 0 到 10（可以控制），记号间隔为 1（可以控制），并且将在图表的左边显示（可以控制）。

图表组件

没有任何一个“图表”组件能够表示所有的图表。图表 UI 模型为每种基本图表（PieChart、BarChart、LineChart、ScatterChart 和 DialChart 等）都提供了一个不同的组件。每种基本图表的 API 都略有差别。例如，对于条形图，可以设置若干个可视属性，如条边框样式和条宽。这些可视属性在 LineChart 中没有类似的属性（尽管线条有厚度，但没有边框或宽度）。因此，BarChart 有 setBarBorder(borderStyle) 方法，而 LineChart 没有。

所有图表类都继承 ChartObject。也可以进行其他逻辑分组，以便以类似的方式处理各种图表类。例如，可能有一些在图表区域绘制网格线的代码适用于所有 RectangularChart 对象。

在使用图表 UI 模型时，一个重点是对 Chart 对象进行强制类型转换以将其转换为适当的类型（请参阅上图）。为了使用 BarChart API，应该首先检查 Chart 对象是否真的是 BarChart，然后相应地对其进行强制类型转换:

```
Component chart = bloxModel.searchForComponent(ModelConstants.CHART);
  // Checks to see if the chart is actually a BarChart
  if (chart != null && chart instanceof BarChart) {
  BarChart barChart = (BarChart) chart;

// Now we can use the specific BarChart APIs
  barChart.setBarWidth(20);
}
```

可以从头开始创建图表对象（请参阅下面的 BarChart 示例），但更常见的情况是，组装人员可能想修改根据 ChartBlox 创建的现有 Chart 对象。为了演示如何最好地做到这一点，请参阅下面的“更改上下文（右键单击）菜单”示例，该示例对 Chart 对象设置定制上下文（右键单击）菜单。

控制图表设置

组装人员可能想配置一些公共项。要了解有关特定 API 的更多详细信息，请参阅 *Blox API Javadoc* 文档，可以通过 DB2 Alphablox 管理页面中的“帮助”菜单打开该文档。

NumericAxis

属性　描述

axisTitle
　　轴的标题。在可能的时候显示（对于饼图，不可能显示轴的标题）。

formatMask
　　设置格式掩码，此格式掩码控制着标记中的数字的显示方式。

scale　设置此轴的最小值、最大值和步长值。

OrdinalAxis

属性　描述

axisTitle
　　轴的标题。在可能的时候显示（对于饼图，不可能显示轴的标题）。

labels　设置显示在每个标记下方的文本标注。

DataSeries

属性　描述

seriesName
数据系列的标题。此标题将显示在图注中。

dataValues
设置数据值。

图注

属性　描述

legendTitle
图注标题显示在图注项的正上方。

position
与 Right、Bottom、TopLeft、TopRight、BottomLeft、 BottomRight 以及 Center 配合使用。Center 将图注放在图表内。

legendLayout
图注项的垂直（Vertical）方向或水平（Horizontal）方向。水平（Horizontal）表示所有图注项显示在一行上。垂直（Vertical）表示在每行上显示 1 个图注项。

ChartTitle、Footnote 和 AxisTitle

这些都是 ChartStatic 对象。

属性　描述

displayText
显示的文本

tooltip
打开 dwellLabelsEnabled 时将显示的工具提示

textStyle
可以控制前景色、字体名称、字体大小和字体角度。

regionStyle
可以控制背景色／图像、边框宽度、边框类型和边框颜色。

图表事件处理

图表组件本身具有与任何其他组件相同的事件处理机制。但是，最终用户可以单击图表的各部分（数据点、标注和图注项等），由于这些图表部分不是完整的组件（它们未扩展 Component 类），所以，对这些事件的处理方式也有所不同。有一个 ChartComponent，它用作 Axis、Legend、AbstractDataSeries 和 ChartStatic（标题和脚注）的超类。每次可以在图表上选择一个 ChartComponent。组装人员可以使用以下示例来在图表控制器（或任何父控制器）中拦截这个选择事件。注意，在示例中，只处理包含特殊属性（Chart.EVENT_ATTR_TARGET）的 SelectedEvent。这个“目标”就是选择的 ChartComponent:

```
chart.setController( new Controller() {
  public void handleSelectedEvent(SelectedEvent event) {
  Chart theChart = (Chart) event.getComponent();
  if (event.getAttribute(Chart.EVENT_ATTR_TARGET) != null) {
  ChartComponent chartComponent =
    theChart.getSelectedChartComponent();
  // If the user clicked on an OrdinalAxis, then figure
  // out the label and print it out
  if (chartComponent instanceof OrdinalAxis) {
  OrdinalAxis axis = (OrdinalAxis) chartComponent;
  Label label = axis.getLabels()[ axis.getSelectedIndex() ];
MessageBox.message( theChart, "OrdinalAxis", "Label "
+ axis.getSelectedIndex() + ": " + label.getDisplayText());
}
  if (chartComponent instanceof Legend
    && theChart instanceof OrdinalChart)
{
  // Remember that the legend items map to 1 to each data series.
  Legend legend = (Legend) chartComponent;
  SingleValueDataSeries dataSeries = ((OrdinalChart)
    theChart).getAllDataSeries()[legend.getSelectedIndex()];
MessageBox.message( theChart, "Legend", "Legend Item "
+ legend.getSelectedIndex() + ": " +
      dataSeries.getSeriesName());
}
  if (chartComponent instanceof AbstractDataSeries
    && theChart instanceof OrdinalChart)
{
  SingleValueDataSeries dataSeries = (SingleValueDataSeries)
     chartComponent;
  Number value = dataSeries.get(dataSeries.getSelectedIndex());
MessageBox.message( theChart, "DataSeries", "Data Point "
+ dataSeries.getSelectedIndex() + ": " + value);
}
}
}
});
```

图表的定制上下文（右键单击）菜单

此示例演示如何使定制右键单击菜单与现有 Chart 对象相关。当最终用户右键单击图表数据点、轴标和图注等时，将显示定制上下文（右键单击）菜单。您需要注意一个有趣的情况，即，每当最终用户执行数据操作（如下寻）或更改图表类型时，都将彻底地重新构建 Chart 对象。每次发生这种情况时，都需要重新使定制右键单击菜单与 Chart 对象相关。每次重新构建 Chart 对象时，都将发送 ComponentRebuiltNotify 事件。handleComponentRebuiltNotify 事件处理程序的作用是在这些情况下重新使上下文菜单与 Chart 对象相关。有一点至关重要，即对 Chart 对象所作的任何修改都必须在 handleComponentRebuiltNotify() 事件处理程序中进行，否则，当图表类型更改时（或者进行了某些数据操作时），您所作的定制将立即丢失:

```
<blox:present id="customRightClick"
width="80%"
height="500">
<%
  PresentBloxModel bloxModel =
    customRightClick.getPresentBloxModel();
  // Find the chartBrixModel and its controller
  bloxModel.addEventHandler( new IEventHandler() {
  public boolean handleComponentRebuiltNotify(
    ComponentRebuiltNotify event)
      throws Exception
{
  Component component = event.getComponent();
```

```
  if (component instanceof ChartBrixModel) {
  ChartBrixModel chartBrixModel = ((ChartBrixModel) component);
  Component chartMaybe =
     chartBrixModel.searchForComponent(ModelConstants.CHART);
    if (chartMaybe != null) {
    Chart chart = (Chart) chartMaybe;
  /*** Make the menu ***/
    Menu headerMenu = new Menu();
headerMenu.add( new MenuItem("headerItem",
"Header Menu Item ..."));
// Add a dedicated controller to the header menu
  headerMenu.setController(new Controller() {
   public void actionHeaderItem(ModelEvent event) {
MessageBox.message(event.getComponent(), "Right Click",
"Test");
}
});
  chart.setRightClickMenu(headerMenu);
  return true;
}
}
      return false;
}
});
%>

<blox:data
dataSourceName="qcc-essbase"
useAliases="true"
query="<ROW (\"All Products\") <ICHILD \"All Products\"
<COLUMN (\"All Time Periods\") <CHILD \"All Time Periods\" !"/>
</blox:present>
```

Blox UI 模型示例

单个工具栏

在本示例中，将两个缺省 DHTML 客户机工具栏组合成单个工具栏。这在增加水平宽度的情况下节省了垂直空间。这些代码将所有导航工具栏按钮追加到标准工具栏末尾并接着除去空的导航工具栏，而不是构造全新的工具栏。由于 UI 模型组件每次只能存在于一个容器中，所以，在将组件添加到容器时，还将从该组件的旧容器中除去该组件。

```
<blox:present id="task1present"
width="800"
height="400">
<%
  // Get the model
  PresentBloxModel model = task1present.getPresentBloxModel();
  // Get the two default toolbars
  Toolbar standard = model.getStandardToolbar();
  Toolbar navigation = model.getNavigateToolbar();
  // Add a separator
  standard.add( ToolbarButton.separator() );
  // Move the buttons (don't use an iterator because
  // the component is changing)
    while (navigation.size() > 0)
      standard.add(navigation.get(0));
// Remove the navigation toolbar and update the toolbar menu
    navigation.delete();
    model.populateToolbarMenu();
%>
</blox:present>
```

禁用上下文（右键单击）菜单

本示例将拦截 Blox 网格上的右键单击事件，并且，将向用户显示一条消息，而不是显示右键单击菜单。由于网格已经有了相关的控制器，所以，本示例将拦截所有 RightClickEvent 的事件处理程序添加到该控制器中。当此事件处理程序返回 true 时，就不再对该事件进行进一步的处理（返回 false 表示允许其他处理程序处理该事件）。

有一个 Blox 属性可以用来禁用右键单击菜单，但本示例的重要之处在于它是定制右键单击行为的基础。例如，处理程序可以根据单元格类型（行头单元格、列头单元格或数据单元格）或者根据用户选择的单元格的内容来启用或禁用右键单击菜单。

```
<blox:present id="task2present"
width="800"
height="400">
<%
  // Get the model
  PresentBloxModel model = task2present.getPresentBloxModel();
// Find the grid and its controller
  GridBrixModel grid = model.getGrid();
    Controller controller = grid.getController();
  //Add custom event handler to intercept the right-click event
  controller.addEventHandler(new IEventHandler() {
  public boolean handleRightClickEvent(RightClickEvent event) {
MessageBox.message( event.getComponent(), "Not allowed",
"Right clicking the grid is not allowed" );
  // Return true to stop the processing this event
  return true;
}
});
%>
</blox:present>
```

定制的文本（右键单击）菜单

当未提供静态页眉或单元格右键单击菜单时（即，当 getHeaderCellRightClickMenu() 或 getCellsRightClickMenu() 返回 null 时），网格 Brix 的上下文（右键单击）菜单是 Blox 组件的菜单栏上找到的“数据”菜单的副本。这是 PresentBlox 或 GridBlox 中新创建的网格 Brix 的缺省行为。定制右键单击菜单有几种方式，从完全替换到将项添加至缺省菜单。使用下表来帮助确定哪个方法最适合于该任务。

表 2. 定制菜单的可能解决方案

需求	解决方案
使用不受当前单元格选择影响的静态菜单替换缺省右键单击菜单。	使用 setHeaderCellRightClickMenu() 和 setCellsRightClickMenu() 方法
根据当前单元格选择使用动态菜单替换缺省右键单击菜单。	截取 GridBrixController 级别的 RightClickEvent 并根据单元格选择提供菜单。确保从事件处理程序返回“true”，以防止对右键单击事件进行缺省处理。有关启动右键单击菜单的详细信息，请参阅 IModelDispatcher 接口。
将菜单项添加至缺省右键单击菜单，在该菜单中，菜单项不依赖于当前单元格选择。	将菜单项添加到菜单栏上的“数据”菜单。这操作只需执行一次，将在右键单击菜单上自动复制这些菜单项。

表 2. 定制菜单的可能解决方案 (续)

需求	解决方案
将菜单项添加至缺省右键单击菜单，在该菜单中，菜单项依赖于当前单元格选择。	截取 BloxModelController 级别的 RightClickEvent，并使用 IModelDispatcher 接口来获取当前右键单击菜单，然后添加定制菜单项。

本示例通过为网格头单元格设置定制右键单击菜单并为网格数据单元格设置另一个定制右键单击菜单来覆盖这种缺省行为。菜单项将向用户显示一个 MessageBox，该 MessageBox 指示了所选单元格的类型以及单元格的值。

虽然本示例添加的右键单击菜单是静态的（即，它们不会根据所选单元格的不同而有所变化），但是，由于显示的消息针对所选单元格进行调整，所以选择菜单项时执行的操作千变万化。为此，与每个菜单项相关的控制器检查当前网格单元格选择并使用该选择来定制消息。

为简单起见，本示例仅检查选择列表中的第一个单元格，该单元格可能不是实际被单击的单元格（如果选择了多个单元格的话）。

```
<blox:present id="task3present"
width="800"
height="400">
<%
  // Get the model
  PresentBloxModel model = task3present.getPresentBloxModel();
// Find the grid and its controller
    final GridBrixModel grid = model.getGrid();
    Controller controller = grid.getController();
    // Make and add the header right click menu
    Menu headerMenu = new Menu();
headerMenu.add(new MenuItem("headerItem","Header Menu Item ..."));
    grid.setHeaderCellRightClickMenu( headerMenu);
// Add a dedicated controller to the header menu
    headerMenu.setController( new Controller() {
    public void actionHeaderItem( ModelEvent event ) {
    // Get the selected cell(s)
        GridCell[] cells = grid.getSelectedCells();
MessageBox.message(event.getComponent(),"Right Click",
"You right clicked on a header cell - " +
         cells[0].getValue());
}
});
    // Make and add the data cell right click menu
Menu cellMenu = new Menu();
cellMenu.add(new MenuItem("cellItem","Cell Menu
Item ..."));
grid.setCellsRightClickMenu(cellMenu);
// Add a dedicated controller to the cell menu
    cellMenu.setController( new Controller() {
    public void actionCellItem(ModelEvent event) {
    // Get the selected cell(s)
        GridCell[] cells = grid.getSelectedCells();
MessageBox.message(event.getComponent(),"Right Click",
"You right clicked on a data cell - " +
         cells[0].getValue());
}
});
%>
```

定制网格布局

本示例显示如何创建一个定制网格布局来在网格单元格被创建时修改各个网格单元格的内容。在本示例中，布局通过对每个头单元格添加 Microsoft Analysis Services 单元格属性作为工具提示来修改单元格（本示例只支持 Microsoft Analysis Services 数据源）。

定制布局标记通过管理网格的大部分细节来为应用程序开发者提供了定制整个网格或各个单元格的方法。这些细节包括在单元格时被使用时处理网格重新构建通知以及处理单元格修改，而不是在网格被构建时直接进行处理。

缺省情况下，在发生下列情况之前，不会实际地创建网格单元格：（1）用户请求包含单元格的页面，或者（2）服务器端代码请求网格中的单元格。通常，最好不要让网格创建所有单元格。

本示例的第一部分演示了 <bloxui:customLayout> 标记，该标记使定制布局类与网格相关。该标记将创建布局类的实例并使其与网格相关。它还管理该布局在布局菜单上的外观，该菜单是一个可选功能部件，它允许用户打开和关闭布局。此外，该标记还将用户的设置保存在用户保存的所有书签中。

```
<blox:present id="task5present" width="800" height="400" visible="true">
<bloxui:customLayout
className="CustomLayout"
showOnLayoutMenu="true"/>
</blox:present>
```

该标记中的 className 属性指定的类必须存在于服务器的类路径中，否则该标记将无法创建布局类的实例。该类本身必须扩展 com.alphablox.blox.uimodel.layout.AbstractLayout。请参阅 AbstractLayout Javadoc 文档以获取所提供的所有方法和服务的完整列表。在本示例中，我们的布局类仅关心单元格的创建，而忽略了网格的创建。

示例的第二部分是一个实际的类，这个类扩展 AbastractLayout 并执行所有工作。getFormatName() 方法返回布局的名称，并且，如果该布局被添加至菜单系统，则该名称将用作菜单项。

所有实际工作都是在 layoutCell() 方法中完成的。每次创建网格单元格时，都会通过引用新创建的单元格来调用 layoutCell() 方法。该方法检查该单元格以了解它是否是头单元格，如果是的话，则将包含成员属性信息的工具提示添加至该单元格。

```
public class CustomLayout extends AbstractLayout {
    protected String getFormatName() {
return "Custom Layout (show MSAS member attributes)";
}
  protected void layoutCell(GridCell gridCell,DataViewBlox dataViewBlox)
    throws Exception {
    // Make sure this is a header cell, and it belongs to the grid Brix
    // (i.e. not added by another layout)
    if (!gridCell.isColumnHeader() && !gridCell.isRowHeader())
return;
    Property[] properties = getHeaderCellProperties(gridCell,
       dataViewBlox);
    if ( properties.length > 0 ) {
    // Create a tooltip with the properties and add to the cell
    StringBuffer buffer = new StringBuffer(200);
    for ( int i=0; i < properties.length; i++ ) {
      if (i > 0)
buffer.append("\r\n");
        buffer.append(properties[i].getName());
```

```
buffer.append("=");
buffer.append(properties[i].getValue());
}
    gridCell.setTooltip( buffer.toString());
}
}
private Property[] getHeaderCellProperties(GridCell gridCell,
    DataViewBlox dataViewBlox) throws ServerBloxException {
    if (!(gridCell instanceof GridBrixCellModel))
      return new Property[0];
    GridBrixCellModel cell = (GridBrixCellModel)gridCell;
  MDBResultSet results =
     (MDBResultSet)dataViewBlox.getDataBlox().getResultSet();
    Axis axis = results.getAxis(cell.isColumnHeader() ?
        Axis.COLUMN_AXIS_ID : Axis.ROW_AXIS_ID);
    Tuple tuple  = axis.getTuple(cell.isColumnHeader() ?
        cell.getNativeColumn() : cell.getNativeRow());
    TupleMember tupleMember = tuple.getMember(cell.isColumnHeader() ?
           cell.getNativeRow() : cell.getNativeColumn());
MDBMetaData meta =
    (MDBMetaData)dataViewBlox.getDataBlox().getMetaData();
Property[] properties =
    meta.getPropertiesOfMember(tupleMember.getUniqueName());
    return properties;
}
}
```

将网格单元格映射至底层的结果集

定制网格布局的这个示例将工具提示以及有关头单元格的唯一名称和数据单元格值的信息添加至每个单元格。此示例演示了两个重要概念:

1. 可以将网格布局类直接放到 JSP 页面中，当布局将只与单个 Blox 配合使用时，这样做可能是合适的。将类代码放在 JSP 文件中可以缩短所有布局的开发调试周期。但是，当以这种方式开发布局时，在完成调试后，应该将类代码放到独立的类文件中。
2. 布局对 GridBrixModel 的 UI 模型转换方法使用 MDBResultSet 来将 UI 模型网格单元格映射到 MDBResultSet 对象。

此示例的第一部分显示如何引用实际 JSP 页面中定义的类。因为大多数 Web 服务器在 JSP 文件每次被编译时都会破坏类名，所以，代码直接从类本身中获取布局的类名。当将布局类放在 JSP 文件中时，应用程序开发者不必关心类路径。

```
<blox:present id="lookupGridCell"
width="700"
height="500">
  <bloxui:customLayout
className="<%= CustomLayout.class.getName() %>"
showOnLayoutMenu="true"/>
</blox:present>
```

此示例的第二部分演示了实现布局的 Java 类。当每个 UI 模型网格单元格被创建时，定制布局将调用 findGridBrixCell() 来获取与所创建的单元格相对应的 MDBResultSet 对象。此方法返回的值将取决于与 UI 模型单元格相匹配的结果集对象（这个对象将是 Cell、TupleMember 或 null）。GridBrixModel 还提供了方法来将单元格从结果集映射到 UI 模型中的单元格。

在实现定制布局时，请记住，UI 模型网格中的单元格顺序与实际 MDBResultSet 中的单元格顺序可能不匹配。对于那些移动行头的蝶形布局之类的布局来说尤其如此。

```
<%!
  public static class CustomLayout extends AbstractLayout {
    protected String getFormatName() {
return "Custom Layout";
}
  protected void layoutCell(GridCell gridCell, DataViewBlox dataViewBlox)
    throws Exception {
  MDBResultSet results =
     (MDBResultSet)dataViewBlox.getDataBlox().getResultSet();
  GridBrixModel grid = (GridBrixModel)gridCell.getGrid();
  Object object = grid.findGridBrixCell( results, gridCell);
  if (object == null) {
gridCell.setTooltip("This cell is not from the MDB result set");
}
  else if (object instanceof Cell) {
gridCell.setTooltip("Cell\r\nValue:" + ((Cell)object).getDoubleValue());
}
   else if (object instanceof TupleMember) {
   TupleMember member = (TupleMember)object;
gridCell.setTooltip("TupleMember" + \r\nUniqueName: " +
      member.getUniqueName() +
"\r\nDimension: " + member.getDimension().getUniqueName() +
"\r\nAxis :" + member.getDimension().getAxis().getIndex());
}
  else
gridCell.setTooltip("Unexpected object: "
     + object .getClass().getName());
}
}
%>
```

Javadoc 文档

要获取完整的 UI 模型 Javadoc 文档，请访问 DB2 Alphablox 信息中心。服务器端 Javadoc 文档列示了所有 UI 模型类及其方法。可以通过在 DB2 Alphablox 管理页面中单击“帮助”菜单选项来打开 *Blox API Javadoc 文档*。

第 11 章 DHTML 客户机 API

本主题阐述 DHTML 客户机 API，此 API 使您能够使用 JavaScript 方法来方便地访问服务器端应用程序逻辑和 API，并允许组装人员利用客户机端脚本编制功能来通过将导航、某些 UI 处理和输入验证工作移到服务器外部进行来提升应用程序的价值。

DHTML 客户机 API 概述

为 DHTML 客户机构建的应用程序的核心逻辑由服务器端组件（如 UI 模型、scriptlet、bean 和其他支持类）构成。因此，DHTML 客户机 API 的目标是使您能够方便地访问服务器端应用程序逻辑和 API，而不是在客户机上显示基于 RPC 的大型 API。它还允许组装人员利用客户机端脚本来通过将导航、某些 UI 处理和输入验证工作移到服务器外部进行来提升应用程序的价值。

DHTML 客户机 API 是一个客户机框架，它提供诸如事件处理、错误处理、通信服务以及 JavaScript 代码的 RPC 机制等服务。

使用 DHTML 客户机 API

当外围页面中的 HTML 或 JavaScript 需要与页面上的一个或多个 Blox 进行交互时，将使用 DHTML 客户机 API。此客户机 API 的主要用途包括:

- **调用服务器端应用程序逻辑**: DHTML 客户机 API 提供了方法来直接对服务器端 bean 调用方法。并且，还将把服务器端 bean 的返回值返回到客户机并将其转换为适当的 JavaScript 对象。
- **事件处理**: 通过 API，组装人员可以将事件发送至服务器以模拟用户与 UI 之间的交互。可以使用 JavaScript 来创建事件对象（如单击事件），并且提供了用来将该事件发送至服务器的方法。这使 Blox 框架外部的 HTML 按钮和其他控件能够模拟用户交互。事件也可以用来更改模型内的组件状态。
- **拦截事件**: 可以将 JavaScript 方法注册为由页面上所有 Blox 生成的所有事件的侦听器。事件侦听器可以选择忽略事件或者让该事件正常地被处理。可以编写 JavaScript 来更改 UI 行为和／或处理客户机上的一些用户选择。
- **轮询服务器以了解更改**: 客户机 API 框架与服务器一起自动处理所有 UI 更新以及客户机与服务器之间的信息传递。但是，组装人员能够显式地进行轮询以了解更改。通常，如果应用程序在客户机框架外部进行更改（这种情况的常见示例包括通过其他框架或者通过使用 HTTP 通信设施（如 XMLHTTP 对象）来与服务器进行通信），此功能就非常有用。
- **处理由服务器或通信层返回的错误**: 客户机框架将调用 JavaScript 方法来处理服务器错误和通信错误。客户机代码可以注册它自己的错误处理程序以处理这些错误。

本节的随后内容将提供 DHTML 客户机 API 的常见用法示例。

DHTML 客户机 API 框架

客户机框架由两个主要的对象（即 `BloxAPI` 对象和 `Blox` 对象）组成，它们为 UI 提供了强大的功能并处理与服务器进行的通信。还提供了一些相关的实用程序对象。

BloxAPI 对象

BloxAPI JavaScript 对象包含许多由页面上的所有 Blox 使用的一般服务。它在客户机与服务器之间提供通信服务，并且为 JavaScript 代码提供了便捷的 RPC 机制。每个框架都刚好有一个 BloxAPI 对象，该对象控制着服务器与该框架中所有 Blox 之间的所有输入和输出通信。

BloxAPI 对象处理下列各项:

- 轮询服务器以了解更改
- 提供用于进行事件和错误管理的 API
- 将更改分派到框架中的各种 Blox
- 处理通信错误和服务器错误
- 提供用于 RPC 访问的 API
- 提供用于发送事件的 API

Blox 对象

框架中的每个 Blox 都具有相关联的 JavaScript Blox 对象。该 Blox 对象负责执行下列操作:

- 对来自服务器的更改通知进行响应
- 对忙状态和忙指示进行响应和处理
- 提供与 DB2 Alphablox 4.x 兼容的方法: isBusy()、updateProperties()、flushProperties()、call() 和 setDataBusy()
- 处理和管理与 Blox 相关联的对话框
- 处理与 Blox 相关联的右键单击菜单

实用程序对象

除了 BloxAPI 和 Blox 对象以外，框架还支持一些实用程序对象，其中，最重要的实用程序对象包括:

- *xxx*Event - 客户机可以发出的每种模型事件类型的特定对象。示例: ClickEvent
- Grid - 提供对一些网格属性（如选择的可视单元格列表）的只读访问
- Exception - 用来将服务器异常传达给客户机的对象

发送事件

UI 模型显示了许多可以由客户机发出的事件，如 ClickEvent。对于其中的每一个事件，DHTML 客户机 API 都会定义 JavaScript 对象。因此，可以使用 JavaScript 来创建事件对象并将该事件发送给服务器。这使 Blox 框架外部的 HTML 按钮和其他控件能够模拟用户交互。事件也可以用来更改定制模型组件的状态。例如，以下 HTML 代码将把 ClickEvent 发送至 UID 为 UID 的模型组件:

```
<input type=button value="Show Dialog"
onclick="bloxAPI.sendEvent(new ClickEvent('container', UID));" >
```

要获取向客户机显示的事件列表，请参阅 *Developer's Reference* 一书。

从 JavaScript 中启动 Blox UI 模型事件

可以从 JavaScript 中生成 UI 模型事件并将该事件发回给服务器。这样就允许页面上的正规 HTML 控件模拟用户单击 Blox 用户界面的操作。例如，可以关闭 Blox 的菜单，但是可以将 HTML 按钮放在页面上以便向用户提供一些 UI 功能。

以下示例将创建一个 HTML 按钮，该按钮在被单击时将调用“数据选项”菜单项。

```
<blox:present id="samplePresent"
width="700"
height="500">
</blox:present>
<%
/* In order to send an event from the client, we need the component's UID */
  BloxModel model = samplePresent.getBloxModel();
  Component component = model.searchForComponent(
    ModelConstants.DATA_OPTIONS );
    int uid = component.getUID();
%>
<input type=button value="Data Options"
onclick="bloxAPI.sendEvent(new ClickEvent('samplePresent',<%= uid %>));">
```

input 元素的 onclick 事件处理程序使用 BloxAPI 来将 ClickEvent 发送至服务器。ClickEvent 是一个 JavaScript 对象，它接收 Blox 名以及目标组件 UID。由于 UID 是动态指定的，因此，当页面被请求时，代码必须在模型中查找该 UID。

拦截事件

可以使用两个设施来在客户机端拦截事件:

1. JavaScript 代码可以为所有客户机端事件注册侦听器。这表示可以拦截、检查以及忽略或处理用户执行的每项操作。这种方法提供了对用户与 UI 之间进行的每项交互的控制。例如，用户每次选择菜单项时，都将生成 ClickEvent，它包含 Blox、该菜单项的 UID 以及该菜单项的名称。
2. UI 模型提供了一个 ClientLink 对象，此对象可以与大部分具有被单击操作概念（即，生成 ClickEvent）的模型组件相关。ClientLink 对象导致视图层使用它自己的逻辑来处理用户的单击，而不是将该操作发回给服务器。对于 DHTML 客户机，将在客户机上以 JavaScript 调用形式或者通过打开新浏览器窗口来处理任何具有相关 ClientLink 的 Component。

拦截客户端事件

本示例演示如何在客户机上拦截 UI 模型事件。当 UI 模型事件执行某些不要求服务器参与的客户机端操作时，开发者就将在客户机上拦截 UI 模型事件。

将对 UI 生成的所有事件调用 JavaScript eventHandler() 函数。由于此处理程序能够看到所有事件，所以，代码必须检查事件类型以及目标 UID（或名称）才能拦截特定的 UI 事件。处理程序返回 false 将允许该事件被处理以及被发送至服务器。返回 true 将停止对该事件的所有进一步处理。

以下示例是客户机端 JavaScript 代码:

```
<script>
function eventHandler(event) {
alert( "At handler for event " + event.getEventClass() +
```

```
" on component " + event.getDestinationName() +
" UID " + event. getDestinationUID() );
      return false;
}
</script>
bloxAPI.addEventListener(eventHandler);
```

直接从用户界面中调用 JavaScript

可以指示组件直接调用客户机端 JavaScript，而不是在客户机上拦截 UI 生成的每个事件。在这种情况下，组件不会将 ClickEvent 事件发送至服务器。

用于对可单击组件指定 JavaScript 方法的代码在服务器上。以下示例在“数据”菜单中查找“数据选项”菜单项并强制该菜单项调用一个 JavaScript 方法（另外，还可以将该菜单项设置为装入浏览器 URL）。

```
<blox:present id="samplePresent" width="700" height="500">

<%
  // Find the component
  BloxModel model = samplePresent .getBloxModel();
  Component component = model.searchForComponent(
    ModelConstants.DATA_OPTIONS );
// Create a client link using javascript:protocol method
ClientLink link = new ClientLink("javascript:
myJavaScriptFunction( );" );
// Set the link on the component
component.setClientLink( link );
%>
</blox:present>
```

当“数据选项”菜单项被单击时，客户机将调用 JavaScript myJavaScriptFunction() 函数，而不是将该事件发送至服务器。这里有一个假定，即页面已经定义了该 JavaScript 函数。

异常处理

当使用 callBean 来调用服务器端代码时，如果服务器端方法有可能会抛出异常，则您的代码应该准备好处理 Java 异常。给定客户机 bean myBean 的 exceptionThrower() 方法，JavaScript 代码在处理结果之前应该检查返回值以确定异常是否已被抛出，如下所示:

```
var retval = myBean.exceptionThrower();
if (retval.constructor == Exception ) {
alert( "Exception returned: " + retval );
} else {
    // Process the response
}
```

使用 DHTML 客户机 API 来调用服务器端逻辑

实际上，可以通过三个方法来从 DHTML 客户机中调用服务器端逻辑。您选择的方法取决于要让服务器在多大程度上使过程自动化。下面按照自动化程度从最低到最高的顺序列示了这些方法。

BloxAPI.call() 和 Blox.call() 方法

这是在 DB2 Alphablox 4 中调用的同一个 call() 方法。此方法允许您调用服务器上的 URL 并以 URL 参数形式传递自变量。此方法可以调用 rmi.jsp 以便自动地传递自变量以及调用 bean 方法。必须对返回给 JavaScript 的值进行解析并将它们转换为所需的数据类型。

例如，以下代码使用 bloxAPI.call() 方法来对一个 bean（MyBean）调用方法，该方法将切换数据布局面板的可视性。

```
<%@ taglib uri='bloxtld' prefix='blox'%>
<%@ taglib uri='bloxuitld' prefix='bloxui'%>
<blox:present id="callpresent"
visible="false"
width="600"
height="500"
chartAvailable="false" >
<blox:grid bandingEnabled="true" />
<blox:data bloxRef="calldata" />
</blox:present>
<jsp:useBean class="MyBean" scope="session" id="myBean">
<%
  myBean.setBlox(callpresent);
%>
</jsp:useBean>
<html>
<head>
   <blox:header />
<script>
// Use call to invoke method on the bean
function showDataLayout(show) {
  var result =
bloxAPI.call("rmi.jsp?bean=myBean&method=showDataLayout&arg1="+show);
alert("Result type: " + typeof result + "\r\n\r\n" + result);
}
</script>
</head>
<body>
<blox:display bloxRef="callpresent" />
<input type="button" value="Hide Data Layout"
onclick="showDataLayout(false);" >
<input type="button" value="Show Data Layout"
onclick="showDataLayout(true);" >
 </body>
</html>
```

BloxAPI.callBean() 方法

此方法将调用服务器上的一个 Java bean，这类似于组合使用上面描述的 call 方法和 rmi.jsp。此方法与该组合的不同点如下所示:

- 在查找和调用 bean 方法时，它直接与服务器配合工作。不涉及其他 JSP 文件（即，不需要 rmi.jsp）。
- 可以为输出方法自变量指定数据类型。
- 返回值将被转换为真正的 JavaScript 对象。
- 支持将大部分简单数据类型和数组用作自变量和返回值。

要在以上示例中使用 callBean，只需将 call() 调用替换为以下内容:

```
var result=bloxAPI.callBean("myBean","showDataLayout",new Array(show));
```

clientBean（<blox:clientBean>）标记

Blox clientBean 标记（<blox:clientBean>）可以嵌套在 <blox:header> 标记中，并且将导致服务器为指定的 Java bean（即 Blox）生成一个 JavaScript 对象。要在以上示例中使用 <blox:clientBean>，请将客户机 bean 标记合并到 Blox header 中。然后，开发者就可以进行正常的 JavaScript 方法调用了:

```
  <blox:header>
<blox:clientBean name="myBean" />
</blox:header>
<script>
// Use ClientBean to invoke method on the bean
function showDataLayout( show ) {
  var result = myBean.showDataLayout( show );
alert( "Result type: " + typeof result + "\r\n\r\n" + result );
}
</script>
```

要点: 在本示例中，未在 <blox:clientBean> 标记中指定任何方法。因此，将为该 bean 中的每个公用方法生成一个 JavaScript 方法。对于带有较多方法的 bean 来说，这可能会产生相当大的开销。因此，建议组装人员显式地列示要为其生成 JavaScript 的方法，并建议组装人员将方法数目保持在最低水平。

注意，在使用 <blox:clientBean> 时，唯一的限制是传递的以及返回的自变量需要受 JavaScript 支持，实际上，这将受支持的自变量限制为基本类型和数组。

将 <blox:clientBean> 与服务器端 Blox 组件配合使用

还可以使用 <blox:clientBean> 标记来从客户机中访问服务器端 Blox。以下示例显示了如何为 PresentBlox 生成 JavaScript 对象:

```
  <blox:header>
<blox:clientBean name="myPresentBlox">
<blox:method name="setDividerLocation">
<blox:method name="setChartFirst"/>
    </blox:clientBean>
</blox:header>
```

要点: 在本示例中，展示了两个方法。在给定对大部分 SSPM Blox 对象可用的方法数目的情况下，必须在 header 中的 clientBean 标记中显式地列示所使用的方法。

为了在 header 中使用服务器端 Blox，对该 Blox 定义 visible="false"，然后使用 <blox:display> 标记来在 HTML 主体中显示该 Blox。

当服务器端 Blox 与 clientBean 配合使用时，将进行下列特殊处理:

- 客户机上的 bean 名称在末尾追加 API。无论服务器端 Blox 具有什么类型都会执行此操作。在以上示例中，实际的 JavaScript 对象将被命名为 myPresentBloxAPI。这样做的原因是，在大多数情况下，在页面上已经有了由 DHTML 客户机添加的 JavaScript 对象。
- 为了方便起见，如果 DHTML 客户机找到 Blox 名以 API 结尾的 JavaScript 对象，它将允许开发者直接对主 DHTML 客户机的 Blox 对象调用方法。因此，即使客户机 bean 名为 myPresentBloxAPI，也可以直接对 myPresentBlox Blox 对象调用方法。例如，myPresentBlox.setChartFirst() 和 myPresentBloxAPI.setChartFirst() 都将首先设置图表。

- 如果服务器端 Blox 具有 DataBlox 或其他嵌套的 Blox（如嵌套在 PresentBlox 中的 Blox），则可以在客户机上访问嵌套的 Blox，而不必创建另一个客户机 bean 部分。为此，对前缀为 data、grid、chart、dataLayout、toolbar 和 page 的父 Blox 列表添加方法。要调用该方法，请对主 Blox 使用适当的 getter，例如，myPresentBlox.getDataBlox().connect()。

以下示例是一个完整的 JSP 页面，它演示了嵌入的 Blox 的用法以及 API 后缀。注意，GridBlox 的客户机 bean 部分中的 data.setQuery，它使该 DataBlox 方法可供 JavaScript 代码使用。

```
<%@ page import="com.alphablox.blox.uimodel.*"%>
<%@ taglib uri='bloxtld' prefix='blox'%>
<%@ taglib uri='bloxuitld' prefix='bloxui'%>
<blox:data id="gridDB" ... />
<blox:grid id="grid" width="700" height="500">
<blox:data bloxRef="gridDB" />
</blox:grid>

<html>
<head>
  <blox:header>
<blox:clientBean name="grid">
<blox:method name="setBandingEnabled" />
<blox:method name="isBandingEnabled" />
<blox:method name="data.setQuery" />
<blox:method name="data.connect" />
    </blox:clientBean>
<blox:clientBean name="gridDB">
<blox:method name="setQuery" />
<blox:method name="connect" />
    </blox:clientBean>
</blox:header>
</head>
<body>
...
<!--
   Calling DataBlox methods via the GridBlox. Since the GridBlox
   is a DHTML Blox and appears on the page, the API suffix is optional.
-->
<input type="button" value="Set query via grid"
onclick="grid.getDataBlox().setQuery('!');
grid.getDataBlox().connect( );">
<!-- Calling datablox methods directly on the DataBlox. Note that
   here the API suffix is mandatory because there is not DHTML client
   Blox for the DataBlox.
-->
<input type="button" value="Set query via datablox"
onclick="gridDBAPI.setQuery('!'); gridDBAPI.connect();">
<input type="button" value="Toggle grid banding"
onclick="grid.setBandingEnabled(!grid.isBandingEnabled());">
<blox:display bloxRef="grid" />
 </body>
</html>
```

DHTML 客户机 DOM API

DHTML 客户机广泛地使用 Internet Explorer DOM。当用户与客户机进行交互时，客户机就会更新 DOM 的某些部分。由于这是 DHTML 客户机的部分实现，所以 DHTML 客户机创建的 DOM 对象和属性在将来的版本中会更改。

要点：由于实现在将来有可能会更改，所以，开发者不应该编写客户机端代码来处理或遍历由 DHTML 客户机生成的 DOM。

第 251 页的第 26 章，『DHTML 客户机 DOM API』 阐述了 DHTML 客户机 DOM API。

使用多个框架

DHTML 客户机将应用程序中的每个框架都视为独立的实体。这表示每个包含 `<blox:header>` 标记的框架都有自己的客户机 API 框架 `BloxAPI` 对象。就服务器和客户机 API 而言，不同框架中的 Blox 也可以位于不同的浏览器中。

由于不同框架中的 Blox 被视为独立的实体，所以，在下列情况下可能会产生意外的结果:

1. 不同框架中的 Blox 引用公共 DataBlox。在这种情况下，对一个框架中的 Blox 执行钻取或其他导航操作不会导致另一框架中依赖于同一 DataBlox 的 Blox 立即更新。在实践中，这种情况应该不会经常出现，甚至完全不会出现。
2. 在一个框架中执行的服务器端代码将修改或以别的方式影响另一框架中的 Blox。

在这两种情况下，只有引起修改的框架中的 Blox 才会立即被更新。对于其他框架中的 Blox 来说，在那些框架执行自动轮询之前，那些 Blox 不会被更新。

如果确实发生这种情况，则具有缺省状态的自动轮询并不足够，这是因为自动轮询可能需要长达两分钟的时间才能更新所有框架中的 Blox。下面是一些建议的选项:

1. 在每个带有受影响的 Blox 的框架中，使用`BloxAPI` 对象来执行手工轮询。
2. 将轮询计时器的轮询时间间隔缩短为小于缺省值，以便更快地进行轮训。
3. 通过将 Blox 放在单个框架中或者通过不允许不同框架中的 Blox 依赖于同一个 DataBlox 来彻底地避免这种情况。

刷新页面

DHTML 客户机在进行更新时，并不会通过更改 HTML 元素内容来刷新页面。因此，当用户与客户机进行交互时，HTML 将不断地更改以便显示新信息。但是，浏览器不会跟踪任何这些 HTML 更改。而是在页面第一次被请求时浏览器对接收到的 HTML 进行高速缓存。当您查看源文件时，浏览器也将显示初始页面的 HTML。

要点: 由于 DHTML 客户机执行增量页面更新，所以，通过浏览器的“查看源文件”选项查看的 HTML 源代码通常与浏览器的当前状态不匹配。这会加大调试难度。

当用户刷新页面或使用浏览器的“后退”按钮返回至某个页面时，浏览器将复原该页面的高速缓存版本。由于对 DHTML 客户机进行的更改是在服务器上维护的，因此，客户机和服务器有办法检测和处理这种情况以确保用户正在查看 Blox 的最新表示。这种机制的副作用是，当刷新页面或使用“后退”按钮时，您可能会短暂地看到 Blox 的原始状态，接着它们会更新为显示当前状态。

第 12 章 使用服务器端事件过滤器和侦听器来捕获事件

您可以捕获数据分析事件并在服务器上处理该事件之*前*或之*后*执行定制操作。例如，当用户用户向下钻取某个成员时，在服务器上处理事件之前，您可以执行某些检查来了解用户是否具有执行该操作的权限。这允许您在必要时取消该操作。或者，每当有人向下钻取数据库的某些敏感部分时，您可能要发送一个电子邮件给金融部门。事件过滤器是服务器端对象，它们允许您在服务器上实际处理事件之*前*捕获用户数据分析事件（如向下钻取或旋转）。事件侦听器允许您在用户事件已处理之*后*得到通知。

事件过滤器的一个重要方面是：它们在操作执行时但在实际处理事件之*前*被触发，因此允许应用程序在操作执行之前删除操作。例如，在用户单击成员以向下钻取时但在数据库中执行向下钻取操作之前将发生 `DrillDownEvent`，并且新的数据将返回到客户机。另一方面，在服务器上成功完成事件之后，事件侦听器允许您执行其他操作。例如，当完成仅隐藏事件之后，您可能要更新另一个 Blox、处理因事件副作用而引起的异常或根据事件结果将消息发送回客户机。可以使用事件侦听器来达到此目的。仅当事件完成而没有错误时才会触发事件侦听器。

事件过滤器对象

事件过滤器对象是服务器端对象，它允许您在实际处理事件之*前*捕获某些用户事件（如向下钻取或旋转）并执行操作。

有两种类型的事件过滤器对象。

- 与 DataBlox 相关的：您可以捕获以下数据分析操作：折叠、向下钻取、深入钻取、向上钻取、展开、仅隐藏、仅保持、成员选择、旋转、仅除去、全部显示、仅显示、交换轴和数据查询。
- 与书签相关的：您可以捕获以下与书签相关的事件：删除书签、装入书签、重命名书签和保存书签。

要使用事件过滤器，您首先需要使用公共 `addEventFilter()` Blox 方法来添加特定事件过滤器对象。在您将事件过滤器添加至 DataBlox、PresentBlox 或其他用户界面 Blox 之后，您就可以编写自己的类来实现相应事件过滤器对象并可以指定在实际处理事件之前要执行的操作。

要执行事件后处理，应使用事件侦听器。有关事件侦听器和这两个对象的用法比较的详细信息，请参阅『事件侦听器对象』。

事件侦听器对象

事件侦听器对象是服务器端对象，它们允许您在发生某些用户事件（如向下钻取或旋转）时得到通知并在处理事件之后执行某些操作。有三种类型的事件侦听器对象。

- 与 DataBlox 相关的。您可以捕获以下数据分析操作的完成情况信息：折叠、向下钻取、深入钻取、向上钻取、展开、仅隐藏、仅保持、成员选择、旋转、仅除去、全部显示、仅显示、交换轴和数据查询。
- 与书签相关的。您可以捕获以下与书签相关的事件的完成情况信息：删除书签、装入书签、重命名书签和保存书签。

- 与 ChartBlox 相关的。您可以在用户更改页面过滤器时捕获事件。

当用户触发的事件（如从 Blox 用户界面交换轴）完成时，相应的事件侦听器将得到通知。要使用事件侦听器，您首先需要使用公共 addEventListener() Blox 方法来添加特定事件侦听器对象。在您将事件侦听器添加至 DataBlox、PresentBlox 或其他用户界面 Blox 之后，您就可以编写自己的类来实现相应事件侦听器对象的类并可以指定当事件完成时要执行的操作。

要执行事件前处理，应使用事件过滤器。有关这两个对象的用法比较，请参阅第 95 页的『事件侦听器与事件过滤器的比较』。

使用事件过滤器和事件侦听器

事件过滤器对象是 com.alphablox.blox.event 包的一部分。您必须在任何使用这些对象的 JSP 文件的开头使用以下 JSP import 语句:

```
<%@ page import="com.alphablox.blox.filter.*" %>
```

此包包括用于各种事件的过滤器的接口。您将需要定义实现这些接口的类，以截取您要捕获的特定事件。这些接口的名称全部以单词 Filter 结束，如 BookmarkDeleteFilter、DrillDownFilter、ExpandFilter 和 HideOnlyFilter。这些过滤器具有相应的方法，如 bookmarkDelete()、drillDown()、expand() 和 hideOnly()，您可以实现这些方法以指定您自己的操作。所有这些方法都需要相应的事件对象作为要处理的输入。这些事件对象名全部以单词 Event 结束，如 BookmarkDeleteEvent、DrillDownEvent、ExpandEvent 和 HideOnlyEvent。

事件侦听器对象是 com.alphablox.blox.event 包的一部分。您必须在任何使用这些对象的 JSP 文件的开头使用以下 JSP import 语句:

```
<%@ page import="com.alphablox.blox.event.*" %>
```

此包包括用于各种事件的侦听器的接口。事件侦听器的使用方式与事件过滤器的使用方式非常相似。您将需要定义实现这些接口的类，以截取您想要在其完成时得到通知的特定事件。这些接口的名称全部以单词 Listener 结束，如 BookmarkDeleteListener、DrillDownListener、ExpandListener 和 HideOnlyListener。这些侦听器具有相应的方法，如 bookmarkDelete()、drillDown()、expand() 和 hideOnly()，您可以实现这些方法以指定您自己的操作。所有这些方法都需要相应的事件对象作为要处理的输入。这些事件对象名全部以单词 Event 结束，如 BookmarkDeleteEvent、DrillDownEvent、ExpandEvent 和 HideOnlyEvent。

例如，如果您要检查要执行向下钻取操作的用户是否被允许这样做，您需要执行以下操作:

1. 使用方法 addEventFilter(*YourDrillDownEventFilter*) 将一个服务器端向下钻取事件过滤器添加至 DataBlox:

   ```
   <blox:present id="myPresent">
   ...
   <%
   myPresent.getDataBlox().addEventFilter(new DDFilter() );
   %>
   </blox:present>
   ```

   在以上示例中，DDFilter 是向下钻取事件过滤器对象的名称。
2. 使您的向下钻取事件过滤器对象实现 DrillDownFilter 接口:

```
<%!
public class DDFilter implements DrillDownFilter
{
//more code here....
}
%>
```

3. 添加在调用 drillDown 方法时要执行的操作。该方法将 DrillDownEvent 对象作为输入。

```
<%!
public class DDFilter implements DrillDownFilter
{
BloxModel model;
// drillDown is the method to implement to capture a drilldown
// events. It takes a DrillDownEvent object as input.
public void drillDown( DrillDownEvent dde ) throws Exception
{
DataBlox blox = dde.getDataBlox();
StringBuffer msg = new StringBuffer("DRILL DOWN event on " +
blox.getBloxName() + "\n");
msg.append("With Axis ID: " + dde.getAxisIndex() + ", ");
msg.append("Nest level: " + dde.getNestLevel() + ", ");
msg.append("Member index: " + dde.getMemberIndex() + ", and ");
msg.append("TupleMember: " + dde.getMember().getDisplayName());
MessageBox msgBox = new MessageBox(msg.toString(), "DrillDown Filter
Message", MessageBox.MESSAGE_OK, null);
model.getDispatcher().showDialog(msgBox);
}
}
%>
```

在 Blox 定制标记内放置添加和除去方法

要确保在每次重新装入页面时不添加新事件，在 JSP 页面上的 Blox 定制标记内放置使用 addEventFilter() 方法的代码。例如，以下代码创建一个 Blox 并添加一个当用户在成员上执行向下钻取操作时将调用的过滤器:

```
<%@ taglib uri = "bloxtld" prefix = "blox"%>
<%@ page import="com.alphablox.blox.filter.*" %>
<blox:present id="myPresent">
<blox:data .../>
<%
myPresent.getDataBlox().addEventFilter(new DDFilter() );
%>
</blox:present>
```

要确保在每次重新装入页面时不添加新事件，在 JSP 页面上的 Blox 定制标记内放置使用 addEventListener() 方法的代码。例如，以下代码创建一个 Blox 并添加一个当用户（仅）隐藏成员时将调用的侦听器:

```
<%@ taglib uri = "bloxtld" prefix = "blox"%>
<%@ page import="com.alphablox.blox.event.*" %>
<blox:present id="myPresent">
<blox:data .../>
...
<%
myPresent.getDataBlox().addEventListener(new HideOnlyHandler() );
%>
</blox:present>
<%!
public class HideOnlyHandler implements HideOnlyListener
{
public void hideOnly( HideOnlyEvent hoe)
{
```

```
...// custom actions here
}
}
%>
```

完整的 drillDownEventFilter 示例

这个完整的示例显示当触发向下钻取操作时拦截向下钻取操作的方式，以及使用 MessageBox UI 模型组件写出输出的方式。

```
<%@ taglib uri="bloxtld" prefix="blox"%>
<%@ page import="com.alphablox.blox.filter.*,
com.alphablox.blox.uimodel.core.MessageBox,
com.alphablox.blox.uimodel.BloxModel,
com.alphablox.blox.DataBlox" %>
<html>
<head>
   <blox:header/>
</head>
<body>
<blox:present id="myPresent">
<blox:data dataSourceName="QCC-Essbase" query="!" />
<% myPresent.getDataBlox().addEventFilter(new
DDFilter(myPresent.getBloxModel()) ); %>
</blox:present>
 </body>
 </html><%!
public class DDFilter implements DrillDownFilter
{
BloxModel model;
public DDFilter(BloxModel model) {
this.model = model;
}
// drillDown is the method to implement to capture a drilldown
// event. It takes a DrillDownEvent object as input.
public void drillDown( DrillDownEvent dde ) throws Exception
{
DataBlox blox = dde.getDataBlox();
StringBuffer msg = new StringBuffer("DRILL DOWN event on " +
blox.getBloxName() + "\n");
msg.append("With Axis ID: " + dde.getAxisIndex() + ", ");
msg.append("Nest level: " + dde.getNestLevel() + ", ");
msg.append("Member index: " + dde.getMemberIndex() + ", and ");
msg.append("TupleMember: " + dde.getMember().getDisplayName());
MessageBox msgBox = new MessageBox(msg.toString(), "DrillDown Filter
Message", MessageBox.MESSAGE_OK, null);
model.getDispatcher().showDialog(msgBox);
}
}
%>
```

通过将 addEventFilter() 方法放置在 Blox 定制标记中，可以确保在每次重新装入页面时，您不会添加多个过滤器。在此示例中，在发生向下钻取事件之前，创建的类显示一个消息对话框，它包含有关当前向下钻取操作的信息。

您可以在同一事件上任意添加多个过滤器，将以添加它们的顺序处理它们，或者在取消事件之前处理它们。

在『与数据进行交互』一节的“Blox 样本程序”中提供了此示例。

完整的 drillDownEventListener 示例

这个完整的示例显示当发生向下钻取操作获得通知的方式以及使用 MessageBox UI 模型组件写出输出的方式:

```
<%@ page import="com.alphablox.blox.event.*,
com.alphablox.blox.uimodel.core.MessageBox,
com.alphablox.blox.uimodel.BloxModel" %>
<%@ page import="com.alphablox.blox.DataBlox" %>
<%@ taglib uri="bloxtld" prefix="blox" %>
<html>
<head>
   <blox:header/>
</head>
<body>
<blox:present id="myPresent2">
<blox:data
dataSourceName="QCC-Essbase"
query="!" />
<% myPresent2.getDataBlox().addEventListener( new
SimpleListener(myPresent2.getBloxModel()) );    %>
</blox:present>
 </body>
 </html><%!
public class SimpleListener implements DrillDownListener
{
BloxModel model;
public SimpleListener(BloxModel model) {
this.model = model;
}
public void drillDown(DrillDownEvent event) throws Exception {
DataBlox blox = event.getDataBlox();
StringBuffer msg = new StringBuffer("DRILL DOWN event on " +
blox.getBloxName() + "\n");
msg.append("With Axis ID: " + event.getAxisIndex() + ", ");
msg.append("Nest level: " + event.getNestLevel() + ", ");
msg.append("Member index: " + event.getMemberIndex() );
MessageBox msgBox = new MessageBox(msg.toString(), "DrillDown
Listener Message", MessageBox.MESSAGE_OK, null);
model.getDispatcher().showDialog(msgBox);
}
}
%>
```

您可以在同一事件上任意添加多个侦听器，将以添加它们的顺序处理它们。

事件侦听器与事件过滤器的比较

事件侦听器用来通知事件是否已成功完成，而事件过滤器用来在服务器接收到事件但在处理事件之前截取该事件。事件侦听器和事件过滤器的实现过程非常相似。下表提供了这两个过程的相似处和差别的总结。

	事件侦听器	事件过滤器
接收到通知时	在处理事件之后	在处理事件之前
包	com.alphablox.blox.event	com.alphablox.blox.filter
包中的接口	以单词 Listener 结束的所有接口，如 DrillDownListener 和 RemoveOnlyListener	以单词 Filter 结束的所有接口，如 DrillDownFilter 和 RemoveOnlyFilter

	事件侦听器	事件过滤器
要实现的方法	这些侦听器有一个对应的方法，如 drillDown() 和 removeOnly()，该方法采用对应的事件对象作为自变量：drillDown(DrillDownEvent) removeOnly(RemoveOnlyEvent)	这些过滤器有一个对应的方法，如 drillDown()、drillDown() 和 removeOnly()，该方法采用对应的事件对象作为自变量：drillDown(DrillDownEvent) removeOnly(RemoveOnlyEvent)
事件	事件对象名与事件过滤器中的相同。	事件对象名事件与事件侦听器中的相同。然而，这些事件有一个 cancelEvent() 和一个 isCanceled() 方法，而事件侦听器中没有这两个方法。

您可以为指定的事件创建一个既能进行事件前处理也能进行事件后处理的事件处理程序。例如:

```
<%!
public class DDHandler implements DrillDownFilter, DrillDownListener
{
public void drillDown(DrillDownEvent event) throws Exception {
// actions to take before the event is processed
}
public void drillDown(com.alphablox.blox.event.DrillDownEvent event) {
// actions to take after the event has been processed
}
}
%>
```

然而，因为在事件过滤器和事件侦听器包中的事件对象具有相同的名称，所以，如果您要使用相同类来进行事件前和事件后的处理，则应指定包括包信息的完整类名。

用于实现事件过滤器的方法

要创建事件过滤器，您必须编写用于实现下面列示的一个或多个事件过滤器方法的类。下表列示要捕获的事件、为了捕获该事件要实现的方法以及该过滤器事件的支持方法。

（当用户执行操作时）要捕获的事件	要实现的接口	可用的事件方法
书签：删除	BookmarkDeleteFilter 中的 bookmarkDelete(BookmarkDeleteEvent)	BookmarkDeleteEvent 方法
书签：装入	BookmarkLoadFilter 中的 bookmarkLoad(BookmarkLoadEvent)	BookmarkLoadEvent 方法
书签：重命名	BookmarkRenameFilter 中的 bookmarkRename(BookmarkRenameEvent)	BookmarkRenameEvent 方法
书签：保存	BookmarkSaveFilter 中的 bookmarkSave(BookmarkSaveEvent)	BookmarkSaveEvent 方法
折叠	CollapseFilter 中的 collapse(CollapseEvent)	CollapseEvent 方法
数据排序	DataSortFilter 中的 dataSort(DataSortEvent)	DataSortEvent 方法
向下钻取／全部展开	DrillDownFilter 中的 drillDown(DrillDownEvent)	DrillDownEvent 方法
深入钻取	DrillThroughFilter 中的 drillThrough(DrillThroughEvent)	DrillThroughEvent 方法
向上钻取	DrillUpFilter 中的 drillUp(DrillUpEvent)	DrillUpEvent 方法

（当用户执行操作时）要捕获的事件	要实现的接口	可用的事件方法
展开	ExpandFilter 中的 `expand(ExpandEvent)`	ExpandEvent 方法
仅隐藏	HideOnlyFilter 中的 `hideOnly(HideOnlyEvent)`	HideOnlyEvent 方法
仅保存	KeepOnlyFilter 中的 `keepOnly(KeepOnlyEvent)`	KeepOnlyEvent 方法
选择成员（例如，在成员过滤器中）	`MemberSelectFilter` 中的 `memberSelect(MemberSelectEvent)`	MemberSelectEvent 方法
旋转	PivotFilter 中的 `pivot(PivotEvent)`	PivotEvent 方法
数据查询	QueryFilter 中的 `query(QueryEvent)`	QueryEvent 方法
仅除去	`RemoveOnlyEvent` 中的 `removeOnly(RemoveOnlyEvent)`	RemoveOnlyEvent 方法
全部显示	ShowAllFilter 中的 `showAll(ShowAllEvent)`	ShowAllEvent 方法
仅显示	ShowOnlyFilter 中的 `showOnly(ShowOnlyEvent)`	ShowOnlyEvent 方法
交换轴	SwapAxisFilter 中的 `swapAxis(SwapAxisEvent)`	SwapAxisEvent 方法

用于实现事件侦听器对象的方法

要创建事件侦听器，您必须编写用于实现下面列示的一个或多个事件侦听器方法的类。下表列示要捕获的事件、为了捕获该事件要实现的方法以及该过滤器事件的支持方法。

（当用户执行操作时）要捕获的事件	要实现的接口	可用的事件方法
书签：删除	`BoomarkDeleteListener` 中的 `bookmarkDelete(BookmarkDeleteEvent)`	BookmarkDeleteEvent 方法
书签：装入	`BoomarkLoadListener` 中的 `bookmarkLoad(BookmarkLoadEvent)`	BookmarkLoadEvent 方法
书签：重命名	`BoomarkRenameListener` 中的 `bookmarkRename(BookmarkRenameEvent)`	BookmarkRenameEvent 方法
书签：保存	`BoomarkSaveListener` 中的 `bookmarkSave(BookmarkSaveEvent)`	BookmarkSaveEvent 方法
ChartBlox 中的过滤器更改	`ChartPageListener` 中的 `changePage(ChartPageEvent)`	ChartPageEvent 方法
折叠	CollapseListener 中的 `collapse(CollapseEvent)`	CollapseEvent 方法
数据排序	DataSortListener 中的 `dataSort(DataSortEvent)`	DataSortEvent 方法
向下钻取 / 全部展开	`DrillDownListener` 中的 `drillDown(DrillDownEvent)`	DrillDownEvent 方法
深入钻取	`DrillThroughEvent` 中的 `drillThrough(DrillThroughEvent)`	DrillThroughEvent 方法
向上钻取	DrillUpListener 中的 `drillUp(DrillUpEvent)`	DrillUpEvent 方法
展开	ExpandListener 中的 `expand(ExpandEvent)`	ExpandEvent 方法
仅隐藏	HideOnlyListener 中的 `hideOnly(HideOnlyEvent)`	HideOnlyEvent 方法
仅保存	KeepOnlyListener 中的 `keepOnly(KeepOnlyEvent)`	KeepOnlyEvent 方法
选择成员（例如，在成员过滤器中）	`MemberSelectListener` 中的 `memberSelect(MemberSelectEvent)`	MemberSelectEvent 方法

（当用户执行操作时）要捕获的事件	要实现的接口	可用的事件方法
以 PDF 格式导出数据	PdfListener 中的 `pdf(PdfEvent)`	PdfEvent 方法
旋转	PivotListener 中的 `pivot(PivotEvent)`	PivotEvent 方法
数据查询	QueryListener 中的 `query(QueryEvent)`	QueryEvent 方法
仅除去	RemoveOnlyListener 中的 `removeOnly(RemoveOnlyEvent)`	RemoveOnlyEvent 方法
全部显示	ShowAllListener 中的 `showAll(ShowAllEvent)`	ShowAllEvent 方法
仅显示	ShowOnlyListener 中的 `showOnly(ShowOnlyEvent)`	ShownOnlyEvent 方法
交换轴	SwapAxisListener 中的 `swapAxis(SwapAxisEvent)`	SwapAxisEvent 方法

第 13 章 连接至数据

在可以通过 DB2 Alphablox 应用程序做一些有用的事情之前，需要执行的第一项任务是连接至数据源。在本节中，您将了解更多关于创建数据源、连接至数据源以及如何管理数据源访问的信息。

创建数据源

在分析应用程序可以做任何有用的事情之前，它们需要访问可以由用户查看和分析的数据。在您一开始需要执行的任务中，其中一项任务是在 DB2 Alphablox 管理页面中创建数据源定义。这些数据源定义指向将要连接的关系数据库或多维数据库，并允许您快速地连接至那些数据库以及从中检索结果集。

创建数据源定义的操作在更大程度上是一项管理任务，但它可以由服务器管理员或开发者完成，只要他们具有管理权限即可。以下是开发者或管理员在定义 DB2 Alphablox 数据源时必须执行的任务的简要描述。

注: 《开发者指南》和“Blox 样本程序”应用程序中使用的所有示例都使用 QCC 数据库，该数据库或者是 QCC-Essbase（对于 DB2 OLAP Server 和 Essbase 而言），或者是 QCC-MSAS（对于 Microsoft Analysis Services 而言）。要安装和配置 QCC，请参阅 DB2 Alphablox CD 上样本数据目录中的 `readme.txt` 文件:

```
<cdromDir>/sampledata/qcc/
```

定义数据源

定义数据源的操作包括下列步骤:

1. 使用“开始”菜单或者通过在 Web 浏览器中输入以下 URL 来访问 DB2 Alphablox 主页:

   ```
   http://<serverName>/AlphabloxAdmin/home/
   ```

2. 使用具有管理员权限的用户名和密码进行登录。应该会显示具有三个选项卡的 DB2 管理页面，缺省情况下，显示的是“应用程序”页面。
3. 单击“管理”选项卡。然后，单击“数据源”以查看可用数据源定义列表。
4. 要定义数据源，请单击现有数据源定义列表下面的“创建”按钮。（如果应用程序所需的数据源定义已存在，则可以跳过这些步骤的余下部分。）
5. 填写“创建数据源”面板上的条目。要获取帮助，请单击此页面上的“帮助”按钮。
6. 单击“保存”以保存新定义。新定义的数据源名称应该会显示在可用数据源定义列表中。

要获取关于数据源的完整描述以及了解在为受支持的多维数据库和关系数据库定义数据源时需要执行的步骤的更多详细信息，请参阅《*管理员指南*》的『定义新数据源』一节。

定义 DataBlox dataSourceName 属性

DataBlox（无论是用作独立的 Blox 还是用作嵌套的 Blox）用来管理表示 Blox 与适当数据源之间的连接。DataBlox 还负责提交查询和从数据源检索结果集。在 DB2 Alphablox 管理页面中定义了数据源之后，您需要告诉 DataBlox 在什么位置获取用来访问适当数据库的信息。要让 DataBlox 指向数据源，请使用 DataBlox 的 dataSourceName 属性。

可以采用两种技术来让 DataBlox 指向数据源:

- 设置 DataBlox 的 dataSourceName 属性
- 通过使用服务器端 Java 方法或 JavaScript 来调用服务器端方法（使用 DHTML 客户机 API），设置 DataBlox 的 setDataSourceName 方法。

设置 dataSourceName 属性

最常用的数据源定义技术是添加 dataSourceName 作为属性并设置它的值。该值应该是您在 DB2 Alphablox 管理页面中定义的其中一个数据源的名称。

例如，在以下代码示例中，嵌套的 DataBlox 将数据源设置为 QCC-Essbase:

```
<blox:present id="myPresent" ...>
...
<blox:data
dataSourceName="QCC-Essbase"
query='<SYM <ROW("All Products")
<COLUMN ("All Time Periods") "2000" Sales !' />
</blox:present>
```

注: 如果您忘记将 dataSourceName 属性添加到 DataBlox 中，数据表示 Blox 就会显示没有数据可用消息。或者，如果该数据源尚未定义，则 JSP 页面就无法正确地被编译，从而导致生成异常。

使用 setDataSourceName() JavaScript 方法

有时候，您可能想使用 JavaScript 或 Java 来通过程序更改数据源，例如，当用户单击按钮时执行此操作。以下示例显示了一个使用 Blox JavaScript setDataSourceName 方法的示例:

使用 DataSourceSelectFormBlox 来设置不同的数据源

执行下列步骤来创建一个带有 DB2 OLAP Server 数据源和 Essbase 数据源选择列表的 JSP 页面。当数据源被选择时，将执行一个缺省查询，该查询将可用的维装入到 DataLayout 面板中，从而允许用户执行特别分析。可以在“Blox 样本程序”的“使用 FormBlox”部分中的“特别分析”示例中找到此示例的完整版本。以下示例使用 DB2 OLAP Server 和 Essbase 版本，但 Microsoft Analysis Services 版本的工作方式是类似的。

1. 在页面的顶部，添加一条 JSP page 伪指令来指定需要提供的 Java 类:

   ```
   <%@ page import="com.alphablox.blox.form.FormEventListener,
                    com.alphablox.blox.DataBlox,
   com.alphablox.blox.form.FormEvent" %>
   ```

2. 在 page 伪指令下面，为将要在此页面上使用的 Blox 标记库（在本实例中，是标准的 Blox 标记库和 Blox 表单标记库）添加 taglib 伪指令:

```
<%@ taglib uri="bloxtld" prefix="blox" %>
<%@ taglib uri="bloxformtld" prefix="bloxform" %>
```

3. 指定将要使用的 DataBlox，从而启用别名成员名（仅限于 DB2 OLAP Server 和 Essbase）并告诉 DataBlox 在启动时不要连接至数据源:

```
<blox:data id="AdHocDataBlox"
connectOnStartup="false"
useAliases="true" />
```

4. 指定 PresentBlox:

```
<blox:present id="AdHocPresentBlox"
visible="false"
width="600"
height="350">
<blox:grid noDataMessage="Select a data source" />
<blox:chart noDataMessage="Select a data source" />
<blox:data bloxRef="AdHocDataBlox" />
</blox:present>
```

我们将公共的 Blox noDataMessage 值设置为“请选择数据源”，这条消息要比缺省的没有数据可用消息好。并且，嵌套的 DataBlox 标记声称使用先前定义的 DataBlox。

5. 现在，我们添加一个 DataSourceSelectFormBlox，它将自动生成可用的 DB2 OLAP Server 数据源和 Essbase 数据源的列表:

```
<bloxform:dataSourceSelect id="dataSourceSelector"
type="MDB"
adapter="IBM DB2 for OLAP"
visible="false"
nullDataSourceLabel="Select the data source">
<%
  dataSourceSelector.addFormEventListener(new
     DataSourceFormBloxEventListener(AdHocDataBlox));
%>
</bloxform:dataSourceSelect>
```

type 属性指示我们仅指定多维数据源，adapter 属性设置将数据源限制为仅包括 DB2 OLAP Server 或 Essbase。并且，不是在页面装入时指定数据源，而是添加 nullDataSourceLabel 选项来通知用户“请选择数据源”。

此外，嵌套的 Java scriptlet 告诉 DataSourceSelectFormBlox 它需要添加页面底部包括的 FormEventListener。此事件侦听器将允许我们在不指定数据源的情况下创建 DataBlox，数据源留待用户选择。

6. 使用下列 <blox:display> 标记来对页面进行布局并指定 DataSourceSelectFormBlox 和 PresentBlox 的显示位置:

```
<blox:display bloxRef="dataSourceSelector"/>
<blox:display bloxRef="AdHocPresentBlox" />
```

请参阅“Blox 样本程序”示例以获取完整的页面布局代码。

7. 最后，添加 FormBloxEventListener 类:

```
<%!
  public class DataSourceFormBloxEventListener
     implements FormEventListener {
    private DataBlox dataBlox;

public DataSourceFormBloxEventListener(DataBlox dataBlox) {
      this.dataBlox = dataBlox;
}
```

```
public void valueChanged(FormEvent event) throws Exception {
      String dataSourceName = event.getFormBlox().getFormValue();
      dataBlox.setDataSourceName(dataSourceName);
      if (dataSourceName != null) {
dataBlox.setQuery("!");
        dataBlox.updateResultSet();
}
  else
        dataBlox.disconnect(true);
}
}
%>
```

这个 DataSourceFormBloxEventListener 类将获取数据源的 FormBlox 值并设置缺省查询，后者将使用所有可用的维来填充 PresentBlox 的 DataLayout 面板。

注: 要了解关于 FormBlox 或 DataBlox 属性和方法的语法及用法详细信息，请参阅 *Developer's Reference* 一书。

连接至数据源或断开与数据源的连接

在将独立的或嵌套的 DataBlox 实例化时，将调用隐式的 connect 方法。如果在 DataBlox 中指定了查询，则将执行该查询并生成结果集。如果将 DataBlox 的 connectOnStartup 属性设置为 false（缺省值为 true），则不会调用 connect 方法，您以后将必须通过程序来进行连接。

在 Blox 与它的数据源建立连接之后，该连接在当前会话中将一直存在。这种缺省行为能够提高性能，它使应用程序不必重复地为每个查询（包括初始查询以及由用户与 Blox 之间的交互产生的查询）打开和关闭数据库连接。

根据任务的不同，可以使用许多不同的选项来通过 DataBlox 属性和方法管理数据源连接，概述如下:

目标	DataBlox 属性和方法	结果
连接至数据源，但不执行查询	<blox:data ... connectOnStartup="true" ... </blox:data> [注意: 未设置 query 属性]	• 未检索任何结果集 • 用户看到公共 Blox noDataMessage 属性的消息（缺省消息为: "没有数据可用"），可以定制该消息
	connect(false);	• 如果连接已存在，则断开连接，然后重新连接 • 建立了连接 • 未执行已定义的查询 • 用户看到公共 Blox noDataMessage 属性的消息（缺省消息为: "没有数据可用"），可以定制该消息
连接至数据源并执行查询	connect(); 或 connect(true); [注意: 假定已设置了查询]	• 执行了已定义的文本查询 • 检索了结果集
	setQuery(); updateResultSet();	• 设置了查询，建立了连接，并且检索了结果集

目标	DataBlox 属性和方法	结果

根据连接属性更改来更新结果集	// Change properties first
	
	updateResultSet();
		• 在应用结果集属性更改之后（例如，在将 useAliases 设置为 true 或 false 之后），更新结果集
		
		[注意：如果正在应用连接属性（如 dataSourceName、username、schema 和 password），则改为使用 connect 方法。]

注: 要了解关于这些 DataBlox 属性和方法的语法及用法详细信息，请参阅 *Developer's Reference* 一书。

下面是一个先设置查询并接着进行连接的 Java scriptlet 示例:

```
<%
String query = "<ROW (\"All Products\") <ICHILD \"All Products\" "+
"<COLUMN (\"All Time Periods\") <CHILD \"All Time Periods\" "+
(Measures) Sales !";
 PresentBlox3.getDataBlox().setQuery(query);
 PresentBlox3.getDataBlox().connect();
%>
```

注: "Blox 样本程序"的"检索数据"部分中的"使用 JSP Scriptlet 的初始查询"示例演示了此技术。

有时，您可能希望通过程序来控制 Blox 连接数据源以及与数据源断开连接的时间。例如，可以设计一个页面来让用户使用选择列表、单选按钮和复选框来进行许多选择，然后，他们可以通过单击一个按钮来提交查看请求。可以通过许多种方法来实现此目的，包括装入缺省视图并将 HTML 表单元素或 FormBlox 预设置为具有缺省值，或者装入不带视图的 Blox 并等待用户进行选择。要获取如何实现此目标的示例，请参阅下面『自动连接和自动断开连接』 。

自动连接和自动断开连接

如下所述，在某些情况下可以对关系数据源和多维数据源使用 DataBlox 的 autoConnect 和 autoDisconnect 属性以便更好地管理 DB2 Alphablox 分析应用程序的性能和可伸缩性。

要了解有关 autoConnect 和 autoDisconnect 属性的语法和用法详细信息，请参阅 *Developer's Reference* 一书的 DataBlox 一节。

关系数据源

当可用端口数目有限并且您正在使用关系数据源时，可以对 DataBlox 设置 autoConnect 和 autoDisconnect 属性以便对应用程序连接的使用进行管理。下表概述了 autoConnect 与 autoDisconnect 属性的所有可能设置组合以及产生的行为:

autoConnect	autoDisconnect	行为
false	false	这些是 DataBlox 的缺省设置。将使用隐式的 connect 方法来对已定义的数据源执行已定义的查询。一旦建立了连接，就将在当前浏览会话期间保持该连接。
true	true	仅当可用的数据库端口有限时才建议使用此组合。将建立初始数据库连接并执行查询，接着返回并显示结果集，然后自动地与数据库断开连接。记住，用户与 Blox 之间的许多交互都将重复此循环并要求重新建立连接。
true	false	此组合与缺省行为实际上没有区别。
false	true	在显示初始结果集之后，用户将无法对该结果集执行任何操作。尽管这种设置组合是有可能的，但通常建议不要使用它。

多维数据源

DataBlox 的 autoConnect 属性对多维数据库不起作用，但是可以对 Microsoft Analysis Services 数据源使用 autoDisconnect 属性来管理分析应用程序的可伸缩性和性能。

（仅限于 Microsoft Analysis Services 数据源）将 autoDisconnect 属性设置为 true 将导致数据源连接在进行查询执行操作（这包括执行查询、进行下寻和上寻、进行旋转以及使用“仅保留”和“仅除去”）之后立即断开。元数据调用不受影响。在每次断开连接之后，将从 java.exe 进程中清除 PivotTable Services 高速缓存内存，并且 DataBlox 将使用先前连接信息立即重新连接。

仅当您遇到由于 PivotTable Services 高速缓存内存消耗过度而导致的可伸缩性问题时，才应该考虑对 Microsoft Analysis Service 使用 autoDisconnect 属性。保持的每个 MSAS 连接都会消耗多达大约 250 MB 的内存，从而快速消耗可用的服务器内存资源。通过将 autoDisconnect 设置为 true，可以防止 PivotTable Services 内存消耗。当 autoDisconnect 设置为 false（缺省值）时，PivotTable Services 高速缓存将得到维护，并且向用户显示频繁访问的数据的速度会较快。

Essbase 和 DB2 OLAP Server 的单点登录

DB2 Alphablox 应用程序支持使用 Essbase 单点登录凭证对象来访问 Hyperion Essbase 和 DB2 OLAP Server 数据源。此功能允许 Essbase 用户登录一次并使用生成的凭证在 DB2 OLAP Server 和 Essbase 数据源之间移动。

单点登录允许用户在仅登录一次之后连接至多个 DB2 OLAP Server 和 Hyperion Essbase 数据源。当在受支持的数据源上认证用户时，将生成一个已加密的记号，它包含一些用户凭证（包括用户名，根据配置的不同，还有用户密码），可以在多个 Essbase 数据源之间传递该记号。可以创建 DB2 Alphablox 应用程序来使用由 Hyperion Common Security Services 创建的凭证对象并通过 DataBlox 传递此用户信息。当将 Essbase 单点登录与 DB2 Alphablox 应用程序配合使用时，用户名和密码不必存储在 DB2 Alphablox 存储库中。

使用 DataBlox credential 属性来传递用户凭证

可以使用 DataBlox credential 属性来将 DB2 OLAP Server 和 Hyperion Essbase 用户凭证传递到受支持的数据源。

以下步骤假定您能够访问支持单点登录的 DB2 OLAP Server

或 Hyperion Essbase 数据源。

要传递凭证对象:

1. 添加 JSP scriptlet 以指定包含用户凭证记号的变量。

   重要: 此 scriptlet 必须出现在将接受用户凭证的数据源的 DataBlox 标记中。
2. 在 DataBlox 标记中，添加 credential 属性并将值设置为 JSP 表达式，该表达式用于检索所指定变量的值。

在以下示例中，使用 JSP page 伪指令来获得必需的 Java

包，然后 JSP scriptlet 指定用来检索用户凭证记号的变量（userCredential）。接着，DataBlox 的 credential 属性检索此信息以访问指定的 Essbase 数据源。

```
<%@ page import="com.hyperion.css.CSSAPIIF" %>
<%@ page import="com.hyperion.css.CSSException" %>
<%@ page import="com.hyperion.css.CSSSystem" %>
<%@ page import="com.hyperion.css.application.CSSApplicationIF" %>
<%@ page import="com.hyperion.css.common.CSSUserIF" %>
<%@ page import="java.io.*" %>
<%@ page import="java.net.*" %>
<%@ page import="java.util.*" %>
<%@ taglib uri="bloxtld" prefix="blox" %>
<%@ page contentType="text/html;charset=utf-8" %>
<%!
public class MyCssApp implements CSSApplicationIF {
//Implements your application contract - code omitted here.
}
%>
<html>
<head>
<blox:header/>
</head>
<body>
<%
String credential = request.getSession().getAttribute("SSOToken");
if (credential == null)
{
HashMap css_context = new HashMap();
String user = request.getParameter("username");
String password = request.getParameter("password");
MyCssApp myApp = new MyCssApp();
CSSSystem system = CSSSystem.getInstance();
CSSAPIIF css = system.getCSSAPI();
css_context.put(CSSAPIIF.LOGIN_NAME, user);
css_context.put(CSSAPIIF.PASSWORD, password);
css.initialize(css_context, myapp);
CSSUserIF css_user = css.authenticate(css_context);
credential = css_user.getToken();
request.getSession().setAttribute("SSOToken", user1.getToken());
}
%>
<blox:data id="myDataBlox"
credential="<%= credential %>"
dataSourceName="EssbaseSSO"
```

```
...
   </blox:data>
<blox:present id="myPresentBlox"
width="700" height="500"
<blox:data bloxRef="myData" />
</blox:present>
</body>
</html>
```

要了解有关 DataBlox credentials 属性的更多信息，请参阅 *Developer's Reference* 一书的 Data Reference 一节。

使用 Blox API 来传递用户凭证

通过使用 Blox API 来调用 DataBlox `setCredential()` 方法，可以将 Hyperion Essbase 用户凭证传递到 Essbase 数据源。

以下步骤假定您能够访问支持单点登录的 Hyperion Essbase 数据源。

要使用 Blox API 来传递凭证对象:

1. 添加 JSP scriptlet 以指定包含用户凭证记号的变量。

   重要: 此 scriptlet 必须出现在将接受用户凭证的数据源的 DataBlox 标记中。
2. 添加 DataBlox 标记，但不要添加 `credential` 属性。
3. 在 DataBlox 标记下添加 JSP scriptlet 以设置用户凭证。

在以下示例中，JSP scriptlet 指定用来检索用户凭证记号的变量（`usercredential`）。然后，DataBlox `setCredential()` 检索用户信息并将它应用到指定的 DataBlox。

```
<%
String userCredential = myGetCSSToken();
%>
...
<blox:data id="myDataBlox"
datasourceName="EssbaseSSO"
...
</blox:data>
...
<%
myDataBlox.setCredential(userCredential);
%>
```

使用 Blox API 来控制用户凭证的设置将允许您执行其他在您想要检索用户凭证之前可能需要执行的操作。有关 setCredential() 方法的更多信息，请参阅 *Blox API Javadoc* 文档。

单点登录的局限性

当将单点登录与 Hyperion Essbase 或 DB2 OLAP Server 数据源配合使用时，您应了解这些已知局限性和其他可能影响您的应用程序的问题。

对于使用 Hyperion Common Security Services 2.6 和 2.7（Hyperion Essbase 和 Hyperion Essbase Deployment Services 7.1.3、7.1.2 和 7.1.1）的数据源，DB2 Alphablox 中提供了单点登录支持 相应的 DB2 OLAP Server 8.2 版本也受 DB2 Alphablox 支持。

- 当您使用 LDAP 服务器进行外部认证时，在 Essbase 7.1.3 中工作的用户凭证记号在 Essbase 7.1.1 或 7.1.2 中将不会工作，反之亦然。相应的 DB2 OLAP Server 8.2 版本将具有相同的行为。

- Essbase 7.1.3 尝试使用用户标识（UID）来查找用户。如果记号是使用用户的 UID 生成的，则用户凭证记号将使用 Essbase 7.1.3 数据源，但该记号在应用于 Essbase 7.1.1 和 7.1.2 时将失败，同时显示“用户是未知的”消息。
- Essbase 7.1.1 和 7.1.2 尝试使用公共名（CN）来查找用户。如果记号是使用用户的 CN 生成的，则该记号在传递到 Essbase 7.1.1 和 7.1.2 时仍将起作用，但它在应用到 Essbase 7.1.3 时将失败，同时出现“用户是未知的”消息。

• 如果您将用户凭证记号应用到 DataBlox，则任何其他与 DataBlox 相关联的用户名或密码将被忽略。如果您已通过使用 DataBlox 标记属性或在 DB2 Alphablox 存储库中指定了用户名或密码，则在使用用户凭证记号时将忽略这些用户名或密码。

第 14 章 检索数据

在连接至数据源之后，下一个任务是从您提交的查询所生成的结果集中检索数据。有时，这些查询是由数据库管理员或数据分析员提供给您的。在更多的时候，您将自己编写查询语句或与别人协作编写查询语句。您对所访问的数据源越熟悉，您就越能够独立地工作。在本节中，您将只学习有关从各种数据源中检索数据以便在 DB2 Alphablox 应用程序中进行查看的基础知识。本节的目标是帮助您解决一些常见的问题。

根据所访问的数据源的不同，用来指定应用程序查询的语法也有很大的变化。在 DB2 Alphablox 应用程序中，查询字符串可以是下列其中一项:

- Essbase 报告脚本: 用于 IBM DB2 OLAP Server 和 Hyperion Essbase 数据源
- 多维表达式（MDX）: 用于 Microsoft SQL Server Analysis Services 和 DB2 Alphablox Cube Server
- SQL 语句: 用于关系数据源

如果您熟悉特定数据源，并且知道如何创建查询以检索数据，则您掌握的大部分知识在 DB2 Alphablox 应用程序中都正好适用。但是，当使用 DB2 Alphablox 时，您应该了解一些有用的技巧，因此，请您务必阅读本章中与您使用的数据源相关的小节。

如果您不熟悉特定的数据源，则下列小节应该能够帮助您获得有关使用数据源时可能会遇到的语法和 DB2 Alphablox 问题的简要概述。要了解有关使用数据源的详细信息，请参阅下面的相关小节以了解在哪里能够找到更多信息。

可以使用 Application Studio Workbench 中包括的查询构建器来输入查询以及针对分析应用程序将要使用的数据源测试那些查询。此工具连接至 DB2 Alphablox 中定义的任何数据源。可以通过几种方法来使用查询构建器开发查询:

- 输入文本字符串并查看得到的分析视图
- 对数据源调用最后一个查询并查看得到的分析视图
- 执行数据源的缺省查询（如果存在的话）并查看得到的分析视图
- 使用 GridBlox 用户界面来在各个轴之间移动维、旋转、钻取和过滤数据以及执行其他操作来得到应用程序所需的分析视图。然后，检索生成该视图所需的查询字符串。

在得到适当的查询字符串之后，可以将其剪切并粘贴到 DataBlox 的查询值中，也可以将其保存在文本文件中以供将来使用。要了解更多信息，请参阅第 129 页的『查询构建器』。

应用程序设计确定了指定应用程序查询的位置。DB2 Alphablox 应用程序可以根据下列各项来发出查询请求:

- Blox 实例化（通过 DataBlox query 属性）
- 用户从预定义查询列表中进行选择，这可能是通过 HTML 表单按钮进行的
- 与用户概要文件相关联的定制属性，该属性包含查询字符串

设置 DataBlox query 属性

DataBlox 的 query 属性确定一个初始查询，在 DataBlox 或嵌套的 DataBlox 装入后，应该对数据库执行这个初始查询。如果未定义任何查询，则缺省查询是空字符串。缺省情况下，在没有结果集的情况下装入的 Blox 将显示"没有数据可用"消息。

要为 Blox 定义初始查询，有两个选项:

- 在 DataBlox 的 query 属性值中定义初始查询，或者
- 使用 Java 方法来设置查询，然后执行该查询

要使用 DataBlox query 属性来定义查询字符串，只需要对 DataBlox 添加 query 属性。应该按照以下格式输入 DataBlox query 属性:

```
query="queryString"
```

其中，*queryString* 是一个字符串，它定义要对数据源执行的查询，该数据源是使用 dataSourceName 属性定义的。

在以下示例中，GridBlox 的嵌套的 DataBlox 将对 QCC-Essbase 数据源执行已定义的 *queryString*:

```
<blox:grid id="myGrid">
<blox:data
dataSourceName="QCC-Essbase"
query='<ROW("All Products") <CHILD "All Products"
<COLUMN("All Time Periods") <CHILD "2000" Sales !'/>
</blox:grid>
```

您可能会发现，在 Java 函数中定义 query 属性对于提高可读性以及简化编码来说十分有益。以下两个任务显示了如何做到这一点。缺省情况下，当使用 query 属性时，Blox 将负责连接至已定义的数据源以及执行查询。当使用方法时，需要使用 DataBlox connect() 方法来执行已定义的查询。以下任务显示了使用 Java 方法来从已定义的数据源检索结果集的示例。

使用 JSP 查询来设置和执行查询

当存在性能问题时，您可能想通过这个简单的窍门来利用 JSP scriptlet 加快所显示的视图的响应速度，在这个 JSP scriptlet 中，使用了两个服务器端方法来生成结果集。可以使用 DataBlox setQuery() 方法来定义初始查询，于是，connect() 方法将导致执行该查询并将结果集返回给外围 Blox。

1. 在 JSP 页面顶部，但在 taglib 伪指令后面，添加适当的 Blox 标记以定义表示 Blox，但将 visible 属性设置为 false 以使 Blox 在数据可用之前不显示。包括一个嵌套的 DataBlox 并设置 dataSourceName 属性以定义数据源，但不要包括 query 属性。

   例如，以下 Blox 标记定义了一个 PresentBlox，并且将 visible 设置为 false，将 dataSourceName 设置为 QCC:

   ```
   <blox:present id="PresentBlox3"
   visible="false"
   width="550"
   height="350">
   <blox:data
   dataSourceName="QCC"/>
     <blox:grid
   ```

```
bandingEnabled="true"/>
<blox:chart
chartType="Bar"/>
</blox:present>
```

2. 在定义表示 Blox 的 Blox 标记下面，添加一个 JSP scriptlet，该 scriptlet 执行三个子步骤:

   - 声明查询变量
   - 设置该查询变量中定义的查询
   - 将该 Blox 连接至数据源，从而导致返回结果集

   以下 scriptlet 示例显示了使用 Java 方法来定义和执行查询:

   ```
   <%
   String query="<ROW(\"All Products\") <ICHILD \"All Products\""+
   "<COLUMN(\"All Time Periods\") <CHILD 2000 "+
   "<PAGE(Measures) Sales !";
    PresentBlox3.getDataBlox().setQuery(query);
    PresentBlox3.getDataBlox().connect();
   %>
   ```

   在此示例中，（通过在变量声明前面指定 `String`）将查询变量声明为字符串，然后将该变量设置为等于所需的查询语句（本示例中的查询是一个 Essbase 报告脚本）。注意，为了便于您阅读和维护，查询进行了折行处理并使用加号连接在一起。在声明查询之后，使用了两个 DataBlox 方法，即 `setQuery()` 和 `connect()`。`setQuery()` 方法在 DataBlox 中设置查询（注意，在本示例中，可以通过指定 `query` 作为自变量来在方法自变量中替换已定义的查询）。然后，通过使用 `connect()` 方法，指示 DataBlox 连接至数据源并执行已设置的查询。

3. 往下，在 JSP 页面的 `<body>` 中，在要显示 Blox 以供查看的位置指定 Blox display 标记。使用 `bloxRef` 属性来引用该表示 Blox（将它的值设置为所显示的 Blox 的名称）。

   对于这里的示例，在 `<body>` 标记中，应该指定以下标记:

   ```
   <blox:display bloxRef="PresentBlox3"/>
   ```

正如您在本示例中看到的那样，即使您了解的 Java 知识有限，也可以使用这种技术。

多维数据源

*《管理员指南》*一书提供了有关多维数据库的概述。要了解有关下列各项的更多信息，请参阅下列各节:

- OLAP 术语与概念
- 多维分析
- OLAP 数据库术语

IBM DB2 OLAP Server 和 Hyperion Essbase

IBM DB2 OLAP Server 和 Hyperion Essbase 是针对分析进行优化的多维数据库，它们通常能够生成亚秒级的查询响应。

要从 DB2 OLAP Server 或 Essbase 立方体中检索数据，您需要使用 Essbase 报告规范语言来生成报告脚本，后者可用作 DB2 Alphablox 应用程序中的查询值。

以下内容提供了有关如何创建 Essbase 报告脚本的基本概述，并提供了有关如何将报告脚本与 DB2 Alphablox 功能配合使用的重要技巧。

要了解关于使用 DB2 OLAP Server 或者 Essbase 和 Essbase 报告脚本的详细信息，请参阅 DB2 OLAP Server 或 Essbase 文档。

创建 Essbase 报告脚本

要将查询传递至 IBM DB2 OLAP Server 或 Hyperion Essbase 数据源，请使用 Essbase 报告规范语言来创建报告脚本。

技巧: 要了解有关报告脚本规范语言的信息，请参阅 DB2 OLAP Server 或 Essbase 安装目录中的联机文档（/docs/techref/RPTIND.HTM）。如果工作站上安装了 DB2 OLAP Server 或 Essbase Application Manager，则可以通过“帮助”菜单访问此文档。

以下有关 DataBlox 的示例指定:

- 将 Market 维和 Accounts 维显示在列轴上。
- 将 Scenario 维和 Product 维显示在行轴上。
- 应该将全部四个维的子代都包括在结果集中。
- 任何未使用的维都应该显示在“其他”轴上。

```
<blox:data ...
query="<SYM <ROW (Scenario,Product)
<ICHILD Scenario <ICHILD Product <COLUMN (Market, Accounts)
<ICHILD Market <ICHILD Accounts !"/>
```

DB2 Alphablox 支持的 Essbase 报告脚本命令

下表列示了大部分 Essbase 报告脚本命令、它们是否受 DB2 Alphablox 支持（即，当在报告脚本中输入它们时，它们是否能够起作用）、等效的或接近等效的 DB2 Alphablox 功能以及使用这些命令的报告脚本示例。

报告脚本命令	报告脚本示例及注释
!	对于执行报告脚本查询来说，此命令是必需的。单独的“惊叹号”查询在网格或图表上返回一个维，并在 DataLayout 面板中返回所有可用维的列表。DB2 Alphablox 不支持多惊叹号报告输出命令。请参阅本表后面的注解。
&1	&CurrentMonth 如果在 IBM DB2 OLAP Server 或 Hyperion Essbase 中定义此命令，就可以将服务器替换变量添加到报告脚本中。这些服务器替换变量主要用来简化脚本维护工作。
ALLINSAMEDIM	<ROW (Scenario) <ALLINSAMEDIM Actual !
ALLSIBLINGS	<ROW (Scenario) <ALLSIBLINGS Actual !
ANCESTORS	<ROW (Measures) <ANCESTORS "Marketing" !
ASYM	<ASYM <COL (Scenario, Year) Actual Jan Budget Feb !
ATTRIBUTE	<ROW (Product) <ATTRIBUTE Bottle !
BOTTOM	<ROW (Year) <DIMBOTTOM Year <BOTTOM (6, @DataCol(1)) !
CALCULATECOLUMN	请参阅 DB2 OLAP Server 或 Essbase 文档以获取示例。

报告脚本命令	报告脚本示例及注释
CALCULATEROW	请参阅 DB2 OLAP Server 或 Essbase 文档以获取示例。
CHILDREN	<ROW (Market) <CHILDREN Market !
CLEARALLROWCALC	请参阅 DB2 OLAP Server 或 Essbase 文档以获取示例。
CLEARROWCALC	请参阅 DB2 OLAP Server 或 Essbase 文档以获取示例。
COLUMN	<COLUMN (Year) <CHILD Year !
DESCENDANTS	<ROW (Year) <DESCENDANTS Year !
DIMBOTTOM	<ROW (Year) <DIMBOTTOM Year !
DUPLICATE	<ROW (Year) <Child Year <DUPLICATE Qtr1 !
FIXCOLUMNS	<COL (Year) {FIXCOLUMNS 3} <DIMBOTTOM Year !
GEN	<ROW (Product) gen2,Product !
IANCESTORS	<ROW (Year) <IANCESTORS Jan !
ICHILDREN	<ROW (Product) <ICHILDREN Colas !
IDESCENDANTS	<ROW (Product) <IDESCENDANTS Product !
INCMISSINGROWS	{SUPMISSINGROWS} {INCMISSINGROWS} <PAGE (Market) "New York" <ROW (Product) lev0,Product !
INCZEROROWS	{SUPZEROROWS} {INCZEROROWS} <PAGE (Market) "New York" <ROW (Product) lev0,Product !
IPARENT	<ROW (Year) <IPARENT Jan !
LATEST	<LATEST Aug <ROW (Year) <CHILD QTR3 Q-T-D !
LEV	<ROW (Product) lev0,Product !
LINK	<ROW (Year) <LINK(<DIMBOTTOM(Year) AND <DESCENDANTS(Qtr1)) !
MATCH	<ROW (Market) <MATCH (Market, C*) !
NAMESON	{SUPNAMES} {NAMESON} <ROW (Market) <CHILD East !
NOROWREPEAT	{NOROWREPEAT} <ROW (Market, Product) <CHILD East <CHILD Product !
OFFCOLVCALCS	请参阅 DB2 OLAP Server 或 Essbase 文档以获取示例。
OFFROWCALCS	请参阅 DB2 OLAP Server 或 Essbase 文档以获取示例。
OFSAMEGEN	<ROW (Market) <OFSAMEGEN East !
ONSAMELEVELAS	<ROW (Market) <ONSAMELEVELAS East !
ORDER	{ORDER 0 5 4 3 2 1} <COL (Product) <CHILD Product !
ORDERBY	<ROW (Product) <DIMBOTTOM Product <ORDERBY ("Product", @DATACOL(1) ASC) !
PAGE	<PAGE (Market) East <ROW (Product) <CHILD Product !
PARENT	<ROW (Year) <PARENT Jan !
REMOVECOLCALCS	请参阅 DB2 OLAP Server 或 Essbase 文档以获取示例。
RESTRICT	<ROW (Product) <DIMBOTTOM Product <RESTRICT (@DATACOL(1) > 10000) !
ROW	<ROW (Year) <PARENT Jan !
SCALE	{SCALE 100} <ROW (Product) <CHILD Product !
SETROWOP	请参阅 DB2 OLAP Server 或 Essbase 文档以获取示例。

报告脚本命令	报告脚本示例及注释
SINGLECOLUMN	<SINGLECOLUMN <COL (Year) Year <ROW (Product) <CHILD Product !
SORTALTNAMES	<ROW (Product) <SORTALTNAMES <DIMBOTTOM Product !
SORTASC	<ROW (Market) <SORTASC <DIMBOTTOM Market !
SORTDESC	<ROW (Market) <SORTDESC <DIMBOTTOM Market !
SORTGEN	<ROW (Product) <SORTGEN <DESCENDANTS Product !
SORTLEVEL	<ROW (Product) <SORTLEV <DESCENDANTS Product !
SORTMBRNAMES	<ROW (Product) <SORTMBRNAMES <SORTDESC <DIMBOTTOM Product !
SORTNONE	<ROW (Product) <SORTMBRNAMES <SORTDESC <SORTNONE <DIMBOTTOM Product !
SPARSE	<SPARSE <ROW (Product, Market) <DIMBOTTOM Product <DIMBOTTOM Market !
SUPEMPTYROWS	{SUPEMPTYROWS} <PAGE (Market) "New York" <ROW (Product) lev0,Product !
SUPMISSINGROWS	{SUPMISSINGROWS} <PAGE (Market) "New York" <ROW (Product) lev0,Product !
SUPSHARE	<SUPSHARE <ROW (Product) lev0,Product !
SUPSHAREOFF	<SUPSHARE <SUPSHAREOFF <ROW (Product) lev0,Product !
SUPZEROROWS	{SUPZEROROWS} <COL (Measures) Sales <ROW (Year) Jan Feb Mar !
SYM	<SYM <COL (Measures, Year) Sales COGS Jan Feb !
TOP	<ROW (Market) <DIMBOTTOM Market <TOP(5, @DATACOL(1)) !
UDA	<ROW (Market) <UDA (Market, "Major Market") !
WITHATTR	<ROW (Product) <WITHATTR(Caffeinated,"<>",True) !

注:

1. 可以使用 DB2 Alphablox 定制属性。
2. DB2 Alphablox 使用 selectableSlicerDimensions 来控制页面显示，但是 <PAGE 命令在报告脚本中用来将数据切片。
3. 在 DB2Alphablox 中，可以使用 suppressMissingOnRows、suppressMissingOnColumns 和 suppressZeros 来实现类似的效果。
4. 在 DB2 Alphablox 中，suppressMissingOnRows 和 suppressMissingOnColumns 消除行和列中缺少的值。
5. 可以在 DB2 Alphablox 中使用 suppressZeros，但是，此属性同时消除行和列中的零。

注: 在 DB2 Alphablox 中，不支持多惊叹号查询（包括多惊叹号（!）报告输出命令）。您可能会发现有几个利用多惊叹号报告输出命令的选择报告脚本在 Blox 中可能会显示结果，但是，使用它们的风险由您自行承担。

具有 DB2 Alphablox 等效命令的不受支持的报告脚本命令

以下 Essbase 报告脚本命令在 DB2 Alphablox 中不受支持。使用列示的 DB2 Alphablox 等效命令代替列示的 Essbase 报告脚本命令。

报告脚本命令
DB2 Alphablox 等效命令

AFTER defaultCellFormat（GridBlox）

BEFORE defaultCellFormat（GridBlox）

COMMAS defaultCellFormat（GridBlox）

DECIMAL
defaultCellFormat（GridBlox）

EUROPEAN
defaultCellFormat（GridBlox）

MISSINGTEXT
missingValueString（GridBlox）

NOINDENTGEN
rowIndentation（GridBlox）

OUTALT useAliases（DataBlox）

OUTALTNAMES
useAliases（DataBlox）

OUTALTSELECT
aliasTable（DataBlox）

OUTMBRNAMES
useAliases（DataBlox）

SUPBRACKETS
defaultCellFormat（GridBlox）

SUPCOMMAS
defaultCellFormat（GridBlox）

不具有 DB2 Alphablox 等效命令的不受支持的报告脚本命令

BLOCKHEADERS	BRACKETS	COLHEADING	CURHEADING	SAVEANDOUTPUT	SAVEROW	SETCENTER	SETROWOP
CURRENCY	DIMEND	DIMTOP	DIMBOTTOM	ENDHEADING	SKIP	SKIPONDIMENSION	STARTHEADING	SUPALL
FEEDON	FORMATCOLUMNS	HEADING	IMMHEADING	SUPCOLHEADING	SUPCURRHEADING	SUPEUROPEAN		
INCEMPTYROWS	INCFORMATS	INCMASK	INDENT	SUPFEED	SUPFORMATS	SUPHEADING	SUPMASK	
INDENTGEN	LMARGIN	MASK	NAMESCOL	NAMEWIDTH	SUPNAMES	SUPOUTPUT	SUPPAGEHEADING	
NEWPAGE	NOPAGEONDIMENSION	NOSKIPONDIMENSION	TABDELIMIT	TEXT	TODATE	UCHARACTERS	UCOLUMNS	
PAGEHEADING	PAGELENGTH	PAGEONDIMENSIONS	UDATA	UNAME	UNAMEONDIMENSION	UNDERLINECHAR		
PRINTROW			UNDERSCORECHAR	WIDTH	ZEROTEXT			

计算脚本

计算脚本是包含指令的文本文件，用于计算 DB2 OLAP Server 或 Essbase 立方体中的数据。在 DB2 Alphablox 应用程序中，可以使用下列 DataBlox 方法来调用计算脚本:

- executeCustomCalc
- executeNamedDBCalcScript
- substituteCalcScriptTokens
- writeback

要了解有关使用这些方法的详细信息，请参阅 *Developer's Reference* 一书。有关在 DB2 OLAP Server 或 Essbase 立方体中使用计算脚本的更多信息，请参阅 DB2 OLAP Server 或 Hyperion Essbase 文档。

替换变量

在 IBM DB2 OLAP Server 或 Hyperion Essbase 立方体中，替换变量充当定期更改的信息的全局占位符。每个变量都具有它被赋予的值，并且随时都可以被数据库管理员更改。使用替换变量有助于减少报告脚本维护工作量，从而使您不需要在 DB2 Alphablox 应用程序中对各个报告脚本进行手工更改。

例如，许多报告脚本都引用报告时间段，如当前月或当前季度。通过使用在 IBM DB2 OLAP Server 或 Hyperion Essbase 服务器上设置的替换变量，如 CurrentMonth 或 CurrentQuarter，可以在一个位置更改赋予的值，适当的报告脚本在报告脚本执行时将动态地更新。

要在报告脚本中引用替换变量，请在变量名前面指定 & 符号。例如，在报告脚本中使用 &CurrentMonth 来引用替换变量 CurrentMonth。以下 DataBlox 示例显示了 &CurrentMonth 在查询属性中的使用:

```
<blox:data
dataSourceName="QCC-Essbase"
query='<ROW("All Products") <COLUMN("All Time Periods")
&CurrentMonth <PAGE(Measures) Sales !'/>
```

当该查询执行时，&CurrentMonth 将被替换为 IBM DB2 OLAP Server 或 Hyperion Essbase 服务器中定义的值。

虽然替换变量能够帮助减少报告脚本维护工作量，但是，有时还是必须手工地更改 IBM DB2 OLAP Server 或 Hyperion Essbase 服务器中的值。作为 DB2 Alphablox 应用程序中的一种替代方法，可以在 JSP 页面中使用 Java 方法来自动地为当前月或其他报告时间段计算值，然后在报告脚本中替换该值。

使用 DB2 OLAP Server 或 Essbase 别名

可以使用 DB2 OLAP Server 或 Essbase 别名（即 DB2 OLAP Server 或 Essbase 立方体中定义的成员的备用名）来改进分析视图的可读性。可以使用别名来引用备用成员名（如在外语中的成员名），也可以引用产品标识值。DB2 Alphablox 支持在报告脚本中以及在各种属性值中使用别名。

缺省情况下，DB2 Alphablox 应用程序显示唯一的成员名，而不显示别名。如果要在分析视图中显示别名，请将 DataBlox useAliases 属性设置为 true。要了解更多信息，请参阅 *Developer's Reference* 一书中的 DataBlox Reference 一节。

使用小数

当显示具有指定小数位数的数字数据值时，可能需要在 DB2 OLAP Server 或 Essbase 查询语句中添加 {DECIMAL} 报告脚本命令。要了解更多信息，请参阅 *Developer's Reference* 一书中有关这些属性的详细描述。

使用下列 **Blox** 标记属性:	要使用属性的 **Blox**:	要使用的报告规范伪指令:
`<blox:grid id="myGrid" ... defaultCellFormat="#,###.00; -#,###.00" > <blox:cellFormat scope="{Sales}" format="#,###.##"/> <blox:cellAlert background="#3333ff" format="#,###.##;(#,###.##" scope="{Scenario}"/> </box:grid>`	GridBlox	{DECIMAL 2}

Microsoft Analysis Services

Microsoft SQL Server 提供了 Microsoft Analysis Services，后者可以用来从多种关系数据源（包括 Microsoft SQL Server、Oracle 以及其他关系数据源）中检索数据。虽然在功能上与 IBM DB2 OLAP Server 和 Hyperion Essbase 相似，但是，Microsoft 使用多维表达式语言（MDX）来查询 Microsoft Analysis Services 多维数据立方体。

要了解有关 Microsoft Analysis Services 的更多信息，请参阅下列资源:

书籍

Spofford, George. 2001. *MDX Solutions: With Microsoft SQL Server Analysis Services.* New York: John Wiley & Sons.

这是一本出色并且全面的教程／参考指南，它阐述了如何使用 MDX 来访问和分析决策支持数据。此书阐述了基本的和高级的 MDX 语句，并提供了最常见问题的确切解决方案。强烈建议认真的开发者阅读此书。

Jacobsen, Reed. 2000. *Microsoft SQL Server Analysis Services Step by Step.* Redmond, Washington: Microsoft Press.

这是一本很好的教程，它介绍了 Microsoft Analysis Services 的所有方面，包括数据库管理、构建数据库以及基本 MDX 用法。

新闻组

如果在上述书籍或 Microsoft 文档中找不到问题的答案，您可以求助于另一种优秀资源 – 因特网新闻组。但是，对于 MSAS，其实只有一个新闻组能够提供问题的答案: `microsoft.public.sqlserver.olap`。您可以使用这个对等新闻组来与其他管理员和开发者讨论 MSAS。此新闻组上的许多优秀贡献者使此新闻组成为其中一个最有用的计算新闻组。George Spofford（MDX 解决方案的缔造者）就曾经是一位经常造访的贡献者，他询问过许多棘手的问题。并且，有数位 Microsoft Analysis Services 小组成员从事着这个新闻组的工作并提供在别处可能找不到的具有洞察力的资源。

要加入此新闻组，请将新闻服务器指向:

`news:msnews.microsoft.com/microsoft.public.sqlserver.olap`

此外，可以通过以下链接访问 OLAP 新闻组，此链接列示了所有与 Microsoft SQL Server 相关的新闻组:

```
http://www.microsoft.com/sql/support/newsgroups/
```

并且，要搜索此新闻组的归档，请将 Web 浏览器指向 Google Groups（Usenet 新闻组的搜索引擎），网址为:

```
http://groups.google.com/
```

在 Google Groups 主页上的搜索字段中，输入以下内容:

```
microsoft.public.sqlserver.olap
```

并单击 Google Search 按钮。几乎会立即向您显示最新发布的内容。您还可以通过搜索关键字来进一步缩小搜索范围并同时将搜索限制为仅对此新闻组进行。当您急于了解问题的答案时，这个资源非常有用。

创建 MDX 查询语句

要将查询传递给 Microsoft SQL Server 2000 Analysis Services，请使用有效的 MDX SELECT 语句。MDX 语法与 SQL 语法有点相似。在以下简单查询语法中，请留意 SELECT、FROM 和 WHERE 关键字的使用:

```
SELECT axis specification ON COLUMNS,
axis specification ON ROWS
FROM cube_name
WHERE slicer_specification
```

注: 除非定义了列，否则对行执行的 MDX 语句查询是无效的。

以下表达式查询 Sales 立方体并返回 California 和 Washington 的所有商店的 Measures 维总结。Measures 维显示在列轴上，Store 维显示在行轴上:

```
SELECT Measures.MEMBERS ON COLUMNS, {[Store].[Store
State].[CA], [Store].[Store State].[WA]} ON ROWS
FROM [Sales]
```

要获取这些州中的每个州的成员（商店）详细信息，请添加 CHILDREN 关键字:

```
SELECT Measures.MEMBERS ON COLUMNS, {[Store].[Store State].
[CA].CHILDREN, [Store].[Store State].[WA].CHILDREN} ON ROWS
FROM [Sales]
```

注意，获取唯一成员名的方法是对维层次结构进行向下级联。例如，假定名为 Stores 的维具有以下层次结构:

```
All Stores
   Canada
   USA
     CA
     OR
   Mexico
```

以下是该层次结构中的有效唯一成员名:

```
[Store].[All Stores]
[Store].[All Stores].[USA]
[Store].[All Stores].[USA].[CA]
```

要了解有关 DB2 Alphablox 所支持的 MDX 语法子集的更多信息，请参阅《*DB2 Alphablox Cube Server 管理员指南*》。

Microsoft 站点以在线方式提供了多维表达式（MDX）的简介，该站点的网址为: http://msdn.microsoft.com。除了这些示例以外，该简介还提供了许多别的示例。

使用 autoDisconnect 属性来清除 PivotTable Services 高速缓存

如果您使用大型 MSAS 立方体，并且，由于 MDX 查询返回大型结果集而导致消耗过多的 PivotTable Services 内存高速缓存，从而导致发生可伸缩性问题，则您可能能够使用 DataBlox `autoDisconnect` 属性来帮助管理基于 MSAS 的分析应用程序的可伸缩性和性能。要了解此属性的用法详细信息，请参阅第 103 页的『自动连接和自动断开连接』。

DB2 Alphablox Cube Server

DB2 Alphablox Cube Server 支持使用一组有限 MDX 命令生成的查询。要了解这些命令及其用法的完整信息，请参阅*《DB2 Alphablox Cube Server 管理员指南》*。

将 SAP Business Information Warehouse（SAP BW）与 DB2 Alphablox 配合使用

在将 SAP BW 与 DB2 Alphablox 配合使用之前，必须满足所讨论的先决条件。

SAP Business Information Warehouse（SAP BW）是 SAP 的企业商业智能解决方案，可以通过 DB2 Alphablox 分析应用程序进行访问。可以使用由 DB2 Alphablox 提供的 OLE DB for OLAP 数据适配器来访问 SAP BW 数据源。通过使用 DB2 Alphablox，可以定制方式来构建复杂的分析应用程序以满足 SAP BW 环境中业务用户的需要。DB2 Alphablox 使用现有的 OLE DB for OLAP 基础结构通过 SAP OLE DB for OLAP 提供程序连接至 SAP BW。

必须满足以下先决条件，才能将 DB2 Alphablox 与 SAP BW 配合使用:

- 必须在 DB2 Alphablox 所在的机器上安装 SAP BW 3.5 Frontend 组件和 Microsoft Data Access Components（MDAC）。SAP BW 3.5 Frontend 包括在 SAP NetWeaver 软件包中。
- 已使用 DB2 Alphablox 测试了 Microsoft 的 MDAC 2.7.1、2.8.0 和 2.8.2。
- SAP Logon Frontend 组件应由 SAP 基础管理员安装。

一旦满足以上条件，您就可以为 SAP BW 数据源定义 DB2 Alphablox 数据源定义。

创建 SAP BW 数据源定义

在创建可以与 SAP BW 配合使用的分析应用程序之前，需要在 DB2 Alphablox 中创建数据源定义。

为了在 DB2

Alphablox 中创建 SAP BW 数据源定义，确保您已满足『将 SAP Business Information Warehouse（SAP BW）与 DB2 Alphablox 配合使用』中描述的先决条件。

要创建 DB2

Alphablox 数据源定义:

1. 在 DB2 Alphablox 管理页面中，单击“管理员”选项卡，然后单击**数据源**菜单项。

2. 在列表下面，单击**创建**按钮。 出现**创建数据源**窗口。
3. 在“数据源”字段中输入您的 SAP BW 数据源。
4. 从**适配器**选择列表中，选择 OLE DB for OLAP 选项。 屏幕将刷新与此适配器相关的字段。
5. 在 **OLAP Server** 字段中，输入**前端**配置系统的**描述字段**中的值。
6. 对于**缺省数据库**字段，输入 InfoCube 的名称。
7. 让**缺省模式**字段保留为空白。
8. 在**提供程序**字段中，输入一个表示您的 SAP OLE DB for OLAP 提供程序的值。向 SAP 基础管理员询问 SFC_CLIENT 和 SFC_LANGUAGE 值。 提供程序字符串应该为以下内容:

   ```
   MDrmSAP;SFC_CLIENT=clientValue;SFC_LANGUAGE-languageValue
   ```

   其中 clientValue 和 languageValue 是您从 SAP 基础管理员处获取的值。
9. 使用 SAP 基础管理员给您提供的用户标识和密码值填充**缺省用户名**和**缺省密码**字段。
10. 单击**保存**按钮。

您现在已创建了一个可以与 DB2

Alphablox 分析应用程序配合使用的 DB2

Alphablox 数据源。

创建 MDX 查询以便与 SAP BW 配合使用

通过使用受 SAP BW 支持的 MDX 语法，您可以开始构建 DB2 Alphablox 应用程序。

此阶段只需这样设置。

- 通过使用 Blox API，您可以使用 DataBlox setQuery() 方法来设置 SAP BW MDX 查询。 有关使用此方法的详细信息，请参阅 *Blox API Javadoc* 文档。 以下是使用 setQuery() 方法来设置 MDX 查询的示例:

  ```
  DataBlox.setQuery("SELECT DISTINCT( crossjoin ( {[0CALYEAR].[2001]},
  {[0D_SALE_ORG].[All]})) ON AXIS(0), DISTINCT( {[Measures].[0D_COST],
  [Measures].[0D_INV_QTY], [Measures].[0D_NETVLINV], [Measures].[0D_TAXAMOUN]})
  ON AXIS(1) FROM [$0D_DECU]");
  ```

- 当因为 SAP OLE DB for OLAP 提供程序的需要而使用 DB2 Alphablox API 时，请在查询和自变量中仅使用唯一名称。 以下是 DB2 Alphablox 方法的示例，该方法使用来自 SAP BW 数据源的唯一名称:

  ```
  ODBOMetaData.resolveMember("[Measures].[0D+NETVLINV]");
  ```

- 有关 SAP BW 以及它的 MDX 表达式语言用法的更多信息，请访问 SAP Help Portal Web 站点（http://help.sap.com/）或参阅您的 SAP BW 文档。
- DB2 Alphablox 当前不支持 SAP BW 上的固有深入钻取和回写。

对 DB2 OLAP Server 和 Hyperion Essbase 的深入钻取支持（使用 EIS）

OLAP 数据源为业务分析员和行业用户提供了对数据趋势的具有深入洞察力的信息，但是，除非提供了某种机制来对底层数据源进行深入钻取，否则不允许这些用户访问原始数据。深入钻取支持（如果 OLAP 数据源提供了此支持的话）允许用户更深入地了解所选单元格在 OLAP 数据库底层事实表中的记录所包含的原始数据。

DB2 OLAP Server 或 Essbase Integration Services 允许 DB2 OLAP Server 或 Essbase 管理员将多维数据映射至更详细的关系数据。可以现成地将 Integration Services 与 Microsoft Excel 配合使用以便查看 EIS 服务器上提供的任何预定义深入钻取报告。使用 Excel 生成的报告是基本报告，这些报告:

- 仅提供了行和列
- 未提供数据格式化
- 不能对用户交互进行控制
- 用户不能同时查看多个报告或工作表

通过使用 DB2 Alphablox 对 EIS 的深入钻取支持，用户可以从 DB2 OLAP Server（或 Essbase）中存储的概要数据和计算的数据深入钻取到关系仓库中（使用星型模式）存储的详细数据。DB2 Alphablox 对 Integration Services 的深入钻取支持利用预定义的 Integration Services 深入钻取关系报告、易于启用和配置并且提供了强大的功能和灵活的定制特性。

现成的 Integration Services 深入钻取支持

通过将 GridBlox `drillThroughEnabled` 属性设置为 true（缺省值为 `false`），您只需完成最少量的工作就可以开始使用 DB2 OLAP Server 或 Hyperion Essbase Integration Services 的固有深入钻取支持。此支持一旦被启用，就会在上下文（右键单击）菜单中添加“深入钻取”菜单选项（当您右键单击网格数据单元格时，将打开此菜单）。DB2 Alphablox 将自动生成一个对话框窗口，此窗口提供了可用 Integration Services 深入钻取报告列表（由 EIS 管理员预定义）。在用户选择报告之后，内置的缺省 JSP 页面将打开一个独立的浏览器窗口，并且在该浏览器窗口中的基本交互式“Alphablox 关系报告”视图中显示所返回的报告数据。

Relational Reporting Developer's Guide 一书提供了关于 DB2 Alphablox 关系报告功能的详细信息。

控制 EIS 深入钻取窗口样式

虽然缺省的深入钻取窗口对于实现您的目标来说可能已经足够了，但是，DB2 Alphablox 还允许您使用嵌套的 GridBlox `<blox:drillThroughWindow>` 标记来定制显示窗口。当此标记嵌套在启用了深入钻取支持的 GridBlox 中时，它将覆盖缺省的现成行为并允许定义定制浏览器窗口属性。

`<blox:drillThroughWindow>` 的标记属性以最常用的窗口定义属性为模型，该窗口定义属性是在用来打开浏览器窗口的 JavaScript `window.open(url,windowName,features)` 方法的 `features` 自变量中定义的。支持的属性包括:

标记属性
　　描述

url 为深入钻取窗口定义 JSP 的位置

name 深入钻取窗口的名称

height 窗口的高度

width 窗口的宽度

resizable
这是一个布尔属性，它确定用户是否可以调整深入钻取窗口的大小。缺省值为 true。

statusbarVisible
这是一个布尔属性，它确定深入钻取浏览器窗口的状态栏是否应该可视。缺省值为 true。

scrollbarVisible
这是一个布尔属性，它确定深入钻取浏览器窗口的滚动条是否应该可用。缺省值为 true。

locationbarVisible
这是一个布尔属性，它确定深入钻取浏览器窗口的位置栏（地址栏）是否应该显示。缺省值为 true。

toolbarVisible
这是一个布尔属性，它确定深入钻取浏览器窗口的工具栏是否应该显示。缺省值为 true。

menubarVisible
这是一个布尔属性，它确定深入钻取浏览器窗口的菜单栏是否应该显示。缺省值为 true。

要了解有关 `<blox:drillThroughWindow>` 标记及其属性的详细信息，请参阅 *Developer's Reference* 一书的 GridBlox Reference 一节。

使用 DB2 Alphablox 关系报告来定制 EIS 深入钻取支持

可以使用 DB2 Alphablox 关系报告 Blox（*Relational Reporting Developer's Guide* 一书对其进行了全面讨论）来生成定制深入钻取支持，定制深入钻取支持能够提供许多在使用 Microsoft Excel 的固有 EIS 深入钻取支持中不可能实现的所需功能。灵活的 DB2 Alphablox 允许您定制深入钻取行为。通过使用 DB2 Alphablox EIS 深入钻取支持，开发者能够:

- 通过将列永久地隐藏在结果集中，从而提供用户级别或角色级别的安全性
- 控制结果集列的顺序
- 将计算的列添加到结果集中
- 创建中断组和总计
- 将列重命名
- 格式化数据
- 同时打开多个报告

使用 RDBREsultSetDataBlox 和 RDBResultSetTag

当使用关系报告来显示定制报告时，请考虑下列各点:

- RDBResultSetDataBlox 和 RDBResultSetTag 允许引用 DataBlox 并获取它的 RDBResultSet，然后将其用作关系报告流水线的数据“生成器”。
- RDBResultSetDataBlox 使用指向关系数据源的 DataBlox 或者指向启用了深入钻取的 DB2 OLAP Server 或 Essbase 数据源的 DataBlox。如果指定了 rowCoordinate 和 columnCoordinate，则 RDBResultSet 的数据应该来自对 bloxRef 属性所引用的 DataBlox 执行的深入钻取操作。要使深入钻取操作能够对 DB2 OLAP Server 或 Essbase 数据源起作用，还必须设置报告名。如果未指定 rowCoordinate 或 columnCoordinate，则 RDBResultSet 的数据应该来自对 bloxRef 属性所引用的 DataBlox 执行的 getResultSet 调用。在这两种情况下，如果无法正确地执行操作（drillthrough 或 getResultSet），则将返回 null。

支持多个报告

要对每个单元格支持多个报告，应用程序必须能够处理多个报告。例如，对于每个报告，您可能想根据不同的列进行分组。并且，每个报告都还可能使用不同的 CSS 样式表。可以使用 controller.jsp 文件来将报告分发给适当的 JSP 页面。此外，如果报告不复杂，则可以使用单个 JSP 文件来处理所有报告。在任何一种情况下，都需要将报告名传递到 JSP 页面中，这是因为，RDBResultSetDataBlox 需要报告名才能工作。

添加定制菜单选项

通过使用 Blox UI 模型，您可以创建自己的定制菜单选项（例如“Drill to Relational”），以便当用户右键单击数据单元格时显示该菜单选项。在这种情况下，您可以使用自己的定制选项，而不是使用 DB2 Alphablox 中提供的现成菜单选项。

在以下 Java 代码示例中，对网格的右键单击菜单添加了“Drill to Relational”选项:

```
Menu cellMenu = new Menu();
cellMenu.add(new MenuItem("cellItem","Drill To Relational"));
grid.setCellsRightClickMenu(cellMenu);
// Add a dedicated controller to the cell menu
DataBlox db = presentBlox.getDataBlox();
cellMenu.setController(new DrillingController(db,grid,
    abSessionName,appName));
```

根据需求的不同，可以创建 DrillController 来处理深入钻取并且在报告返回过多数据行时对特定单元格禁用深入钻取。此外，您甚至可以从一个单元格中同时打开两个不同的报告。

其他定制 EIS 深入钻取支持

如果您决定构建自己的定制深入钻取支持，并且不想使用 DB2 Alphablox 关系报告来显示数据，则需要使用下列 DataBlox 方法:

- RDBResultSet drillThrough(String reportName, Tuple[] coordinates);
- RDBResultSet drillThrough(String reportName, int columnCoordinate, int rowCoordinate);
- String[] getDrillThroughReportNames(int columnCoordinate, int rowCoordinate);

使用返回的 RDBResultSet 来构建您自己的定制报告，该报告对行和列进行迭代以获取数据。并且，使用 getDrillThroughReportNames 来返回特定单元格的可用深入钻取报告列表，然后对特定报告和单元格调用深入钻取。

要了解有关 RDBResultSet 对象及 DataBlox 方法的详细信息，请参阅 *Developer's Reference* 一书中的 DataBlox Reference。

对 Microsoft Analysis Services 的深入钻取支持

OLAP 数据源为业务分析员和行业用户提供了对数据趋势的具有深入洞察力的信息，但是，除非提供了某种机制来对底层数据源进行深入钻取，否则不允许这些用户访问原始数据。深入钻取支持（如果 OLAP 数据源提供了此支持的话）允许用户更深入地了解所选单元格在 OLAP 数据库底层事实表中的记录所包含的原始数据。

可以使用 MDX DRILLTHROUGH 语句来从用来创建指定单元格的事实表（关系数据源）中检索源行集，或者从 Microsoft Analysis Services 立方体中的元组检索源行集。以下是 Foodmart 的示例 DRILLTHROUGH 语句，Foodmart 是 Microsoft Analysis Services 附带提供的样本 OLAP 数据库:

```
DRILLTHROUGH SELECT FROM [Inventory] WHERE (  [Product].[ByManufacturer].[All
Product].[Acme],  [Warehouse].[Whse 8], [Time].[Aug. 2000] )
```

DB2 Alphablox 对 MSAS 的固有深入钻取支持使用以下 DRILLTHROUGH 语句来检索底层结果集:

```
DRILLTHROUGH [<Max_Rows>] [<First_Rowset>] <MDX SELECT>
```

其中，<Max_Rows> 等同于 MDX MAXROWS 值，<First_Rowset> 等同于 MDX FIRSTROWSET 值，<MDX SELECT> 是自动生成的 SQL 查询。<Max_Rows> 值是从 DB2 Alphablox 数据源定义设置中获取的。

在能够使用 DRILLTHROUGH 语句来从底层数据源中检索行集之前，首先必须在"深入钻取选项"对话框中启用 MSAS 立方体以允许进行深入钻取并指定要返回的列。并且，客户机应用程序必须提供深入钻取支持。[要了解有关对立方体配置深入钻取选项的详细信息，请参阅 Microsoft 的 SQL Server Books Online 文档，可以从 Microsoft SQL Server 菜单打开该文档。]

虽然某些客户机应用程序提供了有限的深入钻取支持，但是，DB2 Alphablox 对 MSAS 的深入钻取支持易于配置，并且还允许定制深入钻取行为，这是其他客户机应用程序所不支持的。借助 DB2 Alphablox 深入钻取支持，您可以:

- 控制结果集列的顺序

  虽然 Microsoft Analysis Services 允许您选择要在深入钻取操作中显示的列，但是，不能对结果记录中的列进行排序。DB2 Alphablox ReportBlox 功能允许定义列的显示顺序。可以配置 OrderBlox 以指定结果集列应该具有的显示顺序。
- 通过将列永久地隐藏在结果集中，从而提供用户级别／角色级别的安全性。

  MSAS 的内置安全性仅允许您启用或不启用深入钻取；不能根据用户角色来控制哪些深入钻取列可用。通过使用 ReportBlox 的嵌套的 MembersBlox，可以永久地隐藏列。最终用户无法使用 UI 来再次显示列，但开发者可以使用适当的 API 调用来显示列。
- 将计算的列添加到结果集中

  通过使用 ReportBlox 的 CalculateBlox，可以在关系报告中复制计算，尽管 MSAS 本来并不允许返回计算的列。
- 在不同的窗口中打开多个报告

DB2 Alphablox 提供的现成深入钻取支持允许最终用户打开多个报告窗口，以便对来自多个网格单元格的深入钻取数据进行比较。您还可以进行定制编码以开发自己的打开多个窗口的解决方案 – 在“Blox 样本程序”中，在“检索数据”部分的有关 Microsoft Analysis Services 的内容中，有一个示例显示了一种完成此操作的方法。

现成的 Microsoft Analysis Services 深入钻取支持

通过将 GridBlox `drillThroughEnabled` 属性设置为 true（缺省值为 `false`），您只需完成最少量的工作就可以快速地开始使用 Microsoft Analysis Services 的固有深入钻取支持。此支持一旦被启用，就会在上下文（右键单击）菜单中添加“深入钻取”选项（当您右键单击网格数据单元格时，将打开此菜单）。DB2 Alphablox 将自动地生成适当的 `DRILLTHROUGH` 语句并执行查询。返回的结果集将显示在一个独立浏览器窗口中的交互式 ReportBlox 中。

控制深入钻取窗口样式

虽然缺省的深入钻取窗口对于实现您的目标来说可能已经足够了，但是，DB2 Alphablox 还允许您使用嵌套的 GridBlox `<blox:drillThroughWindow>` 标记来定制显示窗口。当此标记嵌套在启用了深入钻取支持的 GridBlox 中时，它将覆盖缺省的现成行为并允许定义定制浏览器窗口属性。

`<blox:drillThroughWindow>` 的标记属性以最常用的窗口定义属性为模型，该窗口定义属性是在用来打开浏览器窗口的 JavaScript `window.open(url,windowName,features)` 方法的 `features` 自变量中定义的。支持的属性包括:

标记属性
: **描述**

`url` 为深入钻取窗口定义 JSP 的位置

`name` 深入钻取窗口的名称

`height` 窗口的高度

`width` 窗口的宽度

`resizable`
: 这是一个布尔属性，它确定用户是否可以调整深入钻取窗口的大小。缺省值为 true。

`statusbarVisible`
: 这是一个布尔属性，它确定深入钻取浏览器窗口的状态栏是否应该可视。缺省值为 true。

`scrollbarVisible`
: 这是一个布尔属性，它确定深入钻取浏览器窗口的滚动条是否应该可用。缺省值为 true。

`locationbarVisible`
: 这是一个布尔属性，它确定深入钻取浏览器窗口的位置栏（地址栏）是否应该显示。缺省值为 true。

`toolbarVisible`
: 这是一个布尔属性，它确定深入钻取浏览器窗口的工具栏是否应该显示。缺省值为 true。

menubarVisible

这是一个布尔属性，它确定深入钻取浏览器窗口的菜单栏是否应该显示。缺省值为 true。

要了解有关 <blox:drillThroughWindow> 标记及其属性的详细信息，请参阅 *Developer's Reference* 一书的 GridBlox Reference 一节。

使用 DB2 Alphablox 关系报告的定制深入钻取支持

可以使用 DB2 Alphablox 关系报告 Blox（*Relational Reporting Developer's Guide* 一书对其进行了全面讨论）来生成定制深入钻取支持，定制深入钻取支持能够提供许多在固有的 Microsoft Analysis Services 深入钻取支持中不可能实现的所需功能。在"Blox 样本程序"中的"检索 DB2 OLAP Server（和 Essbase）以及 Microsoft Analysis Services 的数据"下面，有一个定制深入钻取支持示例。下面，我们浏览一遍代码并对最重要的代码部分进行解释:

1. 在 JSP 文件中，通过对 <blox:grid> 标记添加 drillThroughEnabled 属性并将值设置为 truee 来启用"深入钻取"上下文（右键单击）菜单选项:

   ```
   <blox:grid ...
   drillThroughEnabled="true" ... />
   ```

2. 通过对 PresentBlox 添加 <bloxui:actionFilter> 标记，设置拦截并处理"深入钻取"上下文（右键单击）菜单选项的右键单击事件:

   ```
   <bloxui:actionFilter
   className="<%= MyDrillThroughClass.class.getName() %>"
   componentName="dataAdvancedDrillThrough" />
   ```

   componentName 属性被设置为 dataAdvancedDrillThrough 值，className 属性必须被设置成为了处理右键单击事件而添加的类的名称。

3. 在页面顶部添加 JSP page 伪指令以指定页面所需的类:

   ```
   <%@ page import="com.alphablox.blox.uimodel.ModelConstants,
   com.alphablox.blox.uimodel.tags.IActionFilter,
   com.alphablox.blox.DataViewBlox, com.alphablox.blox.uimodel.core.Component,
   com.alphablox.blox.uimodel.core.MessageBox,
   com.alphablox.blox.uimodel.GridBrixModel,
   com.alphablox.blox.uimodel.PresentBloxModel,
   com.alphablox.blox.uimodel.core.grid.GridCell,
   com.alphablox.blox.uimodel.GridBrixCellModel,
   com.alphablox.blox.uimodel.core.ClientLink" %>
   ```

4. 为 actionFilter 添加处理程序类:

   ```
   <%!
     public static class MyDrillThroughClass implements IActionFilter
   {
       public void actionFilter( DataViewBlox blox, Component component )
       throws Exception {
           GridBrixModel grid =
              ((PresentBloxModel)blox.getBloxModel()).getGrid();
           GridCell[] cells = grid.getSelectedCells();
   // Make sure that a single data cell is selected
   if (cells.length != 1 || cells[0].isRowHeader() ||
      cells[0].isColumnHeader() || !(cells[0]
         instanceof GridBrixCellModel)) {
   MessageBox.message( component, "Error", "You must select a single
   ```

```
data cell to drill through" );
return;
}
       GridBrixCellModel cell = (GridBrixCellModel)cells[0];
       int rowIndex = cell.getNativeRow();
       int colIndex = cell.getNativeColumn();
       String bloxName = blox.getBloxName();
String urlStr = "someReportBlox.jsp?bloxRef="+bloxName;
urlStr += "&colIndex=";
urlStr += colIndex;
urlStr += "&rowIndex=";
urlStr += rowIndex;
String timestamp = String.valueOf(System.currentTimeMillis());
urlStr += "&reportName=";
urlStr = urlStr + "reportBlox"+timestamp;
ClientLink link =
new ClientLink(urlStr,"reportBlox"+timestamp);
component.getDispatcher().showBrowserWindow( link );
}
}
%>
```

5. 设置定制关系报告。在本示例中，someReportBlox.jsp:

```
<%
String reportName = request.getParameter("reportName");
if(reportName == null) {
reportName = "defaultName";
}
%>
```

6. 使用 Blox 报告标记库的 <blox:RDBResultSetData> 或 <bloxreport:RDBResultSetData> 标记来设置定制关系报告。

```
<blox:report id="drillThrough"
bloxName="<%= reportName %>"
interactive="true">
<blox:rdbResultSetData
bloxRef="<%= request.getParameter(\"bloxRef\") %>"
columnCoordinate="<%= request.getParameter(\"colIndex\") %>"
rowCoordinate="<%= request.getParameter(\"rowIndex\") %>">
</blox:rdbResultSetData>

...
```

   [可选] 如果要支持打开多个报告，则设置缺省的 ReportBlox 名称。

   [可选] 如果正在使用多个报告，则设置 ReportBlox 的 bloxName 属性。可以在 *Developer's Reference* 一书的 Common Blox Reference 一节中找到 bloxName 属性的用法详细信息。

   [可选] 如果要从视图中永久地排除成员，则定义 MembersBlox。

   [可选] 如果要使用中断组和聚集来创建更清晰的布局，则定义 GroupBlox。

   [可选] 定义 CalculateBlox 以添加 MSAS 本来不支持的复制的计算。

   [可选] 定义 OrderBlox 以对列进行排序。MSAS 深入钻取支持本来不允许对列进行重新排序。

请查看完整的代码以在"Blox 样本程序"示例中执行定制深入钻取。

有关创建关系报告视图的详细信息，请参阅 *Relational Reporting Developer's Guide*。在以下示例中，我们将阐述一些与使用关系报告来进行深入钻取相关的要点。在本示例

中，我们将把所检索的页面称为 someReportBlox.jsp 并使用 <blox:RDBResultSetData> 标记来使深入钻取数据到达关系流水线中。

其他定制深入钻取支持

为了提供您自己的定制，一个选项将拦截右键单击事件（该事件控制着由 DB2 Alphablox 提供的深入钻取支持），然后使用服务器端 ClickEvent 来管理深入钻取定制。

如果您想要使用自己的解决方案以开发深入钻取报告，而不是使用 Blox 报告标记库，则需要使用下列其中一个 DataBlox 方法:

- RDBResultSet drillThrough(Tuple[] coordinates)
- RDBResultSet drillThrough(int columnCoordinate, int rowCoordinate)

这些方法将返回一个 RDBResultSet，后者包含对指定坐标位置执行的深入钻取操作得到的关系数据。然后，您就可以根据需要来显示关系深入钻取数据了。

要了解有关 RDBResultSet 对象的详细信息，请参阅 *Developer's Reference* 一书中 DataBlox Reference 的 Relational Result Set Methods 一节。

关系数据源

DB2 Alphablox 支持您使用表示 Blox 来查看关系结果集。与其他受支持的数据库类似，您需要指定数据源和查询。当关系结果集显示时，在标准表示 Blox 中提供了有限的一小组 DB2 Alphablox 功能。ReportBlox 还可以用来显示关系数据以及提供许多对报告的支持。要了解有关使用新增的关系报告功能来显示关系报告的详细信息，请参阅 *Relational Reporting Developer's Guide* 一书。

下面的内容提供了 SQL 语句的概述和总结并阐述了如何将它们与 DB2 Alphablox 配合使用。

创建 SQL 语句

要将查询传递给关系数据源，请使用 RDBMS 支持的 SQL SELECT 语句语法。例如，以下 SQL 语法受几种关系数据源支持:

```
SELECT... FROM... WHERE... ORDER BY... GROUP BY...
```

- SELECT (ALL|DISTINCT) [COLUMNS] 用于标识要包括在结果集中的数据列
- FROM [TABLELIST] 用于标识每个提供数据的数据库表的名称
- WHERE [PREDICATE EXPRESSION] 用于对数据指定过滤器和连接
- ORDER BY [COLUMNNAMES] 用于指定排序顺序
- GROUP BY [COLUMNNAMES] 用于指定组列表

注: 不支持函数。

以下示例指定了:

- 从两个表（分别名为 Actual 和 Projected）中选择名为 SalesQty 和 ProductID 的列
- 只选择实际销售量小于计划销售量的行

```
<blox:data query="SELECT Actual.SalesQty, Actual.ProductID,
Projected.SalesQty, Projected.ProductID FROM Actual,
Projected WHERE Actual.SalesQty < Projected.SalesQty".../>
```

查询构建器

查询构建器提供了一种方便的方法来开发和测试查询语法。此工具使用点击界面，并且不要求您深入了解数据源的查询语言。

查询构建器支持交互式的查询开发工作。您仅仅是处理一个网格，该网格在获得适当的数据视图之前将显示缺省结果集。单击**获取当前查询**按钮将显示具有正确语法的查询语句。您可以将该语句复制并粘贴到一个文本文件中以供将来使用，也可以将其直接粘贴到应用程序模板的查询值中。

生成 Blox 标记按钮是查询构建器的另一项强大功能，此按钮允许您检索重新生成显示同一视图和结果集的 PresentBlox 所需的 Blox 标记语法。同样，您可以将标记复制并粘贴到文本文件中以供将来使用，也可以将其直接粘贴到应用程序中。

使用查询构建器

以下任务显示如何使用查询构建器来开始了解查询语句以及它们对 DB2 Alphablox 应用程序中的视图有何影响。

要访问查询构建器:

1. 在 Application Studio 页面上，单击**工作台**。“工作台”页面打开。
2. 单击**查询构建器**链接。
3. 单击**连接设置**按钮以打开下拉列表。选择要为其构建查询的数据源。“目录”、“模式”、“用户名”和“密码”字段的值是从数据源定义中获取的，这些值将显示在文本框中。
4. 如果有必要的话，更改“目录”、“模式”、“用户名”和“密码”的值。如果要对该连接执行缺省查询，则单击**执行缺省查询**复选框。
5. 单击“连接”按钮。在成功地进行连接之后，将有一个确认指示器显示在“状态框架”中。
6. 执行下列任何一项操作:
   - 在文本框中输入查询字符串并单击**执行查询**按钮。查询结果将显示在页面底部的 PresentBlox 中。

     单击**获取当前查询**按钮以对此数据源检索最近的成功查询。查询字符串将显示在文本框中。
   - 单击**缺省查询**按钮。与此数据源相关联的缺省查询字符串将显示在文本窗口中。如果该数据源是 Alphablox 立方体或 Microsoft Analysis Services 立方体，则确保立方体名正确。

     注: 对于关系数据库连接池功能，您将无法使用**缺省查询**按钮来对关系立方体获取缺省查询。
   - 要查看查询结果，单击**执行查询**按钮。结果集会出现在页面底部的 PresentBlox 中。

- 当数据出现在页面底部的 PresentBlox 中时，可使用标准用户界面来交换轴、向上钻取或向下钻取、在轴之间移动维或执行其他操作。
- 单击**生成 Blox 标记**按钮。这将打开一个文本框，该文本框包含用于复制视图的完整标记，结果集将显示在 PresentBlox 中。
- 展开**不对称查询构建器**窗格，取消选择要除去的列，然后单击**应用列集**按钮。

  注: “不对称查询构建器”窗格只能与 DB2 OLAP Server 或 Essbase 数据源配合使用，并且仅作用于列头。

7. 在开发适当的数据视图之后，单击**获取当前查询**按钮。文本框将显示开发当前数据视图所需的查询字符串。

   在确定适当的查询字符串之后，可以将其复制并粘贴到文本文件中以供将来使用，也可以直接粘贴到查询值中。要退出“查询构建器”，请关闭它的浏览器窗口。
8. 单击**生成 Blox 标记**按钮。这将打开一个文本框，该文本框显示了用于重新生成 PresentBlox 布局和结果集的标记。您可以将此标记复制并粘贴到文本文件中以供将来使用，也可以将其直接粘贴到应用程序中。

使用 JDBC 数据源

DataBlox 使您可以连接至多维数据源以及关系数据源。一旦通过 DB2 Alphablox 管理页面对 DB2 Alphablox 定义了您的关系数据源，您就可以使用 DataBlox 来连接至数据源并检索数据。然而，使用 JDBC 数据源时，您可能需要执行特定的 JDBC 调用、获取 JDBC URL 连接字符串或利用连接池或存储过程。DB2 Alphablox 提供了一个 JDBCConnection Bean，它允许您构造与 DB2 Alphablox JDBC 数据源的 JDBC 连接字符串。DB2 Alphablox 还提供了 StoredProceduresBlox 以在关系数据库中使用存储过程。

使用 JDBCConnection Bean

JDBCConnection bean 是一个允许您获取关于 DB2 Alphablox 关系数据源的信息的 Java bean。通过 JDBCConnection bean，您可以获取 JDBC URL 连接字符串并执行 JDBC 调用而不必创建 Blox。或者，您可以使用此 bean 来覆盖 DB2 Alphablox 中定义的关系（JDBC）数据源的属性。

JDBCConnection bean 是 `com.alphablox.blox.data.rdb` 包中的类，您必须在任何使用此 bean 中的任何 API 的 JSP 文件开头使用以下 JSP import 语句:

```
<%@ page import="com.alphablox.blox.data.rdb.*" %>
```

JDBCConnection bean 示例

以下是一个样本 JSP 文件，它使用 JDBCConnection bean 来打印 JDBC URL 连接字符串。

```
<%@ page import="com.alphablox.blox.data.rdb.*" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.io.*" %>
<html>
<head>
<title>JDBC Connection Bean Example</title>
</head>
<body>
<%
String ds = (String)request.getParameter( "ds" ) ;
```

```
%>
<form name=form method=get>
Enter data source name: 
<input name="ds" value="<%= ds == null ? "" : ds %>"><br />
<input type=submit value="Go"><br />
</form>
<%-- Create the Bean --%>
<jsp:useBean id="jbean"
class="com.alphablox.blox.data.rdb.JDBCConnection"
scope="session" />
<%-- Put in try statement to catch errors --%>
<% try { %>
<%--Test if there is a data source --%>
<% if ( ds != null ) { %>
<%
jbean.setDataSourceName( ds );
%>
<%-- Use the Alphablox bean to get the connection JDBC string --%>
<%= "URL = " + jbean.getURL() %><br />
Properties = <%= jbean.getConnectionProperties( ) %><br />
<%
Connection connection = jbean.createConnection( );
%>
Connection = <%= connection  %><br />
<br />
<%-- If no data source, prompt for one --%>
<% } else { %>
<br />
<b>Please enter a relational data source name!</b>
<br />
<% } %>
<%-- Catch the exception --%>
<% } catch ( Exception e ) {
out.write( "<br />An error has occured: <b>"
+ e.getMessage() + "</b>" ); } %>
 </body>
</html>
```

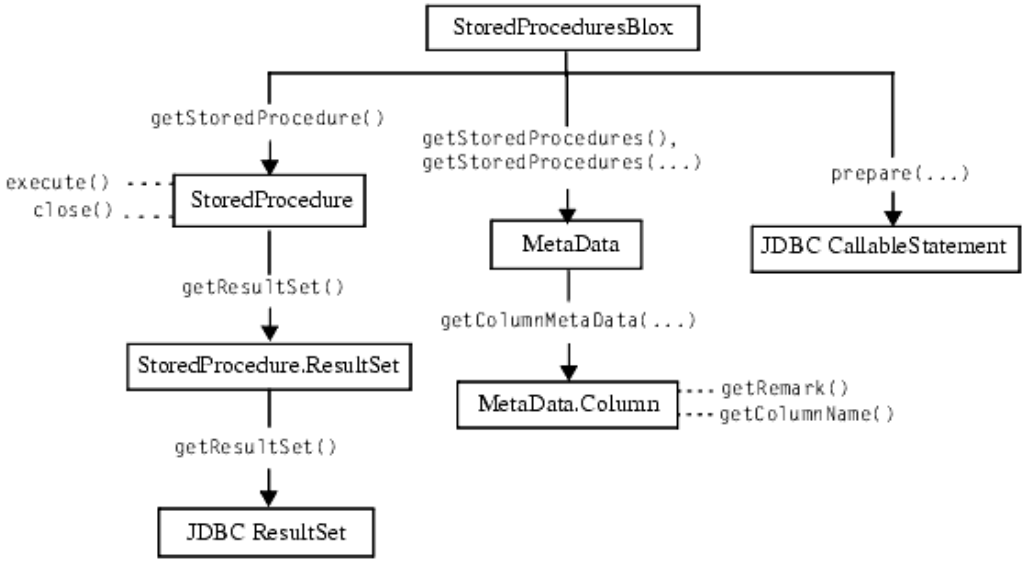
使用 StoredProceduresBlox

StoredProceduresBlox 是使用关系数据库存储过程的起始点。它允许您创建数据库连接和准备存储过程语句。一旦设置了正确的 DB2 Alphablox 数据源和任何其他连接参数，就可以执行以下操作:

- 使用 `prepare(...)` 方法来返回 JDBC CallableStatement 对象，该对象可用来设置任何执行存储过程所需要的存储过程参数
- 使用 `getStoredProcedure()` 方法来访问当前 StoredProcedure 对象，然后可以执行存储过程、获得已执行的存储过程的 ResultSet 或访问 JDBC ResultSet
- 使用 `getStoredProcedures()` 或 `getStoredProcedures(...)` 方法来返回一个或多个使您能够访问各个参数的 MetaData 对象

StoredProcedure 对象和 MetaData 对象是 `com.alphablox.blox.data.rdb.storedprocedure` 包中的单独类。通过从 StoredProceduresBlox 获得 StoredProcedure 和 MetaData 的单独对象，可以准备存储过程一次，然后执行它多次。即使可以在执行期间改变存储过程参数，您仍可以增强性能而不必在每次执行时都准备存储过程。

下图显示与存储过程相关的对象的对象层次结构。



因为 StoredProcedure 对象和 MetaData 对象位于单独的包中，所以您必须在任何 JSP 文件的开头使用以下 JSP import 语句，然后才能使用这些对象中的任何 API:

```
<%@ page import="com.alphablox.blox.data.rdb.storedprocedure.*" %>
```

注: IBM DB2 UDB、Sybase、Oracle 和 Microsoft SQL Server 数据库支持 JDBC 存储过程。

当使用 StoredProcedure 对象来执行已准备的存储过程时，请注意以下内容:

- 如果 DataBlox 用来显示来自存储过程的信息，则必须将 DataBlox 单独地连接至与 StoredProceduresBlox 相同的数据源。
- 如果 DataBlox 用来显示来自存储过程的信息且该存储过程还具有输出参数，则在获取输出参数之前，必须首先使用结果集。这是一个 JDBC 限制。
- 如果存储过程具有输入和输出参数，则应该使用 `StoredProceduresBlox.prepare(...)` 来获取 JDBC CallableStatement 对象。此对象允许您获取和设置存储过程上的输入和输出参数。
- 一旦执行了存储过程，且使用了任何输出参数和结果集，则您需要调用 `StoredProceduresBlox.disconnect()` 来断开与任何资源的连接并释放这些资源。如果您要使数据库连接保持打开，则请调用 `StoredProceduresBlox.close()` 以释放所使用的任何资源。
- 如果抛出了 DataException，则通过查看 `DataException.getNestedException()`，可能会以 SQLException 的形式获得额外的信息。

一旦执行了存储过程，则它返回 StoredProcedure.ResultSet 对象，该对象使您可以访问 JDBC ResultSet 对象。如果您需要直接使用 JDBC ResultSet 对象，则请使用 `ResultSet.getResultSet()` 方法来访问此对象。

建议您在使用存储过程时还导入 java.sql 包，所以您的 JSP 文件应该导入两个包:

```
<%@ page import="com.alphablox.blox.data.rdb.storedprocedure.*" %>
<%@ page import="java.sql.*" %>
```

StoredProceduresBlox 示例

本节包含 6 个示例，它们演示了如何使用 StoredProceduresBlox。要了解更多示例，请参阅 Javadoc 文档。

- 『连接至不具有 DataBlox 的数据源』
- 『使用 StoredProceduresBlox 来连接至要与 DataBlox 配合使用的数据源』
- 『获取名称与指定模式匹配的存储过程列表』
- 第 134 页的『获取每个存储过程的所有参数列表』
- 第 135 页的『执行具有一个输入参数和两个输出参数的存储过程』
- 第 135 页的『对 DataBlox 设置存储过程结果集』

连接至不具有 DataBlox 的数据源

本示例演示当您只想获取参数或运行不需要 DataBlox 的 INSERT SQL 存储过程时，如何连接至不具有 DataBlox 的数据源。

```
<%@ page import="com.alphablox.blox.StoredProceduresBlox" %>
<%@ page import="com.alphablox.blox.data.rdb.storedprocedure.*" %>
<%@ page import="java.sql.*" %>
<%@ taglib uri="bloxtld" prefix="blox" %>
<blox:storedProcedures id="mySP"/>
<%
mySP.setDataSourceName("sales");
mySP.connect();
%>
```

使用 StoredProceduresBlox 来连接至要与 DataBlox 配合使用的数据源

本示例演示在什么情况下才需要将用来显示来自存储过程的信息的 DataBlox 单独地连接至与 StoredProceduresBlox 相同的数据源。

```
<%@ page import="com.alphablox.blox.StoredProceduresBlox" %>
<%@ page import="com.alphablox.blox.data.rdb.storedprocedure.*" %>
<%@ page import="java.sql.*" %>
<%@ taglib uri="bloxtld" prefix="blox"%>
<blox:storedProcedures id="mySP"/>
<blox:data id="myDataBlox" visible="false"/>
<%
myDataBlox.setDataSourceName("sales-sql");
  myDataBlox.connect();
mySP.setDataSourceName("sales-sql");
mySP.connect();
%>
```

获取名称与指定模式匹配的存储过程列表

此示例演示如何使用 `getStoredProcedures(...)` 方法来获取名称以“procedure”开始的存储过程列表。此方法返回一组 MetaData 对象。MetaData 对象包含每个存储过程的参数信息。

```
<%@ page import="com.alphablox.blox.StoredProceduresBlox" %>
<%@ page import="com.alphablox.blox.data.rdb.storedprocedure.*" %>
<%@ page import="java.sql.*" %>
<%@ taglib uri="bloxtld" prefix="blox"%>
<blox:storedProcedures id="mySP"/>
<%
mySP.setDataSourceName("sales-sql");
mySP.connect();
MetaData procedures[] =
mySP.getStoredProcedures("procedure%");
```

```
%>
<%
if (procedures.length == 0) {
%> <strong>No procedures found.</strong> <%
} %>
```

通过 MetaData 对象，您可以访问指定存储过程的各个参数。

获取每个存储过程的所有参数列表

本示例演示如何使用 MetaData 对象来访问每个存储过程以及每个存储过程的参数。本示例假定您已返回一个 MetaData 对象，如先前示例中所示:

```
MetaData procedures[] =
mySP.getStoredProcedures("procedure%");
```

我们现在将在表中列示每个存储过程及其目录、模式、名称和备注信息:

```
<table border="1" >
<tr><th colspan="4">Stored Procedure Information</th></tr>
<tr><th>Catalog</th><th>Schema</th><th>Name</th><th>Remarks</th></tr>
<%
for (int i = 0; i < procedures.length; i++) {
String catalog = procedures[i].getCatalog();
String schema = procedures[i].getSchema();
String name = procedures[i].getName();
String rem = procedures[i].getRemark();
String type = null;
%>
<tr><td><%= catalog %></td>
<td><%= schema %></td>
<td><%= name %></td>
<td><%= rem %></td></tr>
<%
}
%>
</table>
```

我们还可以获取每个存储过程的每个参数的详细信息:

```
//for each of the stored procedure, we will get the MetaData.Column //
object which contains the detail of the parameters
<%
for (int spCount = 0; spCount < procedures.length; spCount++) {
String currProcedure = procedures[spCount].getName();
MetaData.Column cMeta[] =    procedures[spCount].getColumnMetaData();%>
//for the current stored procedure, we will get the list the
//detail for each parameter in a table
<table border="1">
<tr><th colspan="7">Stored Procedure Params for
<%=currProcedure %></th></tr>
<tr><th>Catalog</th><th>Schema</th><th>Name</th><th>Column Name</
th><th>Type</th><th>Type Name</th><th>Remark</th></tr>
//Iterate through the parameters in the current stored procedure
<% for (int i = 0; i < cMeta.length; i++) {
String catalog = cMeta[i].getCatalog();
String schema = cMeta[i].getSchema();
String name = cMeta[i].getName();
String colName = cMeta[i].getColumnName();
short type = cMeta[i].getType();
String typeName = cMeta[i].getTypeName();
String remark = cMeta[i].getRemark();
%><tr><td><%= catalog %></td>
<td><%= schema %></td>
<td><%= name %></td>
<td><%= colName %></td>
<td><%= type %></td>
```

```
<td><%= typeName %></td>
<td><%= remark %></td></tr><%
} %>
</table>
<% }
%>
```

执行具有一个输入参数和两个输出参数的存储过程

本示例演示 prepare() 方法如何返回 JDBC CallableStatement 对象，您可以使用该对象来执行具有输入和输出参数的存储过程。

```
<%@ page import="com.alphablox.blox.data.rdb.storedprocedure.*" %>
<%@ page import="com.alphablox.blox.data.rdb.*" %>
<%@ page import="com.alphablox.blox.StoredProceduresBlox" %>
<%@ page import="java.sql.*" %>
<%@ taglib uri="bloxtld" prefix="blox"%>
<blox:storedProcedures id="mySP"/>
<%
mySP.setDataSourceName("storeSales");
mySP.connect();
// param 1 is an integer output, param 2 is a string input,
// param 3 is a string output
CallableStatement cstmt = mySP.prepare("{call a_procedure(?, ?, ?)}");
cstmt.setString(2, "users/admin%");
cstmt.registerOutParameter(1, Types.INTEGER);
cstmt.registerOutParameter(3, Types.VARCHAR);
mySP.execute();
int out1 = cstmt.getInt(1);
String out3 = cstmt.getString(3);
%>
...
<!-- Closes all resources associated with executing the stored procedure -->
<%
mySP.close();
%>
...
<!--Disconnects from the data source -->
<%
mySP.disconnect();
%>
```

对 DataBlox 设置存储过程结果集

此示例演示如何将存储过程结果集输入到 DataBlox。

```
<%@ page import="com.alphablox.blox.data.rdb.*" %>
<%@ page import="com.alphablox.blox.data.rdb.storedprocedure.*" %>
<%@ page import="com.alphablox.blox.StoredProceduresBlox" %>
<%@ page import="java.sql.*" %>
<%@ taglib uri="bloxtld" prefix="blox"%>
<blox:storedProcedures id="mySP"/>
<blox:data id="myDataBlox" visible="false" />
<%
myDataBlox.setDataSourceName("sales-sql");
  myDataBlox.connect();
mySP.setDataSourceName("sales-sql");
mySP.connect();
mySP.prepare("{call a_procedure}");
mySP.execute();
mySP.loadResultSet(myDataBlox, 1);
%>
```

第 15 章 显示数据

公共的用户界面 Blox（GridBlox、ChartBlox 和 PresentBlox）为开发者提供了许多可供选择的分析视图定制方法。本主题将帮助您确定应该使用哪个 Blox 以及如何有效地使用用户界面 Blox。

选择用于显示数据的 Blox

在 DB2 Alphablox 应用程序中，可以使用任何公共表示 Blox（GridBlox、ChartBlox 和 PresentBlox）来向最终用户显示结果集，但具体使用哪个特定 Blox 的决定取决于用户需求。

用户界面 Blox 每次只能显示来自一个数据源的结果。如果需要在同一个页面上同时显示来自多个数据源的结果，则有几个选项:

- 在页面上放置多个 Blox，每个 Blox 指向不同的数据源。
- 在页面上放置一个 Blox，通过使用服务器端 Java API 或客户机端 Blox 客户机 API 更改数据源以显示来自多个数据源的结果。请参阅第 100 页的『使用 DataSourceSelectFormBlox 来设置不同的数据源』以获取示例。

选择数据表示 Blox 组件

正如本指南前面所讨论的那样，用户主要对用于显示数据的 Blox（ChartBlox、GridBlox 和 PresentBlox）感兴趣。本指南着重讲述在 DHTML 客户机中可用的用户界面 Blox 的用法。

下表对每个数据表示 Blox 的优点和缺点作了概述:

Blox	优点	缺点
GridBlox	• 用户可以方便地对具有小幅差别的数值进行比较 • 可以创建提醒以突出显示特定单元格中的信息 • 可以对个别单元格或一组单元格添加信息链接（使用 `cellLink` 和 `cellAlert` 属性） • 可以使用信息链接（使用应用程序定义中定义的头链接）来在行头和列头中添加信息链接	• 当用户尝试快速地查看差别时，他们可能更难以发现值之间的重大差别 • 不能访问 DataLayout 面板
ChartBlox	• 以各种图表格式显示多维数据或关系数据	• 当值只有少量变化时，直观地查看数据不容易发现值之间的差别 • 不能在图表上访问信息链接（请参阅 GridBlox 的优点） • 用户不能查看与数据相关的提醒 • 不能访问 DataLayout 面板

Blox	优点	缺点
PresentBlox	• 将数据的网格视图和图表视图组合到单个 Blox 中 • 用户可以在网格视图与图表视图之间进行切换，也可以同时显示这两个视图（当启用了分割窗格时） • DataLayout 面板允许高级用户处理维的显示 • 当只启用了网格或图表时，用户可以访问 DataLayout 面板或 Page 面板（在独立的 GridBlox 和 ChartBlox 中，这些面板不可视）	• 有时，当数据密度太高时（图表包含太多的数据点，难以进行解释），只应该允许用户查看网格；可以将 PresentBlox `chartEnabled` 属性设置为 `false`，但这将失去 PresentBlox 的其中一个优点

可供 DHTML 客户机使用的显示格式

DHTML 客户机的显示是不同的可用 Blox 显示格式的其中一种格式。在 DB2 Alphablox 显示页面上创建的新应用程序具有 DHTML 格式。其他显示格式是打印机、Excel、PDF 和 XML。根据所使用的特定显示格式的不同，可以通过公共 Blox 的 `render` 属性、通过 `render` URL 属性或通过本节描述的其他机制来设置显示格式。

DHTML 格式（render=dhtml）

缺省情况下，在 DB2 Alphablox 中创建的应用程序是使用 DHTML 显示的。此格式使用标准 DHTML 技术（HTML、CSS、DOM 和 JavaScript）以及服务器端 Java 技术来创建具有高度交互式格式的数据显示，DHTML 客户机与 Java 客户机的功能一样强大，但不需要使用 Java 插件或启用了 Java 的浏览器。DHTML 显示格式的另一项优点是，不需要 DHTML 格式的内置支持就可以扩展和定制由 Blox UI 模型定义的用户界面。

当使用公共 Blox 的 `render` 属性或 `render` URL 属性时支持 DHTML 显示格式。

注: 在 DHTML 中，如果页面被设置为以另一种格式显示，则会由于未装入基本的 JavaScript 文件而无法显示 Blox。

打印机格式（render=printer）

“打印机”格式生成一个 Blox 数据视图，该视图针对使用浏览器内置打印功能进行打印而进行优化。该 Blox 视图是使用 HTML 表和 CSS 样式生成的，并且还将所有可选择的页面过滤器转换为所选过滤器列表（包括维名及其所选成员）。“打印机”格式根据构建 DHTML 客户机的相同 Blox UI 模型生成可视表示。

“打印机”显示格式是作为公共 Blox 的 `render` 属性或 `render` URL 属性受支持的。

通常，要使用此格式，您将在带有 Blox 视图的页面上提供一个 HTML 按钮或链接，从而允许用户单击该按钮或链接以请求可打印的副本。作为对用户请求的响应，DB2 Alphablox 使用纯 HTML 来显示页面，然后将该页面传递给客户机。在页面显示在浏览器中之后，用户可以单击浏览器的“打印”按钮来将该页面发送至打印机。要了解详细信息，请参阅第 140 页的『打印 Blox 输出』。

PDF 格式（render=pdf）

通常，用户希望生成 PDF 文件的原因是为了供将来参考或者与别人共享。DB2 Alphablox 提供了一个备用打印传递机制，即 pdf 格式，该机制将页面上的用户界面 Blox 以 PDF 格式导出。当公共 Blox render 属性被设置为 pdf，或在 render URL 属性值被设置为 pdf 的情况下调用 JSP 页面时，该页面将直接装入 Adobe Acrobat。

对最终用户提供了更完善的**以 PDF 格式导出**功能，以允许他们将网格和图表中的数据以 Excel 格式导出。可在菜单栏中的“文件”菜单下找到**以 PDF 格式导出**选项。工具栏上也有一个**以 PDF 格式导出**按钮。当用户选择此选项时，会弹出一个对话框，以允许用户指定页面方向、要添加到 PDF 中的页眉或页脚以及他们想要网格或图表显示的方式。DB2 Alphablox 还为应用程序开发者提供了一些选项以便通过编程来定制页面布局，如指定图表的大小、何时应该出现换页符以及带有徽标和已定义文本的页眉和页脚。

要了解更多有关 PDF 报告的工作方式以及如何在应用程序中添加此功能的信息，请参阅第 218 页的『以 PDF 格式导出』。

以 Excel 格式导出（render=xls）

当 render Blox 属性或 render URL 属性设置为 xls 时，DB2 Alphablox 以 HTML 格式传递将装入到 Microsoft Excel 电子表格的输出。当使用此格式时，将把返回的页面的 MIME 类型设置为 application/vnd.ms-excel，即 Microsoft Excel 的标准 MIME 类型。

因为此显示格式只设置 HTML 输出上的 MIME 类型，所以数据不是本机 Excel 格式。网格中的数据按原样装入，而图表则作为图形装入。

对最终用户提供了更完善的**以 Excel 格式导出**功能，以允许他们将网格和图表中的数据以本机 Excel 格式导出。可在菜单栏中的“文件”菜单下找到**以 Excel 格式导出**选项。工具栏上也有一个**以 Excel 格式导出**按钮。当用户选择此选项时，会弹出一个对话框，以允许用户选择要使用的 Excel 模板以及是要在 Excel 中打开它还是要将电子表格保存到他们的系统中。

有关“以 Excel 格式导出”这个高级功能的完整说明，请参阅第 215 页的『以 Excel 格式导出数据』。

XML 格式

对于独立的 DataBlox（即，未嵌入在另一个 Blox 内的 Blox），可以通过将 render URL 属性设置为 xml（render=xml）来以 XML 格式显示从应用程序数据源返回的查询结果集。第 227 页的『以 XML 格式导出』对 XML 格式的使用作了进一步的说明。

指定传递格式

缺省情况下，应用程序页面是以 DHTML 格式传递的。不需要执行任何特殊步骤就可以启用多种应用程序传递格式。添加到应用程序 URL 中的属性通知 DB2 Alphablox 以指定的格式传递页面。例如，以下 URL 请求以 DHTML 格式传递 MyApplication 页面:

```
http://<server>/applications/ThisView.jsp?render=dhtml
```

并且，如果要让页面以“打印机”格式显示，则 URL 将类似于:

```
http://<server>/applications/ThisView.jsp?render=printer
```

当将 render URL 属性与 DHTML 客户机配合使用时，有效值包括:

Render 属性
: **描述**

dhtml 缺省值。在客户机上使用 DHTML 技术来显示 Blox。

none 从页面中除去所有 Blox。

printer
: 以打印机就绪格式进行显示，不提供交互性

xls 以 HTML 格式显示页面并将页面的 MIME 类型设置为 application/vnd.ms-excel，从而导致以 Microsoft Excel 格式导出页面

xml 以 XML 格式显示（仅适用于显式 DataBlox）

打印 Blox 输出

可以使用两种方法（“打印机”格式和“PDF 报告”格式）来生成显示了 Blox 结果的可打印页面。“打印机”格式使用 PNG 图像和 HTML 表来在 Web 页面上显示 Blox 组件。并且，将任何 PageBlox 页面过滤器（独立的 PageBlox 页面过滤器或嵌套在 PresentBlox 中的 PageBlox 页面过滤器）由交互式 HTML 选择列表转换为静态维列表以及每个维的所选切片。

PDF 选项生成 PDF 文件，从而比组装人员和用户生成的可打印文件更灵活。要了解有关使用“PDF 报告”格式的详细信息，请参阅第 218 页的『以 PDF 格式导出』。

通过基于 HTML 的打印功能进行打印

虽然可以使用浏览器的“打印”按钮来打印 HTML 页面上的 Blox，但是，可以使用 DB2 Alphablox 来以打印机更易使用的格式显示 Blox 输出，如以下示例所述。显示的输出将显示在浏览器窗口中，从而将交互式 Blox 替换为它的可打印版本。

注: 缺省情况下，Microsoft Internet Explorer 浏览器不打印背景色和背景图像。用户如果要打印 Internet Explorer 中的背景色和图像，他就必须手工配置浏览器。因此，您应该假定大部分用户不会启用此设置。由于您无法控制此项浏览器设置，所以，您应该在假定大部分用户未修改此设置的情况下设计可打印页面 – 大部分用户既不了解此项设置也不知道在浏览器选项中的什么位置能够找到此项设置。

要设置此属性，请从浏览器的菜单栏中单击“查看”，然后单击“Internet 选项”。选择“高级”选项卡并向下滚动到“打印”部分。然后，选取“打印背景颜色和图像”选项并单击“应用”。

使用 render=printer URL 属性来创建可打印页面

显示可打印页面的最简单方法是使用 render=printer URL 属性。通过在页面 URL 末尾追加 ?render=printer，该页面上的 Blox 就将以打印机易使用的格式显示。如果该 Blox 是 PresentBlox，则可选择的页面过滤器将显示在 Blox 视图顶部的列表中，从而向查看者指示生成该可打印副本时哪些切片处于活动状态。通常，最好打开新的浏览器窗口来显示视图的可打印版本，而不是对现有页面应用 URL 属性。

执行下列步骤来使用此方法创建可打印页面:

1. 在有 Blox 的页面上，添加一个按钮或链接来生成一个可打印页面。例如，以下 HTML 代码在 JSP 页面的主体中创建一个标注为“打印预览”的按钮，该按钮将在新窗口打开当前页面的视图并以打印机方式显示该视图:

   ```
   <form>
   <input type="button" value="Print Preview"
   onclick="window.open('mView.jsp?render=printer','_new')">
   </form>
   ```

2. 现在，打开视图并测试该按钮。您应该会看到弹出一个新窗口，该窗口以可打印格式显示当前视图。

您还应该会注意到，此页面包括您添加的按钮。通常，这并不是您所期望的，因此，可以将公共服务按钮（包括“打印”按钮和“以 Excel 格式导出”按钮）放在框架集的一个框架中，并在另一个框架中显示分析视图。于是，这些按钮可以在一个独立的可打印页面中打开视图页面，并且该页面不显示按钮。

更好的方法是在独立的窗口中打开新页面。定制打印页面可能使可打印页面更合理。以下任务提供了一种生成定制打印页面的方法。

使用 <blox:display> 标记来创建定制打印页面

下列步骤显示如何创建一个页面，该页面使用 <blox:display> 标记来显示一个新页面，这个新页面显示同一个数据视图，但以可打印格式显示该视图:

1. 创建分析视图并添加一个按钮，该按钮将用来为此视图打开定制打印页面。在以下代码段中，创建了一个按钮，该按钮将打开一个新窗口，该窗口显示 MyView-print.jsp。

   ```
   <form>
   <input type="button" value="Print View"
   onclick="window.open('MyView-print.jsp','_new')">
   </form>
   ```

2. 现在，创建定制打印页面。除了包括用于显示视图的 <blox:display> 标记以外，您还可以考虑包括下列各项:

   - 所打印的视图的标题
   - 内容概述
   - 公司名或徽标图形
   - 页面的生成日期
   - 关于使用的警告（即，内部使用或机密）
   - 版权声明

   以下代码段是一个简单的打印页面示例，它将生成包含标题、以打印机方式显示的 Blox 以及打印日期的可打印页面:

   ```
   <h2>My Grid View</h2>
   <blox:display bloxRef="MyGridBlox2" render="printer"/>
   <p>
   Printed: <script>document.write(new Date());</script>
   </p>
   ```

3. 保存定制打印页面。

通过使用此方法，可以提供许多定制打印选项。

可以在“Blox 样本程序”的“显示数据”下面找到以上示例的有效版本。在生成的打印页面中，用户可以选择浏览器的“文件”菜单以打印该页面。

CSS 主题

DB2 Alphablox 使用级联样式表（CSS）主题来控制所显示的页面的布局和外观的各个方面。主题能够改善应用程序的外观。可以创建定制主题，这有助于贵公司在 DB2 Alphablox 应用程序中采用企业外观或者创建能够很好地与基于 Web 的现有应用程序或门户网站集成的外观。

指定主题

可以用 4 种方式来为 Blox 的外观指定主题:

- DB2 Alphablox 管理页面上的**缺省 HTML 客户机主题**设置为此 DB2 Alphablox 实例上的所有应用程序设置缺省主题。在初始安装之后，已选择了 coleman 主题。要在 DB2 Alphablox 管理页面上指定不同的系统缺省主题，请单击**管理**选项卡。在左边菜单中的**常规属性**部分下，请选择**系统**。**缺省 HTML 客房机主题**选项提供了一个下拉列表用于选择主题。它在系统级别上设置该主题。
- DB2 Alphablox 管理页面上每个应用程序的定义都包括了一个**缺省主题**设置。它在应用程序级别上设置主题，此设置覆盖系统级别设置。如果**缺省主题**设置为“default”，则将应用系统级别的缺省主题。
- <blox:header> 标记具有 theme 属性。它在页面级别上设置主题，此设置覆盖应用程序级别或系统级别上的主题设置。
- 当调用页面时，附加到 URL 的 theme URL 属性覆盖所有其他主题设置。

要将 theme 属性指定为 URL 属性，请使用以下格式:

```
http://.../application/view.jsp?theme=financial
```

当使用 URL 属性来定义主题时，定义的 theme 属性将应用于该页面上的所有 Blox。

CSS 主题文件

可以在 DB2 Alphablox 存储库的 theme 目录中找到 DB2 Alphablox CSS 主题的文件:

```
<alphabloxDirectory>/repository/theme
```

DB2 Alphablox 主题由以下结构组成:

<themeName>.properties
: 包含关于文件位置、布局和颜色的主题说明。<themeName>.properties 文件设置阐述如下。

<themeName>_dhtml.css
: 包含 DHTML 显示格式所特有的主题说明。

i
: 这个文件夹包含该主题的专用图像。

DB2 Alphablox 主题文件是根据它们所属的主题命名的（例如，financial.properties 和 financial_dhtml.css 是 finanical 主题选项的主题文件。这些文件位于一个主题目录中，该目录也是根据主题选项命名的。例如，financial 主题文件位于以下目录中:

```
<alphabloxDirectory>/repository/theme/financial
```

注: 要开发 portlet，请使用“门户网站主题实用程序”创建一个主题，该主题将把您选择的门户网站主题与 DB2 Alphablox 主题组合在一起。可以在 DB2 Alphablox 主页上的“管理”选项卡获得此工具。

在 themeName.properties 文件中定义的 CSS 主题属性

主题属性文件（<themeName>.properties）是一个纯文本文件，它定义了下列与 DHTML 相关的主题属性:

属性　描述

name = <*themeName*>
主题的名称（例如，coleman 或 financial）

description = *description*
显示在服务器控制台上的主题描述

bgcolor = #3d3d5f
<themeName>.css 中未指定的 ChartBlox 区域的背景色

fgcolor = white
themeName.css 中未指定的 ChartBlox 区域的前景（文本）色

css = <themeName>.css
themeName.css 文件的名称

background = true
导致显示 Blox 背景

privateimages = true
对于此主题使用的所有图像，使用此主题的专用图像目录（<repository>/theme/<*themeName*>/i/）

windowbgcolor = #655973
PresentBlox 中的图表显示区域的背景色

windowfgcolor = white
PresentBlox 中的图表显示区域的前景（文本）色

bkgrd_image_chart = *imageFile*
要显示在图表中的背景图像

chart_color_series
创建图表颜色（包括折线、条和饼图切片等的颜色）时使用的 18 种颜色，。

.css 文件中定义的 CSS 类

themeName.css 文件是一个纯文本文件，它定义下面描述的 CSS 类。要查看为特定主题中某个类设置的值，请打开该主题的 *themeName*_dhtml.css 文件，该文件位于:

<alphabloxRepository>/theme/<themeName>/*themeName*_dhtml.css

技巧: 可以在万维网联盟 Web 站点（http://www.w3c.org/style/css）上找到级联样式表规范。

下表列示了作用于整个应用程序的样式、旧样式以及 *themeName*_dhtml.css 文件中的覆盖:

作用于整个应用程序的样式

csApBg 应用程序背景色 – Blox 的整体背景

csCmpBg
组件背景色 – 各个 Blox 的数据区背景

csCmpBrdr
组件边框 – 各个 Blox 及控件的边框

csThmClr
基本主题颜色 – 由文本标注、信息文本和装饰元素使用

csFntClr
缺省字体颜色 – 由数据、消息和功能文本使用

csFntSpc
缺省字体规格 – 主要由菜单、工具栏和按钮使用

csLblFnt
缺省标注字体规格 – 由 GUI 交互式控件的标注使用

csGrdFnt
缺省网格字体规格 – 由所有网格单元格使用

csSlctBg
缺省选择背景色

csSlctClr
缺省选择字体颜色

csDsbldClr
缺省禁用字体颜色

csThrDBrdrRsd
凸出的 3 维边框

csThrDBrdrLwrd
凹下的 3 维边框

csBckClr
背景色（Blox 组件）

csWndwClr
“窗口”颜色（Blox 组件）

网格样式

csDmnsnHdrs
维头

csClmnHdrs
列头

csRwHdrs
行头

csRwHdrsNnBnd
行头 – 无色带

csRwHdrsBnd
行头 - 带色带

csDtCl 数据单元格

csDtClBnd
数据单元格 - 带色带

csDtClNnBnd
数据单元格 - 无色带

覆盖

样式类 描述

csRwHdrs
行头

csRwHdrsNnBnd
行头 - 无色带

csRwHdrsBnd
行头 - 带色带

csDtCl 数据单元格

csTdClBnd
数据单元格 - 带色带

csDtClNnBnd
数据单元格 - 无色带

覆盖已定义的样式

在 JSP 页面的 head 部分中添加 <blox:header> 标记的其中一个结果是提供指向适当 CSS 文件的自动链接。

注: 另请参阅有关对数据单元格和单元格提醒应用样式的信息，如第 153 页的『使用格式掩码来突出显示数据』和第 154 页的『使用单元格提醒来突出显示数据』 所述。

要覆盖已定义的样式，可以创建全新的主题:

1. 复制现有的主题目录并对其指定新名称。
2. 将该目录中的 *themeName*.css 和 *themeName*.properties 重命名。
3. 适当地更改这些文件的内容。
4. 在应用程序 URL 中，提供新主题的名称。对于名为 view.jsp 的页面，URL 可能变为:

   ```
   /<applicationName>/view.jsp?theme=MyTheme
   ```

注: 要装入新主题或重新装入经过修改的主题，请在 DB2 Alphablox 控制台中使用 load theme 命令。注意，此命令不接受特定主题名作为参数；此命令仅仅从存储库中装入所有主题。

注: Web 浏览器频繁地对文件（如按钮和图标 GIF 文件）进行高速缓存。当测试新主题或经过修改的主题时，可能需要确保清除浏览器高速缓存的高速缓存才能看到所作的更改。

对单元格提醒应用样式

单元格提醒样式不是在 CSS 文件中定义的，而是通过 DataBlox `cellAlert` 属性在 HTML 中以直接插入方式定义的。由于 CSS 具有级联效果，所以，直接插入的样式就是浏览器显示的样式，从而覆盖为数据单元格定义的任何其他样式。

用户界面外观

可以将 Blox 配置为显示不同的颜色方案、字体和色带特征，这可以使应用程序页面外观更加完美。正如可以使用级联样式表来控制 Web 页面上的字体颜色和背景色一样，也可以使用 Blox 属性和 HTML 主题来控制应用程序页面上的 Blox 外观。下列各节讨论常用的外观属性。

下表提供了经常修改的表示 Blox 外观特征的列表:

Blox　常用外观属性

GridBlox

- `missingValueString`: 确定显示什么内容来替换缺省的 `#MISSING`
- `rowHeadingsVisible`: 指定是否应该在网格中显示数据值左边的行标题
- `gridLinesVisible`: 显示或不显示网格线

ChartBlox

- `chartType`: 更改图表类型
- `depthRadius`: 可以用来对标准条形图创建轻微的 3 维效果
- 标注和位置

PresentBlox

- `dividerLocation`: 当启用了 splitPane 时，确定装入 PresentBlox 时应该将分割条显示在什么位置
- `splitPane`: 启用分割条，从而允许同时显示网格和图表
- `splitPaneOrientation`: 确定分割条是水平的还是垂直的

DataBlox

- `suppressMissing`: 不显示没有数据的行或列
- `suppressNoAccess`: 不显示用户对数据不具有访问权的行和列
- `suppressZeros`: 不显示包含全零的行或列

ToolbarBlox

- `removeButton`: 从工具栏中除去已定义的按钮
- 按钮颜色和大小

ReportBlox

- 请参阅*《关系报告开发者指南》*。可以修改大部分 HTML 元素的 CSS 样式设置。

要获取外观属性的完整列表，请参阅 *Developer's Reference* 一书中每个 Blox Tag Reference 章节的"by Category"一节中的外观属性和标记列表。

下列各节更详细地讨论 GridBlox、ChartBlox 和 PresentBlox 中使用的一些常用外观属性。

网格外观

网格是相当不错的信息来源，但是，有时难以阅读它们或难以注意到重要的信息。作为开发者，您可以更改网格的许多外观属性，但最经常更改的属性如下所示:

- `missingValueString`: 确定显示什么内容来替换缺省的 `#MISSING`
- `rowHeadingsVisible`: 指定是否应该在网格中显示数据值左边的行标题
- `gridLinesVisible`: 显示或不显示网格线

下列各节讨论了这些属性中的其中一些属性，并讨论了如何使用它们来创建更有用的网格。

行色带

在包含许多行的网格中，缺省情况下启用了行色带。如果有必要的话，为了美观或为了避免类具有已选择的单元格提醒颜色，可以通过修改 CSS 主题属性来更改行色带所使用的背景色和前景色。要了解有关 CSS 主题的信息，包括可修改的 CSS 主题列表，请参阅 第 142 页的『CSS 主题』。

单元格外观

可以将显示结果集的网格数据单元格定制为使用不同的前景色和背景色以及字体来进行显示。要控制这些选项，请使用 CSS 主题。要了解有关 CSS 主题的信息，包括可修改的 CSS 主题列表，请参阅 第 138 页的『打印机格式（render=printer）』。

图表外观

可以通过许多方法来定制 DB2 Alphablox 应用程序中的图表外观以满足用户的特定需求。这里讨论了几个经常更改的 ChartBlox 属性，即 `chartType`、`depthRadius` 和 `chart_color_series`。

图表类型

最经常更改的 ChartBlox 属性是 `chartType`，其缺省值为 `Vertical Bar, Side-by-Side, 3D Effect`。ChartBlox 具有许多其他图表类型，它可以满足几乎每个用户的需求，但是，最常使用的四种图表类型是 `Bar`、`Line`、`3D Bar` 和 `Pie`。在 `chartType` 属性设置中，可以使用这些短名称，也可以使用它们的完整图表类型名。要获取所有受支持图表类型的有效名称的完整列表，请参阅 *Developer's Reference* 一书的 ChartBlox Tag Reference 一节。下表列示了 4 个可用的图表缩写名及其完整的图表类型名:

缩写名 完整的图表类型名

`Bar` Vertical Bar, Side-by-Side

`3D Bar` 3D Bar

`Line` Vertical Line, Absolute

Pie Pie

对图表添加 3 维外观

ChartBlox 的缺省 chartType 属性是 Vertical Bar，Side-by-Side，3D Effect，这将显示具有轻微 3 维效果的 2 维条形图。标准条形图（Bar 或 Vertical Bar，Side-by-Side）或折线图（Line 或 Vertical Line，Absolute）是平面的并且是 2 维的，但是，通过设置 depthRadius 属性，可以对这些图表添加您自己选择的轻微 3 维效果，从而使它们具有另一种外观。要对平面的条形图和折线图添加一点深度，可以组合使用 depthRadius 属性和 chartType 属性，如以下示例所示:

```
<blox:chart
...
chartType="Bar"
depthRadius="5"
... />
```

depthRadius 可以接受介于 0 与 100 之间的整数。depthRadius 设置还将影响其他 2 维图表的外观。

图表颜色

可以根据所使用的 CSS 主题来设置用于创建折线、条和饼图切片的图表颜色。要指定与特定主题所使用的缺省颜色不同的图表颜色，您可以定义自己的图表颜色系列（由 18 种不同的颜色组成）。要指定另一颜色系列以供主题使用，请打开 *themeName*.properties 文件（例如 coleman.properties），此文件位于以下目录中:

<alphabloxRepository>/theme/*themeName*/

找到 chart_color_series 属性并重新定义用来定义颜色的 18 种颜色。图表颜色是使用十六进制代码（例如 #E0CB68）指定的，这与 Web 页面中常用的值相同。

PresentBlox 外观

以下主题描述了几种更改 PresentBlox 外观的简易方式。

分割窗格

缺省情况下，PresentBlox 实例显示两个窗格，一个带有网格，另一个带有图表。splitPane 属性（缺省情况下，它设置为 true）允许用户同时以表格表示法和图形表示法查看数据。并且，当用户与一个窗格中的数据进行交互时，另一个窗格将同时进行更新以反映那些更改。例如，当用户对图表中的数据进行下寻时，网格窗格将反映新的结果集。

缺省情况下，当分割窗格可用时，分割条设置为 vertical。当图形 Y 轴上显示的项较多时，您可能想考虑将 dividerLocation 更改为 horizontal，以使显示的网格和图表的宽度都与 PresentBlox 相同。通常，在使用水平设置时，您可能会发现，图表显示在网格上方时外观较好。可以通过将 chartFirst 属性设置为 true（从而覆盖缺省值）来指定这种显示方式。

由于缺少可用的屏幕空间，或者由于初始结果集的数据密度导致初始图形不可读，您可能会认为将 splitPane 设置为 false 结果更好。虽然这是一个合理的选择，但是，

您可能想考虑将分隔窗格选项保留给用户使用（注意，在任何可用的工具栏选项中都无法更改这个特定选项），此时，您可以更改分割窗格分割条的初始显示位置，将其放在一边但保持其可用。

dividerLocation 属性允许您设置分割条的初始位置。可以接受的值的范围是 0 到 1，其中，值 0 表示应该显示右边的（或底部的，这取决于 splitPaneOrientation 的设置）内容，值 1 表示只应该显示左边的（或顶部的）内容。请尝试几个不同的设置以了解将该设置更改为除缺省值 0.5 以外的值是否有意义。

要了解有关 splitPane、splitPaneOrientation、dividerLocation 和 chartFirst 属性的完整信息，请参阅 *Developer's Reference* 一书中的 PresentBlox Tag Reference 一节。

修改 DataLayout 属性

缺省情况下，DataLayoutBlox（即 DataLayout 面板）是可用的，但最终用户看不到它。对于主要由高级用户使用的分析视图，您可能会决定要在 PresentBlox 装入时显示 DataLayout 面板。要在 PresentBlox 装入时显示 DataLayout 面板，需要将嵌套的 DataLayoutBlox 的 visible 属性设置为 true，如以下示例所示:

```
<blox:present id="myPresentBlox">
...
<blox:dataLayout visible="true"/>
...
</blox:present>
```

如果您不想将 DataLayout 面板提供给用户使用，例如，如果您只希望普通用户查看和使用 PresentBlox，则可以通过将 PresentBlox dataLayoutAvailable 属性设置为 false 来禁用 DataLayout 面板。这将自动地导致 DataLayout 按钮不显示在工具栏上。以下代码段显示了此属性的正确用法:

```
<blox:present id="myPresentBlox"
dataLayoutAvailable="false">
...
</blox:present>
```

DataLayout 面板的可用性是由 PresentBlox dataLayoutAvailable 属性设置确定的，并且，该属性设置是 <blox:present> 标记的一个属性。DataLayout 面板的可视性是 DataLayoutBlox 对象本身的一个属性，因此，您使用 <blox:dataLayout> 标记的 visible 属性来控制它。

修改菜单栏属性

菜单栏是基于文本的菜单，它显示在 Blox 顶部，并根据该 Blox 是 GridBlox、是 ChartBlox 还是 PresentBlox 来自动地与相关菜单合并。缺省情况下，<blox:grid>、<blox:chart> 和 <blox:present> 的 menubarVisible 标记属性设置为 true。要从这些 Blox 中的其中一个 Blox 中除去菜单栏，请对 Blox 标记添加 menubarVisible 标记属性并将值设置为 false。

除了显示或不显示菜单栏以外，没有任何标记或标记属性可以控制菜单栏外观。高级开发者可以使用 Blox UI 模型的可扩展性来定制独特的菜单栏。

修改工具栏属性

缺省情况下，在 PresentBlox 上为用户提供了工具栏。开发者经常会修改几个公共外观设置。工具栏被认为是始终可用的，但是，可以使用 ToolbarBlox 标记的 visible 属性将其设置为对用户不可视。

即使您决定保持工具栏可视，也仍然可以选择禁用（即除去）某些工具栏按钮。要从工具栏中除去按钮，请使用嵌套的 ToolbarBlox 的 removeButtons 属性来除去您确定不想要的按钮。例如，以下示例显示如何从 ToolbarBlox 中除去 Help:

```
<blox:present ...>
<blox:toolbar removeButtons="Load,Save,Help"/>
</blox:present>
```

注: 注意，您必须记得对 removeButtons 标记属性添加 Load 和 Save，这是因为它们包括在此属性的缺省值字符串中。如果您忘记将它们添加到字符串中，则 Load 和 Save 按钮将显示在工具栏中。

数据外观

当从数据源中检索结果集时，可能已预先对返回的数据进行了格式化，也可能未进行格式化。一旦将结果检索到 DB2 Alphablox 应用程序中，就可以使用若干个选项来控制数据的外观和格式化。通过使用 DataBlox 属性，可以消去全零的行或列、缺少数据（或包含空值）的行或列、包含重复数据的行或列或者包含用户无权访问的数据的行或列。在 GridBlox 中，可以对数值进行格式化，包括符号（$ 和 % 等等）或分组（显示逗号或句点）。并且，在 GridBlox 中，可以采用千位分组、百万位分组或适合于数据的分组格式来显示数值。

GridBlox 属性

有三个 GridBlox 属性对于更改数据值的格式化（并因而更改网格中的数据外观）来说非常有用，这三个属性是 defaultCellFormat、cellFormat 和 formatMask。在下列任务中，您将学习如何使用这些格式掩码属性来完成一些常见任务。要了解有关这些属性的完整用法详细信息，请参阅 GridBlox Tag Reference 一书。

以千位分组和十亿位分组格式来格式化值

最终用户的一项常见请求是通过消除无关数字（在本质上，这是进行四舍五入）来摆脱不必要的数据干扰。如果一个用户正在处理涉及数以万计美元的预算，则查看美分甚至美元可能没有什么用，并且将毫无必要地占用数据单元格中的空间。作为开发者，您可以通过更改 GridBlox 的 defaultCellFormat 属性来为此用户提供帮助。例如，要以千位分组格式显示所有网格值，您有两个选项。要以千位分组格式显示值，并且在值后面显示“K”以表示以千计，则输入以下设置:

```
defaultCellFormat="#,###K"
```

值中的 K 告诉 DB2 Alphablox 要以千位分组格式显示数值，但在值的末尾追加 K。

虽然 K 指示值是以千位分组格式显示的，但是，许多用户不希望在每个字段中都看到 K。要以千位分组格式显示网格值，但不显示 K 后缀，可以通过输入以下设置来使用以千位分组格式计算数值的功能，此功能适用于 DB2 Alphablox 应用程序中的所有格式掩码:

```
defaultCellFormat="#,###/1000"
```

在某些情况下，您可能想在末尾显示一个特殊字符，例如，显示“B”表示十亿。要将“B”添加至此值的末尾，请将以上设置修改成:

```
defaultCellFormat="#,###/1000'B'"
```

要了解有关 defaultCellFormat 和其他格式掩码属性的用法详细信息，请参阅 GridBlox Tag Reference 一书。

显示特定成员的百分比

当使用 GridBlox defaultCellFormat 属性时，整个值网格都将受到影响。通常，您想要将值的格式化限制到特定的行或列，或者对除了特定成员的数据值以外的所有数据值使用 defaultCellFormat 属性。要将数值格式化限制到特定的成员，从而只影响一行或一列，可以使用 cellFormat 属性。与 defaultCellFormat 属性不同，cellFormat 是带下标的属性，它要求使用它自己的 Blox 标记。由于它是一个单独的标记，并且表示的是 GridBlox 属性，所以，cellFormat 标记必须嵌套在 GridBlox 标记的主体中。以下代码段显示了当您只想将 Variance % 值显示为在每个成员值后面显示百分号（%）时非嵌套 GridBlox 中的 cellFormat 标记:

```
<blox:grid id="myGridBlox">
<blox:cellFormat
format="#,###.00%"
scope="{Scenario: Variance %}" />
</blox:grid>
```

在此示例中，Variance % 以百分比格式显示值，并且显示两个小数位。

控制小数外观

虽然小数对数据外观的影响可能不会立即体现出来，但是，这里提供了数据显示导致网格可读性降低的示例。注意，在以下网格中，出现在排列中的各行中带有任意数值位的数值未对齐。

7.654
3.21
43.21
543.2
3
3.2

正如您注意到的那样，即使在这个很小的数值样本中，由于您必须在心里尝试根据小数点位置来将数值对齐，所以难以对值进行比较。可以定义格式掩码以使这些值中的所有小数按列对齐。例如，可以使用 <blox:cellFormat> 标记来使 Variance % 按列对齐，如下所示:

```
<blox:cellFormat
format="#,###.000"
scope="{Scenario:Variance %}"/>
```

Variance % 列现在将把数据显示成:

7.654
3.210
43.210
543.200
3.000
3.200

正如您在本示例中学到的，适当的格式掩码将使数据更具可读性、更有意义并且外观更佳。还可以使用格式掩码来将负数值显示在圆括号中或者显示为红色、显示货币符号以及显示百分比符号。请参阅 *Developer's Reference* 一书中的 GridBlox Tag Reference 一节以了解其他格式掩码选项。

第 16 章 突出显示信息以及对信息添加注释

您如何使别人对某个重要方面与其余数据有所不同的信息加以注意？这常常被称为“异常报告”或“信号灯”，其共同目标是提醒用户对某些信息加以注意，那些信息对于决策来说可能非常重要。本主题讨论使用 DB2 Alphablox 单元格提醒和信息链接来解决此问题。单元格提醒可用来通过更改数据单元格样式来突出显示信息，并且可以用来根据某些条件显示链接。信息链接（包括头链接和单元格链接）也可以提供一种方法来突出显示单元格以便向用户提供更多信息。

用户对网格中特定数据添加和查看注释的能力是另一种功能强大的信息突出显示方法。本节中的一个主题阐述了如何使用 CommentsBlox 来在应用程序中添加此功能。

概述

在访问公司数据库中存储的大量信息的同时，也会产生如何帮助用户快速查找重要信息的问题。DB2 Alphablox 开发者可以对网格使用诸如单元格提醒和超链接之类的技术来根据某些业务条件突出显示信息或者将用户链接至更多与他们使用的应用程序相关的信息。在下列各节中，您将了解如何使用单元格提醒、单元格链接和单元格提醒链接来引起对重要信息或辅助信息的注意。

用户对网格中特定数据添加和查看注释的能力是另一种功能强大的信息突出显示方法。在本主题中，您还将了解如何使用户能够对多维数据库中的数据单元格添加注释以及显示那些注释。

使用格式掩码来突出显示数据

缺省情况下，网格中的负数值在值的前面显示减号。此外，可以使用 defaultCellFormat 或其他格式掩码属性来将负数值显示为括在圆括号中或者显示为红色。与 Microsoft Excel 相同，在 DB2 Alphablox 应用程序中可以使用格式掩码来将负数值显示为红色。

以红色突出显示负数值

要将网格中的所有负数值突出显示为红色，请使用其中一个将把负数值显示为红色的格式掩码来设置 defaultCellFormat。所有值都将根据 cellStyle 中的值进行格式化，缺省情况下，它将所有值都显示为黑色。

在以下示例中，正数值将把所有值都显示为具有两个小数位并使用逗号来将分组隔开。所有负数值（由分号右边的格式掩码指示）都将使用与正数值相同的格式显示，但是这些值将是红色的:

```
<blox:grid ...
defaultCellFormat="#,###.00;[red]#,###.00"
</blox:grid>
```

如果您只想将特定成员的负数值显示为红色，则需要使用 cellFormat 属性。在以下示例中，成员 Actual 的负数值将显示为红色:

```
<blox:grid ...>
<blox:cellFormat
format="#,###.00;[red]#,###.00"
scope="{Scenario:Actual}"/>
</blox:grid>
```

要了解有关 defaultCellFormat 和 cellFormat 属性的用法详细信息，请参阅 GridBlox Tag Reference 一书。

用圆括号突出显示负数值

另一种方法是将负数值显示为括在圆括号内（但不显示减号），这种做法在财务领域很常见。尽管这是一种常见的数字格式化做法，但可能也有助于引起对网格中的负数值加以注意。在以下示例中，负数值将括在圆括号内:

```
<blox:grid ...
defaultCellFormat="#,###.00;(#,###.00)"
</blox:grid>
```

如果您愿意的话，可以将这两种突出显示方法配合使用。下列设置将导致负数值显示在圆括号内并显示为红色:

```
<blox:grid ...
defaultCellFormat="#,###.00;[red](#,###.00)"
</blox:grid>
```

要了解有关 defaultCellFormat 和 cellFormat 属性的用法详细信息，请参阅 *Developer's Reference* 一书的 GridBlox 一节。

使用单元格提醒来突出显示数据

如果不仔细审阅大型网格中的所有数值以找出证明需要进一步加以注意的偏差或趋势，信息分析工作就难以进行。如果分析员遗漏了哪怕是一个值的重要偏差，公司就可能会付出高昂的代价。一寸光阴一寸金，任何能够加快分析员工作速度并帮助他们不会遗漏重要更改的措施都会大大提高效率。许多分析应用程序（包括 Flash 报告和执行记分卡）使用异常报告（即信号灯）来指示需要对某些数据加以注意。DB2 Alphablox 提供了可以用来突出显示此类关键信息的 Blox 属性和方法。

在 DB2 Alphablox 应用程序中，可使用 GridBlox cellAlert 属性来突出显示用户可能希望得到提醒的重要信息:

- 与期望值的重大偏差
- 负数值（指示潜在的盈利能力问题）
- 超出可接受范围的比率

下面讨论了两种使用 cellAlert 属性来突出显示信息的方法: 单元格格式化和单元格提醒链接。

单元格格式

根据您正在开发的应用程序的不同，用户可能会要求根据某些业务逻辑来对值设置“信号灯”。信号灯是一个常用的术语，它用来描述一种异常报告形式，在这种异常报告形式下，使用红色、黄色和绿色等您在驾车时遇到的真实信号灯信号来突出显示不同的值范围。信号灯是一种易于理解的技术，它是将生活中一个领域的知识（驾车或在街

上散步）应用到另一个领域（商业智能）。信号灯在分析应用程序中的典型应用是，根据通常以三种标准信号灯颜色反映的条件来对数据值的背景进行突出显示。下表列示了三种标准信号灯颜色并描述了它们的常用含义:

背景色 描述

红色 危险级别，用户应该对这些值特别关注。

黄色 值不在可接受的或期望的范围内，可能值得加以注意。

绿色 值是可接受的并且处于“安全的”或“良好的”范围内，不需要加以注意。

以下任务说明了如何创建简单的信号灯报告系统来提醒用户对重要的数据更改加以注意。

简单的信号灯报告系统

在创建用于报告的信号灯通知系统时，有许多可能的变化。在本主题的先前内容中，您了解了如何使用 GridBlox `defaultCellFormat` 或 `cellFormat` 属性来在网格中突出显示负数值。虽然这在许多情况下是一个不错的解决方案，但是，在有些情况下，负数值实际上是“好的”值，因此使用格式掩码来以红色突出显示它们可能不起作用。GridBlox `cellAlert` 属性允许按下列各项定制提醒:

- 单元格背景色
- 数据值样式，包括字体和颜色
- 当条件符合时显示的单元格链接

要了解有关可以与 `cellAlert` 属性配合使用的所有属性的完整详细信息，请参阅 GridBlox Tag Reference 一书。执行下列步骤来为网格上的成员创建一个简单的信号灯系统:

1. 选取要对其突出显示值范围的成员。

   在本示例中，对 Variance % 成员应用信号灯。虽然在网格中将显示四个列（Actual、Forecast、Variance 和 Variance %），但只有 Variance % 成员将显示指示关注度级别的背景色。

2. 添加 `<blox:cellAlert>` 标记以定义应该显示为具有红色背景的值。

```
  <blox:cellAlert
scope="{Scenario:Variance %}"
condition="LT"
value="0"
background="red">
</blox:cellAlert>
```

3. 添加 `<blox:cellAlert>` 标记以定义应该显示为具有黄色背景的值。

```
  <blox:cellAlert
scope="{Scenario:Variance %}"
condition="between"
value="0"
value2="10"
background="yellow">
</blox:cellAlert>
```

4. 添加 `<blox:cellAlert>` 标记以定义应该显示为具有绿色背景的值。

```
  <blox:cellAlert
scope="{Scenario:Variance %}"
condition="GT"
value="10"
background="green">
</blox:cellAlert>
```

5. 运行报告。

示例: 要查看此报告子系统的有效示例，请参阅“Blox 样本程序”的“突出显示数据”部分中的“信号灯”示例。

当使用信号灯和异常报告时，您要记住下列几个要点:

- 确保范围覆盖所有的值。例如，如果将大于零的值显示为绿色，并将小于零的值显示为黄色，则当值为零时，不会进行突出显示。
- 对于首项和末项，除非对值的范围有固定的限制，否则请考虑对范围一端使用 GT 或 GTEQ，并对值范围的另一端使用 LT 或 LTEQ。这能够避免将来当值超出预定义范围时必须执行额外的维护工作。
- 行色带、代样式和单元格样式所使用的颜色方案可能会干扰单元格提醒，从而导致重要的提醒被遗漏。例如，如果缺省行色带显示隔行黄色背景，并且提醒也使用黄色作为背景色，则用户很可能会遗漏该提醒。
- 提醒所使用的颜色方案会影响打印页面的效果。根据打印机的不同，具有红色背景的单元格在打印时可能会变为黑色或变得非常暗，从而盖住了那些单元格中的值。
- 具有提醒背景色的单元格可能会由于值颜色与背景色之间对比不强烈而导致联机可读性降低。具有红色背景的黑色值尤其难以阅读。并且，请记住，颜色及颜色对比变化是随监视器的不同而有所变化的。

单元格提醒链接

除了由于与单元格提醒条件相符而更改单元格外观以外，您还可以定义在某些条件符合时显示的链接。例如，当特定的值符合条件时，您可能想弹出一个文本窗口以指出有关该值的一些内容以及单元格提醒在此情况下为何适用。

为单元格提醒创建提醒消息

在本任务中，我们的目标是在用户单击单元格的链接图标时弹出一个文本窗口。要执行的步骤如下所示:

1. 创建要为提醒弹出的窗口。

   例如，您的窗口可能是带有一条简短消息、不带导航元素并带有一个“关闭窗口”按钮的简单 HTML 窗口。它可能类似于:

```
<html>
<head>
<title>Alert Message</title>
</head>
<body>
Your alert message here
 </body>
</html>
```

   在本示例中，假定将该文件保存为 alertMessage.html。

2. 在 GridBlox 标记内嵌套 GridBlox cellAlert 标记。

```
<blox:grid id="myGridBlox"
...>
  <blox:cellAlert
condition="LT"
value="0"
link="http://<serverName>/<appName>/notes/alertMessage.html"/>
</blox:grid>
```

注: 在 cellAlert 标记中使用的链接应该是绝对 URL（显示了页面的完整路径），也可以是从应用程序上下文开始的相对 URL（包括最前面的正斜杠）。当该链接是绝对 URL 时，它必须在 URL 中包括服务器名。下列两个示例都有效:

```
link="http://<serverName>/<appName>/notes/alertMessage.html"
link="/notes/alertMessage.html"
```

在以上 URL 中，serverName 必须是正在运行 DB2 Alphablox 应用程序的服务器。

3. 测试应用程序页面。

注: 在将单元格提醒与单元格链接一起使用时，您要了解，如果两者都包含图像，则单元格提醒优先于单元格链接。如果将单元格提醒应用于数据单元格，并且还有定义了图像或图像对齐的已定义单元格链接，则单元格链接不显示。但是，如果有未包括链接或图像的单元格提醒，则两者都将被应用。

使用图表系列颜色来突出显示数据

在 DB2 Alphablox 应用程序中，除了使用 GridBlox cellAlert 属性来突出显示网格中的数据之外，您还可以在图表中指定不同的数据系列颜色以表示需要引起注意的数据。例如，如果数据值下降到指定的阈值之下，则您可以将条的颜色设置为红色。

在图表中创建信号灯效果涉及到两个常规任务:

- 设置数据系列颜色。
- 设置每个图注项的颜色和显示标注以说明颜色所表示的含义。

设置数据系列颜色

图表中的每个数据系列包含许多数据点且表现为单个图注项（饼图除外）。所有数据系列扩展 com.alphablox.blox.uimodel.core.chart 包中的 AbstractDataSeries 类。AbstractDataSeries 提供用于设置和获取数据系列名称的基本功能、存储所选数据点的状态并针对最终用户数据点选择事件执行一些基本事件处理。SingleValueDataSeries 对象扩展 AbstractDataSeries，对于每个数据点（如 BarChart、LineChart 或 PieChart），它是是包含单值的所有数据系列的超类。

可以使用 ChartBlox setSeriesColorList() 方法来设置数据系列的颜色。然而，此方法允许您根据系列的顺序来设置颜色，以便您定制这些颜色以符合应用程序的主题或颜色模式。要基于数据值设置颜色，应使用 AbstractDataSeries 或 SingleValueDataSeries 的 setColor() 方法。

设置图注项的颜色和显示标注

com.alphablox.blox.uimodel.core.chart.common 包中的 Legend 类允许您处理图表图注的可视特征。缺省情况下，图注项的文本来自所绘制的数据系列的名称。例如，如果条形图上绘制了 4 个条，则每个条都是一个 BarDataSeries 且具有系列名称。此系列名称

将显示为图注项的名称。Legend 对象使您可以访问并控制每个 LegendItem，从而允许您对图注中显示的项、应该使用的填充颜色或模式以及显示标注的内容进行特定的控制。

对于信号灯图表，您通常要设置每个图注项的颜色和显示文本。例如，通过 LegendItem 的 `setFillColor()` 和 `setText()` 方法，您可以指定：绿条以指示达到或超过销售目标的数据系列；黄条以指示有可能不满足季度目标的数据系列。

要获取信号灯图表的完整示例，请参阅『信号灯图表示例』。

信号灯图表示例

本示例通过以下操作来演示简单的信号灯条形图（`chartType = "Bar"`）：

- 如果数据值小于 3000000，则将条的颜色设置为红色。
- 如果数据值小于 6000000，则将条的颜色设置为绿色。
- 如果数据值不满足以上任一条件，则将条的颜色设置为黄色。
- 设置图表中图注项的颜色和显示文本以解释颜色的含义。

本示例中演示了几个技巧：

- 我们使用 PresentBlox，而禁用它的嵌套 GridBlox、DataLayoutBlox 和 PageBlox。工具栏和菜单栏的可视性为 `false`，因此整个显示区域的 300 x 300 个像素全部被图表占用。
- 为图注上的 DoubleClickEvent 创建了一个定制控制器 MyController。当用户双击某个图注项时，将在新的浏览器窗口中装入特定于该图注项的 URL。
- 条形图中的数据系列为 SingleValueDataSeries 对象。要获取条形图中的所有数据系列：

  ```
  BarChart bChart = (BarChart) chart;
  SingleValueDataSeries series[] = bChart.getAllDataSeries();
  ```

  然后我们对数据系列进行迭代并基于数据值设置条的颜色。
- 每个数据系列的 LegendItem 都已修改以描述颜色所表示的含义：

  ```
  LegendItem items[] = new LegendItem[3];
  items[0] = new LegendItem();
  items[0].setFillColor(Color.green);
  items[0].setSymbolType(LegendItem.SHAPE_BAR);
  items[0].setText("Meeting the quarterly goal");
  items[0].setEventIndex(0);
  ```

有关 SingleValueDataSeries 和 LegendItem 对象的更多信息，请参阅第 157 页的『使用图表系列颜色来突出显示数据』。

完整的示例如下：

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="com.alphablox.blox.uimodel.BloxModel,
com.alphablox.blox.uimodel.ChartBrixController,
com.alphablox.blox.uimodel.ChartBrixModel,
com.alphablox.blox.uimodel.ModelConstants,
com.alphablox.blox.uimodel.core.Component,
com.alphablox.blox.uimodel.core.Controller,
com.alphablox.blox.uimodel.core.chart.BarChart,
com.alphablox.blox.uimodel.core.chart.Chart,
com.alphablox.blox.uimodel.core.chart.SingleValueDataSeries,
com.alphablox.blox.uimodel.core.chart.common.ChartComponent,
com.alphablox.blox.uimodel.core.chart.common.Legend,
com.alphablox.blox.uimodel.core.chart.common.LegendItem,
```

```
com.alphablox.blox.uimodel.core.event.ComponentRebuiltNotify,
com.alphablox.blox.uimodel.core.event.DoubleClickEvent,
com.alphablox.blox.uimodel.core.event.IEventHandler"%>
<%@ page import="java.awt.*"%>
<%@ page import="java.util.ArrayList"%>
<%@ taglib uri="bloxtld" prefix="blox" %>
<html>
<head>
<blox:header/>
</head>
<%
class MyController extends Controller{
public boolean handleDoubleClickEvent(DoubleClickEvent event) throws Exception {
  Chart theChart = (Chart) event.getComponent();
ChartComponent chartComponent = theChart.getSelectedChartComponent();
if (chartComponent instanceof Legend){
  Legend legend = (Legend) chartComponent;
int selected =legend.getSelectedIndex();
String color = ((String[])legend.getUserObject())[0];
String url = "http://webexhibits.org/pigments/indiv/color/"+ color + ".html";
theChart.getDispatcher().sendClientCommand("window.open('"+url+"',
'"+color+"')");
return true;
}
return false;
}
}
%>
<body>
<blox:present id="TrafficLight"
gridAvailable="false"
dataLayoutAvailable="false"
pageAvailable="false"
menubarVisible="false"
toolbarVisible="false"
width="300"
height="300">
<blox:dataLayout visible="false" />
<blox:chart
chartType="Bar"
legendPosition="bottom"/>
<blox:data
dataSourceName="QCC-Essbase"
query="<COLUMN (\"All Time Periods\") (\"All Time Periods\")
<ICHILD \"2001\" <ROW (\"Measures\") Sales !"
parentFirst="true"
useAliases="true" />
<%
BloxModel model = TrafficLight.getBloxModel();
model.addEventHandler( new IEventHandler() {
public boolean handleComponentRebuiltNotify(ComponentRebuiltNotify event)
throws Exception {
Component component = event.getComponent();
if (component instanceof ChartBrixModel) {
Chart chart=(Chart)((ChartBrixModel) component).searchForComponent(
ModelConstants.CHART);
if (chart instanceof BarChart) {
//Gets the data series in the bar chart
BarChart bChart = (BarChart) chart;
SingleValueDataSeries series[] = bChart.getAllDataSeries();
for (int i = 0; i < series.length; i++) {
for (int j = 0; j < series[i].size(); j++) {
Number val = series[i].get(j);
if (val != null && val.doubleValue() < 3000000){
series[i].setColor(j, Color.red);
}
else if (val != null && val.doubleValue() > 6000000){
```

```
series[i].setColor(j, Color.green);
}
else {
series[i].setColor(j, Color.yellow);
}
}
}
}
LegendItem items[] = new LegendItem[3];
items[0] = new LegendItem();
items[0].setFillColor(Color.green);
items[0].setSymbolType(LegendItem.SHAPE_BAR);
items[0].setText("Meeting the goal");
items[0].setEventIndex(0);
items[1] = new LegendItem();
items[1].setFillColor(Color.yellow);
items[1].setSymbolType(LegendItem.SHAPE_BAR);
items[1].setText("Potential problem areas");
items[1].setEventIndex(1);
items[2] = new LegendItem();
items[2].setFillColor(Color.red);
items[2].setSymbolType(LegendItem.SHAPE_BAR);
items[2].setText("Not meeting the goal");
items[2].setEventIndex(2);
chart.setController(new MyController());
Legend legend = chart.getLegend();
legend.setLegendItems(items);
String desc[] = new String[]{"greens", "yellows", "reds"};
legend.setUserObject(desc);
chart.changed();
component.changed();
}
      return false;
}
}
);
((ChartBrixController)(model.getChart().getController())).synchronize();
%>
</blox:present>
</body>
</html>
```

本示例演示了创建信号灯图表的基本方法和途径。因为在用户执行诸如深入钻取等数据导航操作时，指定的颜色和阈值将继续应用于图表数据，所以颜色可能会变得令人误解。可能还需要其他定制代码来控制数据导航操作、复位数据系列颜色或更改阈值。另一个选项是将 Blox 用户界面设置为是不可单击的（通过使用 `<bloxui:component>` 标记的 `clickable` 属性）。

信息链接

在分析应用程序中，有时您可能需要包括链接以指向有关网格数据的更多信息。链接可以有许多不同的用途，包括:

- 链接至有关标题的更多信息
- 链接至有关特定单元格的信息
- 根据业务逻辑链接至单元格的提醒信息

DB2 Alphablox 提供了三种类型的信息链接: 头链接、单元格链接和单元格提醒链接。下面列示了这些链接类型中的每种链接类型的优点和缺点。

链接类型

用途概述

头链接

- 当维成员显示在行头或列头中时，这些链接将显示在维成员右边
- 这些链接是在 DB2 Alphablox 管理页面的应用程序定义页面中定义的
- 当成员具有相关联的链接时，这些链接始终可视
- 所有头链接共用一个图标图像

单元格链接

- 可以将链接定义为具有作用域。
- 这些链接是使用 GridBlox `cellLink` 属性定义的
- 可以根据 `cellLink` 定义不同的图像
- 可以导致打开信息窗口，也可以触发 JavaScript 函数执行

单元格提醒链接

- 此链接是作为 `cellAlert` 属性的一部分定义的
- 可以与单元格链接一起使用，但是，如果两者都包含图像，则单元格提醒链接具有优先权
- 可以用来根据条件逻辑或作用域来显示
- 可以导致打开信息窗口，也可以触发 JavaScript 函数执行

请参阅下面每种信息链接类型的用法详细信息。

使用头链接

头链接是特定于应用程序的信息链接，您可以定义头链接以便在用户单击信息图标（此图标显示为蓝色的圆，中间有一个白色的“i”）时显示 Web 页面或触发 JavaScript 函数，该信息图表显示在网格中的行成员或列成员旁边。只有应用程序定义页面的“头链接”文本框中定义的成员的头中才会显示头链接。

要为特定应用程序添加头链接，请在 DB2 Alphablox 主页中打开应用程序定义页面。在页面底部附近有一个“头链接”文本框，在该文本框中，可以使用以下语法来定义头链接:

```
memberName = URL
```

其中，`memberName` 是数据源中定义的唯一成员名，*URL* 是绝对 URL（显示了页面的完整路径）或从应用程序上下文开始的相对 URL（包括最前面的正斜杠）。当该链接是绝对 URL 时，它必须在 URL 中包括服务器名。例如，要创建指向 Diet Cola 产品页面的信息链接，可以使用以下头链接定义:

```
Diet Cola = http://productServer/products/dietcola.html
```

或者

```
Diet Cola = /<pathTo>/dietcola.html
```

注: 不支持 JavaScript 协议方法。

使用单元格链接

就象可以使用头链接来在行头和列头中放置链接一样，可以使用 GridBlox 的 cellLink 属性来对数据单元格定义超链接。与头链接不同，还可以使用单元格链接来通过使用 JavaScript 协议方法调用 JavaScript 方法。与其他具有下标的属性相同，单元格链接是根据它们的下标值被求值的，那些下标值是在运行时动态生成的，或者是由开发者定义的。

单元格链接的编号指定了它们的求值顺序，此顺序以下标值为 1 的 cellLink 开始。第一个与单元格的条件和作用域相匹配的已定义单元格链接就是唯一适用于该单元格的链接。在定义单元格链接时，务必考虑可能的重叠。并且，单元格提醒链接优先于使用 cellLink 创建的链接。也就是说，如果为特定数据单元格定义了单元格提醒链接和单元格链接，则单元格提醒链接将显示在该单元格中，但单元格链接不显示。

在同一个数据单元格中同时显示单元格提醒和单元格链接是有可能的，但是，如果两者都定义了图像元素（image、image_align 或 link），则单元格提醒链接将优先于单元格链接 - 在一个单元格上只能提供一个图标和链接，并且假定（带有链接的）单元格提醒比单元格链接更重要。

以下是单元格链接属性标记的一个示例，该标记嵌套在 GridBlox 标记中:

```
<blox:grid id="cellLinkGridBlox">
<blox:cellLink
scope="{Scenario:Variance %}"
description="Opens information window"
link="/<applicationDirectory>/links/Manhattan.html"/>
<blox:data dataSourceName="QCC-Essbase"
query='<ROW("All Products") "All Products"
<COLUMN(Scenario) <CHILD Scenario
<PAGE("All Locations") Manhattan Sales !'/>
</blox:grid>
```

上面定义的单元格链接将仅显示在 Variance % 下面的 Manhattan 数据单元格中。将要打开的页面位于应用程序文件夹中的 links 子目录中。

指定的 URL 可以是绝对 URL（显示了页面的完整路径），也可以是从应用程序上下文开始的相对 URL（包括最前面的正斜杠）。当该链接是绝对 URL 时，它必须在 URL 中包括服务器名。在以上示例中，<serverName> 表示服务器，<applicationDirectory> 表示文件所在的应用程序目录的名称。

要了解关于 cellLink 属性及其相关联的 getCellLink() 和 setCellLink() 方法的语法及用法详细信息，请参阅 *Developer's Reference*。

使用单元格提醒链接

如第 156 页的『单元格提醒链接』所述，cellAlert 还可以显示链接。cellAlert 属性创建的链接与 cellLink 属性创建的链接共同为您提供了不同的方法来突出显示用户可能想了解的信息。

另一个选项是使用 UI 模型的可扩展性来扩展用户界面以获取关于某个单元格的多幅图像。

要了解关于 cellAlert 链接以及通过 cellLink 创建的链接的详细信息，请参阅 GridBlox Tag Reference 一书。

网格数据单元格中的注释

在组织机构内共享信息的行为经常涉及到对数据添加注释，但这些注释在电子邮件消息中或别的位置经常会丢失。DB2 Alphablox 包括了将这些重要的注释添加至注释数据库以及在用户分析上下文中查看它们的能力。通过使用此功能，用户可以通过对特定数据单元格检索注释或者通过在应用程序页面上的不同列表中查看注释来查看与那些单元格相关联的注释。

使用 CommentsBlox 组件可以支持两种类型的注释，即单元格级别的注释和指定的注释。单元格级别的注释是与特定数据单元格相关并显示在网格中的注释。可以根据一组维来定义单元格级别的注释。指定的注释是具有可以用来定义注释作用域的字符串地址的注释。

如果已经为一个网格中的数据单元格添加了注释，并且该网格支持注释，则将显示注释指示器。缺省情况下，注释指示器是一个红色小三角，它显示在具有相关注释的数据单元格的右上角。当用户右键单击具有注释指示器的单元格时，在上下文（右键单击）菜单中会出现“注释”选项。提供了两个子菜单选项，“添加注释”允许用户将新注释添加到所选单元格，“显示注释”允许用户查看所选单元格上的可用注释。

注: 要使用“注释管理”对话框，您需要具有创建和废弃关系表的权限。要在开发定制注释应用程序时使用 CommentsBlox API，可能需要具有选择、插入、更新、删除、创建和废弃表的权限。

关键术语

描述

注释集合

单个立方体的一组注释的存储库。它存储在关系数据库中。

CommentsBlox

表示页面上的注释集合。它包括一组标记，这些标记嵌套在 DataBlox 中。

CommentsSet

一组注释，这些注释是为单个数据单元格存在的，或者具有相同的地址或名称。它包括所有具有一个作用域或地址的注释（例如，Product:100，Year:Qtr1，Scenario:Actual 的注释）。

注释的元素

单个的注释具有下列部件:

注释元素

描述

作者 必需。注释的作者。缺省情况下，在创建注释时，DB2 Alphablox 将此字段设置为当前已登录的用户。

时间戳记

必需。注释的创建时间。当第一次将注释保存至注释集时，服务器将自动设置此时间。

注释文本

必需。注释的文本，此文本可以包括超链接，甚至可能包括格式化文本（使用 HTML）。对文本大小没有限制。

定制字段

为了提供最大的灵活性，可以为注释集中的注释定义其他定制字段。例如: 主题、重要性和单元格值。

地址 每个注释都有地址。对于单元格级别的注释，地址将是唯一地标识与该注释相关的单元格的 <dimension, member> 列表。对于指定的注释，地址仅仅是由开发者定义含义的字符串。例如，一组 Blox 级别的注释的地址可以由与那些注释相关联的 Blox 的名称组成。对于应用程序级别的注释，地址可以是该应用程序的名称。组装人员可以使用此字符串来为注释定义名称空间以及帮助提供个性化功能。

单元格级别的注释可以具有一个地址方案，该方案包含特定立方体中的一小组维。未包括在集合定义中的维将被忽略。例如，如果立方体包含 Time、Measures 和 Product 这三个维，并且您将当前集合中的注释定义成是使用 Time 和 Measures 的值指定的，则定义的任何注释都适用于 Product 的任何值。通常，应该将所有可能作为报告一部分被处理的维都包括在注释定义中。

单元格级别注释的这种地址方案具有两项用途。首先，它使得对注释进行的管理更为简单，在较大型的轮廓中尤其如此。其次，它使得跨立方体和数据源共享注释变得更为容易。根据 Product、Time 和 Measures 定义的注释集可能适用于许多数据源，而根据轮廓中每个维定义的注释集则有特定于立方体和数据源的风险。

定义注释集合

要定义注释集合，需要创建关系数据源。此数据源可以保存多个集合。创建注释集合的任务需要执行两个步骤:

1. 创建一个数据源并在 DB2 Alphablox 数据源定义页面上定义它。
2. 创建注释集合存储库。

   要定义注释集合，请打开 DB2 Alphablox 管理页面，然后单击“管理”选项卡。在左边的菜单中，在“运行时管理”下面，单击“注释管理”以打开“注释管理”窗口。

   集合需要集合名以及从立方体中选择的维，并且要求创建要使用的字段。作者、时间戳记和注释文本字段是自动定义的，但您也可以创建定制字段。

   “注释管理”窗口还提供了有关配置注释集合的帮助。

启用单元格注释

要对网格中的数据单元格启用注释，需要执行下列步骤:

1. 在独立的或嵌套的 GridBlox 中，添加 `commentsEnabled` 属性并将其设置为 `true`。
2. 在以上网格的独立的或嵌套的 DataBlox 中，添加 CommentsBlox 标记，并指定注释集合的 `collectionName` 和 `dataSourceName` 属性。数据源和集合名是使用 DB2 Alphablox 管理页面的“管理”选项卡中的“注释管理”定义的。

以下示例显示如何使 PresentBlox 能够支持注释:

```
<blox:present id="myPresentBlox">
<blox:grid commentsEnabled="true" />
<blox:data dataSourceName="QCC-Essbase" query="<%=query%>">
       <blox:comments
```

```
collectionName="sales_comments"
dataSourceName="CommentsCollection" >
</blox:comments>
</blox:data>
</blox:present>
```

一旦完成此操作，用户就可以右键单击数据单元格并添加或查看注释。开发者不需要执行任何其他步骤就可以提供基本注释支持了。

添加定制注释支持

用户添加他们自己的注释以及查看别人的注释的能力是一项功能强大的协作和信息共享机制。启用注释的操作是现成的，且易于配置和使用。但是，CommentsBlox 的强大功能和灵活性允许开发者定制对注释的使用。以下是几个可能的定制示例，它们通过使用 CommentsBlox 来进一步增强应用程序。

注: 要了解关于 CommentsBlox 的语法和用法详细信息，请参阅 CommentsBlox Tag Reference 一书。

有时，用户可能希望能够添加关于特定主题或特定分析视图的注释，而不必将那些注释与特定数据单元格相关联。通过使用 CommentsBlox 标记和服务器端 Java API，可以很容易地实现此目标。在“Blox 样本程序”应用程序的“对数据添加注释”部分下面，“页面上的一般注释”示例显示了一个允许添加一般注释（该注释显示在网格下面）并在单独的注释窗口中查看该注释的示例。

用户可能希望能够打印出与特定网格相关联的所有注释。Blox 样本程序的“对数据添加数据”部分中包括的“打印网格中的注释”示例提供了一个按钮，该按钮将打开一个新的浏览器窗口并显示该网格中的所有注释。

第 17 章 与数据进行交互

本主题着重讲述用户行为以及与 Blox 的交互。讨论的主要问题包括如何控制和修改 Blox 的行为、用来限制行为的基本技术以及如何捕获用户操作和控制交互及操作。

交互功能注意事项

交互式的可视表示 Blox 使用户能够处理他们看到的视图、对数据进行下寻和上寻、更改图表类型以及执行许多其他选项。根据应用程序、用户以及他们掌握的技巧水平的不同，您可能会决定排除或限制应用程序控制权。下列小节讨论了在限制与 Blox 的交互时要考虑的问题。

允许有限制的交互或不允许交互

如果用户只需要重要数据的一个简单视图，并且对处理数据以进行更深入的分析不感兴趣，那么使用 GridBlox、ChartBlox 或 PresentBlox 就可以了。您甚至可以考虑在未提供交互功能的 Blox 视图中显示数据切片。

例如，为了防止与 Blox 进行交互，可以添加 Blox UI component 标记（<bloxui:component>）并将 clickable 标记属性设置为 false。例如，以下代码显示了使用嵌套的 <bloxui:component> 标记来生成禁用了用户交互的 PresentBlox:

```
<blox:present id="myPresentBlox"
width="80%"
height="70%"
menubarVisible="false">
   <blox:toolbar
visible="false" />
<blox:data
bloxRef="dataBlox" />
<bloxui:component name="myPresentBlox"
clickable="false" />
</blox:present>
```

对于由于“太忙”而无法钻取数据或者对了解如何处理数据不感兴趣的用户来说，禁用交互功能可能是最佳的解决方案。例如，公司的高层管理人员可能只对查看公司运作状况的快照视图感兴趣，而将详细的分析工作留给业务或财务分析人员完成。

以下任务显示了如何禁用整个 Blox 或嵌套在另一个 Blox 中的所选 Blox。

禁止对列进行旋转和钻取

在“Blox 样本程序”的“Blox UI 标记”部分中，蝶形报告示例包括一个事件过滤器以防止用户对列进行旋转或钻取。这两项用户操作都将导致显示的非对称报告的显示不再正确。为了防止用户陷入进退两难的僵局，您可以使用 UI 模型添加一个事件过滤器来捕获用户的旋转或钻取尝试，从而保持视图可用。

以下代码段显示了对网格使用的用于防止对列进行旋转和钻取的事件处理程序:

```
<%
  GridBloxModel model =
    butterflyReportGridBlox.getGridBloxModel();
```

```
  model.populateDataNavigationButton();
  model.getGrid().getController().addEventHandler(
     new IEventHandler() {
  public boolean handleGridDataActionEvent(GridDataActionEvent
       event ) throws Exception {
   GridBrixCellModel cells[] = event.getGridCells();

// If any of the cells is a header cell, then ignore the data action
  for ( int i=0; i < cells.length; i++ )
  if ( cells[i].isColumnHeader() )
  return true;
      return false;
}
} );
%>
```

要获取完整的代码示例，请参阅“Blox 样本程序”。

要了解关于将事件处理程序与 Blox UI 模型的 DHTML 可扩展功能配合使用以便象此示例一样定制应用程序的详细信息，请参阅 第 66 页的『Blox UI 模型事件』。

使用 Blox 属性来修改交互性

也可以使用 Blox 属性和方法来控制用户交互。当在数据表示 Blox 上启用了 Toolbar、DataLayout 和 Page 面板时，用户可以与数据进行更多的交互。您可能会发现，尽管这样做对某些用户有帮助，但有些用户将很快地失去对数据的控制，当他们不熟悉数据的结构时尤其如此。除了上面描述的使用 `<bloxui:component>` 标记和 Blox `visible` 属性的技术以外，还可以使用其他 Blox 属性来调整视图的交互性，从而启用某些面板而不启用其他面板以及限制可能导致用户陷入困境的方式。并且，通过使用个性化技术，可以使用服务器端 Java、JavaServer Pages 和 JavaScript 方法来根据用户登录定制交互性。

下表列示了一些常用的 Blox 属性，这些属性将对用户与数据的交互产生影响:

Blox	属性（或相关联的方法）	描述和注释
GridBlox	`cellAlert`	可以使用单元格提醒链接来打开信息窗口或调用 JavaScript 函数
	`cellLink`	可以使用单元格链接来打开信息窗口或调用 JavaScript 函数
	`expandCollapseMode`	启用类似于 Windows 资源管理器的工作方式，即使用加号和减号图标来在网格中进行上寻和下寻
	`writebackEnabled`	允许授权用户根据具有作用域的单元格来直接将数据输入到网格中

Blox	属性（或相关联的方法）	描述和注释
DataBlox	drillDownOption	确定钻取操作是转至下一代、所有后代、底部代、同代还是同一代
	drillKeepSelectedMember	保留正被钻取的所选成员
	drillRemoveUnselectedMembers	在进行钻取时除去未选择的成员
	enableKeepRemove	指定“仅保留”和“仅除去”选项是否可用
	enableShowHide	指定“仅显示”、“全部显示”和“仅隐藏”选项是否可用
DataLayoutBlox	interfaceType	指定用户如何选择维，即是使用下拉列表还是使用拖放树界面
ReportBlox	请参阅*《关系报告开发者指南》*。	

注: 由于属性更改而导致的交互行为更改会在通常熟悉的行为不能按预期那样起作用时令用户感到困惑或惊讶。例如，将 DataBlox 的 drillKeepSelectedMember 和 drillRemoveUnselectedMembers 设置为 true 有时会很有用，这可以帮助用户有效地管理网格或图表上的可视信息量。一项重要的注意事项是，当用户进行钻取时，如果一个应用程序中的或者多个应用程序中的所有视图并不具有相同的行为方式，当特定 Blox 的行为与该用户遇到的所有其他 Blox 的行为有所不同时，他就会非常困惑。在此类情况下，一种为用户提供帮助的方法是在页面上清晰地注明该用户应有的信息。此外，可以使用单选按钮或复选框来允许用户在两种钻取行为之间进行切换。

网格

网格或者是独立的 GridBlox，或者嵌套在 PresentBlox 中。在任何一种方式下，用户都可以对数据进行钻取、旋转、排序和探索。但是，PresentBlox 中使用的网格包括一些在显式 GridBlox 中未提供的附加功能。下表提供了独立的 GridBlox 与嵌套的 GridBlox 之间的关键区别概述:

功能	独立的 GridBlox	嵌套的 GridBlox（位于 PresentBlox 中）
DataLayoutBlox	需要独立的 DataLayoutBlox，使用同一个 DataBlox	是使用 PresentBlox dataLayoutAvailable 提供的
PageBlox	需要独立的 PageBlox，使用同一个 DataBlox	是通过将 pageAvailable 设置为 true 提供的
ChartBlox	需要独立的 ChartBlox，使用同一个 DataBlox	网格自动地与图表同步。可以提供图表。

由于上面列示的功能是使用 PresentBlox 提供的，因此，在大多数情况下，您想让用户能够访问此功能。

与网格相似，用户可以对图表中显示的数据进行上寻和下寻。但是，与网格不同，用户可能不会意识到他们可以与图表进行交互 – 不存在可视信号（如网格的向上／向下箭头或加号／减号图标）来帮助用户了解他们可以直接与图表进行交互。用户第一次意识到他们可以对图表进行钻取可能是当他们看到别人这样做时，或者仅仅是无意中尝试这样做时，或者是在右键单击图表元素并看到菜单选项时。当用户将光标悬浮在图表栏和数据点上时，大多数用户会发现数据值和标注会显示。如果有必要的话，您可以使用 ChartBlox 的属性（如饼图的 `pieFeelerTextDisplay` 和条形图的 `dataTextDisplay`）来显示值或标注，而不要求用户将光标移到图表元素上。

无论用户是从远程位置通过因特网访问应用程序还是坐在办公室中靠近您的位置使用该应用程序，他们可能都未曾接受培训或者对应用程序以及那些应用程序的用法了解有限。作为开发者或应用程序设计者，您需要考虑如何使分析应用程序尽可能地简单易用。如果向用户显示图表页面并且未提供指示信息，当您发现许多用户从来不与图表进行交互而仅仅是查看您向他们显示的内容时，您不应该感到惊讶。当您进行设计时，请考虑如何提高用户取得成功以及学习更高效地使用图表的可能性。添加用于访问 DB2 Alphablox 帮助或定制帮助页面的“帮助”或“技巧”按钮将有助于他们了解如何使用应用程序。此外，还可以将一些简短的提示直接放在页面上。在某些情况下，您可能会发现，使用 ChartBlox 的 `footnote` 属性将用户信息直接放在图表脚注中可能会很有用。

允许用户控制所显示的生成

向图表添加交互性的一种简单方法是向 ChartBlox 添加 `totalsFilter` 属性并将该属性设置为 2。通过将值设置为 2，您允许在图表底部显示总计过滤器滑块面板。于是，根据所使用的查询的不同，用户可能能够控制所显示的维级别。在以下 ChartBlox 示例中，`totalsFilter` 属性设置为 2:

```
<blox:chart id="totalsFilterChartBlox"
width="90%"
height="90%"
chartType="Bar"
totalsFilter="2"
title="Truffle Sales for 2001">
<blox:data
dataSourceName="QCC-Essbase"
query='<ROW ("All Products") <ICHILD "All Products"
<COLUMN ("All Time Periods") <DESCENDANTS "2001" Sales !'/>
</blox:chart>
```

当显示在页面上时，在图表下方的面板上将显示两个代选择器。左边的选择器允许用户控制 All Products 维的代级别，右边的选择器允许用户选择 2001 年的若干个时间段的代级别。

示例: “Blox 样本程序”中的“与数据进行交互”部分中的“启用了图表 totalsFilter 选择器”示例显示了 `totalsFilter` 滑块面板的使用。

DataLayout 界面

当在 PresentBlox 上提供了 DataLayout 面板（DataLayoutBlox）时，用户可以通过访问该面板来在“行”轴、“列”轴和“其他”轴（页面过滤器轴）之间移动维，以便在网格和图表中创建他们希望查看的数据布局。缺省情况下，DataLayout 面板在下拉列表（即选择列表）中显示维，该列表允许用户单击维名并选择一个选项来移动该维。此外，开发者也可以将 DataLayout 面板设置为使用拖放树界面，此界面的行为与 Windows 资源管理器更为类似。要显式地设置 DataLayout 面板的界面类型，请将 DataLayoutBlox 的 `interfaceType` 属性设置为 `dropLists`（缺省值）或 `tree` 这两个值的其中一个值。以下示例显示了已被设置为显示树界面的 DataLayoutBlox:

```
<blox:present ...>
...
   <blox:dataLayout
interfaceType="tree" />
...
</blox:present>
```

注: 界面类型只能由开发者设置 – 未提供允许用户选择此界面选项的用户界面选项。缺省情况下，用户将看到下拉列表界面。

网格与图表之间的交互

可以使用 GridBlox 和 ChartBlox 组件来一个个地显示网格和图表，也可以将网格和图表一起嵌套的 PresentBlox 组件中。当作为独立的 Blox 出现时，每个 Blox 都可以使用隐式数据源（在嵌套的 DataBlox 中定义）或显式的独立数据源（使用独立的 DataBlox 组件）。网格视图与图表视图也可以共享公共的独立 DataBlox 来作为它们的数据源。对于嵌套在 PresentBlox 中的 GridBlox 和 ChartBlox 来说，情况始终如此。

当 GridBlox 组件与 ChartBlox 组件共享公共数据源（使用独立的 DataBlox）时，对 GridBlox 执行的操作将在 ChartBlox 中反映出来。因此，当用户对 GridBlox 中的成员进行下寻时，共享同一 DataBlox 的 ChartBlox 也将执行并显示相同的钻取操作。由于网格与图表共享同一个数据源，所以在 PresentBlox 中会发生网格与图表之间的这种同步行为。

头链接、单元格提醒、单元格链接和其他 GridBlox 功能在图表中不可用。为了同时使用这两种图表以及这些功能，您可能想使用 PresentBlox。并且，如果网格在 PresentBlox 视图中不可视，则用户将看不到提醒或者无法访问基于网格的链接，除非他们通过网格组件来访问它们。

您要认识到的另一项重要情况是，数据的格式化是在网格和图表中独立设置的。因此，如果要让网格和图表对值使用相同的格式化，则您需要记住设置所有下列 Blox 属性:

Blox 属性

ChartBlox

`y1FormatMask y2FormatMask`

GridBlox

`defaultCellFormat cellFormat`

在网格和图表中设置“没有数据可用”消息

当数据源不可用或者尚未检索到结果集时，DB2 Alphablox 网格和图表将显示以下缺省消息：“没有数据可用”。

如果检索结果集所需的时间比用户预期的时间要长，则缺省的“没有数据可用”消息可能会具有误导性。尽管在那个时候确实没有数据可用，但是如果用户再等待一小段时间，数据通常就会出现。如果检索操作花费的时间不止几秒钟，用户就可能会认为应用程序的工作不正常，从而尝试重新装入应用程序或页面，而不会等待足够长的时间来让数据出现。当这种情况可能成为问题时，许多 DB2 Alphablox 开发者将 noDataMessage 设置为类似于下列其中一条消息的消息：请等待数据... 或正在等待数据...。

这通常是一种不错的解决方案，但是在确实没有数据可用时除外。在这些情况下，此消息不会更改为指示没有数据可用。在决定修改 noDataMessage 之前，请考虑它的影响，但是，更清晰的消息的好处通常胜于数据源实际上不可用的可能性。

要修改网格和图表中显示的消息，请对 PresentBlox、GridBlox 或 ChartBlox 添加 noDataMessage 属性。如果对 PresentBlox 添加了 noDataMessage 属性，则在嵌套的 GridBlox 和 ChartBlox 中都将显示新消息。如果您希望单独地为嵌套的 GridBlox 和 ChartBlox 设置值，可以单独地对每个嵌套的 Blox 设置 noDataMessage 属性。对 PresentBlox 中嵌套的 GridBlox 和 ChartBlox 进行的下列设置将导致在网格和图表中显示不同的消息：

```
<blox:present ...>
<blox:grid
noDataMessage="Grid not available"/>
<blox:chart
noDataMessage="Chart not available"/>
</blox:present>
```

如果审慎地使用，对 noDataMessage 属性进行的更改就会使应用程序更为优秀。要了解有关 noDataMessage 的更多信息，请参阅 *Developer's Reference* 一书的 Common Blox 一节。

HTML 表单元素和 FormBlox 组件

使用 PresentBlox、GridBlox 和 ChartBlox 创建的网格视图和图表视图可以包括内置的功能部件，如工具栏、页面过滤器和上下文（右键单击）菜单。如果提供了工具栏，用户就可以使用用于修改图表、网格和数据外观及行为的选项菜单。但是，提供许多选项有时会使初学者和普通用户感到不知所措。根据您的特定需求的不同，最佳的做法可能是提供数目有限的选项。通过组合使用 HTML 表单元素、JavaScript、Java 和丰富的 Blox API，您可以创建定制分析应用程序并根据用户的需求和技能来调整选项或者调整选项以定制交互。并且，在“Blox 样本程序”中，您能够找到一些使用 HTML 表单元素（按钮和复选框等）来提供受控交互性或者用于更改数据视图的选项的示例。

但是，在大多数情况下，更引人注目的选项可能是使用 FormBlox 组件。当使用 Blox 表单标记库来管理 HTML 表单元素时，就可以使用该组件。在本指南第 41 页的『使用 Blox 表单标记库』以及在 Blox 样本程序示例中更全面地讨论了关于 FormBlox 组件及其用法的详细信息。

并且，“Blox 样本程序”提供了许多使用 HTML 元素和 FormBlox 组件来控制 Blox 交互性的示例。特别是，请您查看“使用 FormBlox 和逻辑 Blox”部分或“与数据进行交互”部分中的示例。

下列小节着重讲述了某些标准 HTML 表单元素及其等效的 FormBlox 组件。

选择列表

通过使用 Blox API 属性和方法，可以通过固定的选项列表（使用 `fixedChoiceLists`）、虚拟根（使用 `dimensionRoot`）和成员过滤器（使用 `moreChoicesEnabled` 和 `moreChoicesEnabledDefault`）来定制 PageBlox 页面过滤器。但是，PageBlox 页面过滤器有时无法满足您的所有需求。

如果关闭网格视图和图表视图上的工具栏，则可以创建下拉菜单项来替换不再提供的工具栏。硬编码的选择列表使您能够为最终用户提供受控的选项，同时使得分析视图更简单易用。例如，“图表”按钮附带提供的图表类型列表所带的图表类型对于特定视图来说可能过多，因此，您可以在选择列表中提供一小组有限的图表类型。这样，用户就可以使用一些数据显示方式选项，而不会被提供对特定视图可能没有意义的选项。

级联选择菜单对于让最终用户从一个列表中选择选项并接着根据他们对其他列表所作的选择来提供辅助菜单选项来说很有用。目前，您可能主要使用 HTML 表单元素和 JavaScript 来创建自己的级联菜单。但是，通过使用 Blox 表单标记库中提供的 MemberSelectFormBlox，您可以快速地创建级联菜单并使其与某个数据视图相联系，而且编码量要小得多。所使用的 FormBlox 组件还将处理持久性。也就是说，在用户会话期间，动态生成的 HTML 表单元素将在用户离开页面以及以后返回时保持它们的选择。要查看这种情况的实际应用，请查看“Blox 样本程序”的“使用 FormBlox 和逻辑 Blox”部分，在那里，您可以找到一个 MemberSelectFormBlox 示例，该示例有三个根据用户选择动态更改的选择列表。

Blox 中的页面过滤器通常按照指定的顺序来只显示某个维的成员。但是，通过进行定制编码或使用 MemberSelectFormBlox 将页面过滤器移至 HTML 表单元素，您可以创建动态生成的页面过滤器，这些页面过滤器基于对数据源执行的定制查询。

复选框和单选按钮

当网格视图或图表视图未提供工具栏时，可以使用定制编码的或使用 FormBlox 组件（CheckBoxFormBlox 和 RadioButtonFormBlox）创建的复选框和单选按钮来向用户提供当菜单栏或任务栏不可用时无法访问的选项。由于用户没有经常使用过工具栏提供的各种对话框，所以他们可能甚至不知道某些选项可用。在视图中不提供工具栏和菜单栏的另一个优点是，您可以提供一组有限的选项，这些选项在页面上将更为明显。例如，可以提供根据复选框是否被选取而切换状态的“消除缺少的值”和“消除全零”复选框。单选按钮非常适合于提供相互不兼容的选项。在“Blox 样本程序”中有大量的示例使用 FormBlox 组件来管理复选框和单选按钮。

标准 HTML 按钮

在应用程序中，标准 HTML 按钮对于执行查询、复位查询和生成视图来说非常有用。但是，对于显示不同的视图来说，这些按钮就不那么合适，这是因为，除非页面上的

标题发生更改，否则用户可能无法说出他们是使用了哪个按钮获得当前视图的。使用单选按钮可能是一种不错的替代方法，这是因为选择的特定项始终具有突出显示的单选按钮。

文本字段

在分析应用程序中，文本输入字段不象其他 HTML 表单元素那么常用，但是它们在特殊情况下可能特别有用。在大多数情况下，由于固定的选项（如单选按钮、复选框和选择列表提供的那些选项）可以帮助防止期望的值遇到拼写错误和问题（例如，用户输入了字符而不是数字），所以它们更为可取。在某些情况下，最佳的或唯一的做法选项是允许用户自己输入值。例如，在分析应用程序中，可以使用文本字段来允许用户输入要回写至数据源的数据或者通过设置他们自己的单元格提醒阈值级别来将应用程序个性化。可以使用文本字段和文本区域来允许用户在应用程序中添加注释。在 DB2 Alphablox 应用程序中，您可以选择使用定制编码的 HTML 文本字段或者使用 EditFormBlox 组件来创建要在分析应用程序中使用的文本字段。EditFormBlox 具有内置的自动更改其他 Blox 组件属性的功能。

使用工具栏按钮

用户与之交互的每个 Blox 组件可以包含一个用于访问 Blox 功能的工具栏。DB2 Alphablox 提供了数种方法来创建定制的 Blox 工具栏，这些工具栏能够增强用户的感受。

缺省情况下，每个 Blox 都显示它的工具栏。要使工具栏在特定 Blox 上不可视，请将 Blox 的 `visible` 属性设置为 `false`。

当与设置为 false 的 Blox `clickable` 属性配合使用时（请参阅第 167 页的『允许有限制的交互或不允许交互』），结果将是静态的数据表示，而不是交互式的 UI。这可能适合于快速快照和执行报告。当应用程序使用定制 HTML 用户界面来替换工具栏功能时，使 Blox 工具栏不可视也可能是合适的。

缺省情况下，基于文本的菜单栏显示在 Blox 工具栏上方。要使其出现，请将 Blox `menubarVisible` 属性设置为 `true`。

缺省情况下，工具栏上的每个按钮都不会显示描述性的文本标注。要打开此文本（这样做会增加某些按钮所需的显示空间），请将 `textVisible` 属性设置为 `true`。注意，打开工具栏文本会导致工具栏变大，这将占用更多的 Blox 区域。

可以使用嵌套的 ToolbarBlox 的 `removeButtons` 属性来指定要从工具栏中除去的按钮。要了解有关这些以及其他 ToolbarBlox 功能的详细信息，请参阅 *Developer's Reference* 一书的 ToolbarBlox 一节。

事件

DB2 Alphablox 提供了一些用于处理事件的属性和方法。*事件*是可以用来触发进一步的处理的正常操作。

Blox 可以捕获下列用户操作并将它们视为事件:

- 下寻或上寻
- 旋转

- 选择头或单元格菜单项
- 更改页面过滤器
- 装入或保存书签
- 更改网格单元格中的数据值
- 仅保留或仅除去
- 仅隐藏或仅显示

从第 66 页的『Blox UI 模型事件』 开始的内容对 Blox UI 模式事件作了描述。并且，UI 模型显示了许多可以由客户机发出的事件，如 `ClickEvent`。对于其中的每一个事件，DHTML 客户机 API 都会定义 JavaScript 对象。因此，可以使用 JavaScript 来创建事件对象并将该事件发送给服务器。要了解更多信息，请参阅第 84 页的『发送事件』、第 85 页的『拦截事件』和第 86 页的『直接从用户界面中调用 JavaScript』。

第 18 章 输入和修改数据

可以使用 DB2 Alphablox 来输入数据或将数据数据源回写至数据源。您可以使用计算的成员来根据从数据源中检索的数据创建新数据。

回写至多维数据源

您可以使用 DB2 Alphablox Java 方法来修改结果集并更新其底层数据源。要使用户能够更新数据单元格值并接着将那些值回写到底层数据源，您需要利用一组属性和相关联的方法来达到以下效果和结果:

- 使网格能够被编辑
- 定义网格中可供编辑的单元格
- 指定可编辑的单元格的显示样式（通常是前景色或背景色）
- 指定已编辑的单元格的显示样式
- （可选）指定当用户编辑单元格时要进行的处理

GridBlox 属性和相关联的回写方法

对于设计和管理回写应用程序来说，下列 GridBlox 属性及相关 Java 方法是必需的或可用的:

属性和相关方法	描述
• writebackEnabled • isWritebackEnabled() • setWritebackEnabled()	启用回写所必需的；允许用户编辑网格单元格
• cellEditor • getCellEditor() • setCellEditor()	启用回写所必需的；指定数据单元格可编辑区域的定义及突出显示规则

在『GridBlox Java 回写方法』中讨论了其他与属性不相关的方法。要指定用户已编辑或可编辑的单元格的外观，可以使用 CSS 主题。

GridBlox Java 回写方法

以下是所有不具有相关属性的 GridBlox Java 方法:

getWritebackValue(); setWritebackValue()
设置网格中特定数据单元格的值或者返回网格中已更改的特定数据单元格的值

listCellEditorIds()
以整数数组形式返回已定义的所有单元格编辑器的标识列表

getChangedCellList()
返回已编辑的单元格的 String

getChangedCellValues()
返回已编辑的单元格值的 String

启用 GridBlox 组件进行数据回写

以下示例包括对 GridBlox 组件启用回写能力所需的最少属性以及其他常用属性。

```
<blox:grid id="Grid1"
width="800"
height="500"
writebackEnabled="true">
<blox:data
bloxRef="Data1"/>
<blox:cellEditor
scope="{Market:New_York}"/>
</blox:grid>
```

上述 GridBlox 属性用于启用回写功能、定义可编辑单元格以及更改可写数据单元格的外观。为了将数据回写至数据源，提供了一组与 DataBlox 回写相关的方法供您使用。请参阅『DataBlox 的与回写相关的方法』。

DataBlox 的与回写相关的方法

在配置了 GridBlox 以启用回写功能之后，需要使用服务器端 DataBlox 方法来执行回写操作。这些回写方法是为需要将数据回写至 DB2 OLAP Server、Essbase 或 Microsoft Analysis Services 2000 数据源的应用程序设计的。某些方法仅适用于 DB2 OLAP Server 或 Essbase。可用的 DataBlox Java 回写方法如下:

writeback()
这是一个很方便的回写方法，它接收 3 个自变量并使用下列方法:

- lockCurrentDataSet()
- setDataValues()
- commitData()
- unlockAll()
- executeCustomCalc()
- refresh()

lockCurrentDataSet()
锁定被请求结果集；不锁定整个数据库

setDataValues()
更改结果集中位于指定坐标处的数据值

commitData()
将当前数据集回写到数据库中

unlockAll()
将 DB2 OLAP Server 或 Essbase 数据库中先前被锁定的任何数据解锁

executeCustomCalc()
对 DB2 OLAP Server 或 Essbase 数据库执行计算脚本

refresh()
刷新当前数据集

executeNamedDBCalcScript()
执行指定的 DB2 OLAP Server 或 Essbase 计算脚本

启用至多维数据库的回写功能

通过使用服务器端 Java API，您可以创建应用程序页面来允许用户将数据回写至多维数据库。“Blox 样本程序”包括三个示例，其中一个示例使用定制 Java 类（建议的方法），另外两个示例使用 Blox UI 控制器（一般控制器和定制控制器）。

限制：DB2 OLAP Server 和 Hyperion Essbase 查询不支持在回写操作中使用属性维。

“Blox 样本程序”中提供的定制 Writeback 类包括以下步骤：

1. 添加一个 JSP page 伪指令以导入所需的类：

```
<%@ page import="bloxsampler.writeback.Writeback,
com.alphablox.internal.PresentBlox" %>
```

2. 为页面上使用的 Blox 标记库添加一个 JSP taglib 伪指令：

```
<%@ taglib uri="bloxtld" prefix="blox" %>
<%@ taglib uri="bloxuitld" prefix="bloxui" %>
```

3. 在页面的 head 部分中，请添加必需的 <blox:header/> 标记，此标记将自动地添加必需的 CSS 和 JavaScript 方法：

```
<head>
   <blox:header />
</head>
```

4. 添加一个 JavaScript 函数，此函数使用 Blox UI 模型来在服务器上执行模拟的单击事件，就象是用户单击了 PresentBlox 菜单栏中的**数据 → 回写**菜单选项一样：

```
<script language="JavaScript">
function wb() {
bloxAPI.sendEvent(new ClickEvent('wb3PresentBlox',
<%= wb3PresentBlox.getBloxModel().searchForComponent(
"dataWriteback").getUID() %>));
}
</script>
```

5. 添加一个 PresentBlox 并将它的网格 writebackEnabled 属性设置为 true，然后包括一个 scriptlet，该 scriptlet 引用定制 Writeback 类（带有两个自变量，它们分别是 Blox 名和作用域）：

```
<blox:present id="wb3PresentBlox"
height="400"
width="600"
pageAvailable="true"
chartAvailable="false"
dataLayoutAvailable="false">
<blox:data
dataSourceName="QCC-Essbase"
query='<SYM <ROW("All Products") <CHILD Truffles
<COLUMN(Scenario) "Initial Budget" Manhattan
<ICHILD "Jan 01" "Units Sold" !' />
<%
new Writeback(wb3PresentBlox,"{Scenario:Initial Budget}");
%>
</blox:present>
```

6. 在页面上添加一个按钮，用户单击此按钮时，将把数据回写至数据源。

```
<form>
<input type="button" value="Submit Changes"
onclick='setTimeout( "wb();", 1 );'>
</form>
```

JavaScript wb 函数是在停顿一小段时间之后被调用的，这样做是为了允许在此函数被调用之前更新服务器上的数据。如果未使用 JavaScript setTimeout 函数，则可能无法将正确的数据回写至数据源。

在本示例中，回写功能现已被合并到 PresentBlox 中。在“Blox 样本程序”中，有关“输入和更改数据”的部分提供了这种回写技术的一个有效示例。 Java 类的源代码位于应用程序的 WEB-INF/src/ 目录中。您可以根据需要修改此源代码以进行任何其他所需的更改，然后编译源代码以便在应用程序中使用。

将数据回写至 Microsoft Analysis Services

对于 Microsoft Analysis Services，只能将数据回写至叶级成员。要更新非叶成员中的数据，需要在 DataBlox setQuery 或 executeQuery 方法中使用 MDX UPDATE CUBE 命令。要了解有关 UPDATE CUBE 命令的更多信息，请参阅 Microsoft Analysis Services 文档。

更新关系数据源

DB2 Alphablox 支持使用标准 SQL 语句来更新关系数据源。这些语句包括（但不限于）insert、update、create 和 delete。可以使用 Java 方法来构造适当的 SQL 语句，然后将该语句传递至应用程序的 DataBlox。将数据回写至关系数据源的操作并不会影响用户的数据视图，但是在用户能够看到数据更改效果之前，必须重新执行查询。要重新执行查询，需要调用 DataBlox setQuery() 方法，然后调用 connect() 方法。

使用回写功能来更新关系数据源

要使用 DataBlox Java 方法来更新关系数据源，请将一个包含当前数据的新列插入表中。

1. 创建名为 query1 的 SQL 查询字符串。此查询字符串将把当前日期插入到名为“review_data”的表中。

   ```
   String query1 = "insert into review_data values(TO_CHAR(SYSDATE,
   'HH:MM:SS-MMDD'))";
   ```

2. 对 PresentBlox 的数据源调用适当的 setQuery 和 connect 方法来传递将插入新列的 SQL 查询。在本示例中，PresentBlox 命名为 Present1。

   ```
   Present1.getDataBlox().setQuery(query1);
   Present1.getDataBlox().connect();
   ```

创建日历控件

Blox UI 模型提供了一个 DateChooser 组件，该组件使您可以创建图形日历控件，以允许用户选择在生成分析视图和其他应用程序时使用的日期。此组件在 Web 页面上添加一个文本字段，且该文本字段旁边有一个小日历图标。当用户单击该图标时，会打开一个小日历，以允许用户选择日期并使用已正确格式化的日期填充该文本字段。

DateChooser 组件包括以下子组件。

子组件	描述
文本字段的图形用户界面	DateChooser 组件扩展“编辑”组件并支持大多数“编辑”API，如 insertText()、setValue()、getValue() 和 clear()。

子组件	描述
用于启动日历的图像	DateChooser 组件包括一个用于启动日历的“图像”子组件，该子组件显示在文本字段旁边。 `getIcon()` 方法提供对此图像组件的访问权。例如，`getIcon().setImageURL("myCalendar.gif")` 设置要用于该图标的图像文件。缺省情况下，该图像基于主题。将该图像文件放置在 DB2 Alphablox 存储库中主题的图像目录中。要将图像放置在其他位置，请调用图像的 `setThemeBasedImage(boolean)` 方法并将它设置为 `false`。有关详细信息，请参阅 Blox API Javadoc 文档。
日历适配器	此 CalendarAdapter 对象是实现 com.alphablox.blox.uimodel.ICalendar 接口的包装程序。此接口符合 java.util.Calendar API（如 `getTimeInMillis()` 和 `getFirstDayofWeek()`）的子集。使用 ICalendar 接口，您可以提供自己的日历对象以实例化 CalendarAdapter 对象。您的日历对象必须实现 ICalendar 中的所有方法才能工作。 缺省情况下，DateChooser 组件使用英语语言环境创建阳历。International Components for Unicode（ICU）库提供了一组日历对象和 API，它们使您可以轻松地创建非阳历日历。
日期格式适配器	此适配器是实现 com.alphablox.blox.uimodel.IDateFormat 接口的包装程序。IDateFormat 接口符合 java.text.DateFormat API（如 `getCalendar()`、`setCalendar()` 和 `format()`）的子集。日期格式将应用到显示所选日期的文本字段。支持以下日期格式: • FULL（缺省值） • LONG • MEDIUM • SHORT 要获取不同日期格式的示例，请参阅 http://java.sun.com/j2se/1.4.2/docs/api/java/text/DateFormat.html。 使用 IDateFormat 接口，您可以创建具有特定于语言环境的日期格式的非阳历日历。您创建的日期格式化程序对象需要实现 IDateFormat 接口中的所有方法。

相关概念

第 185 页的『创建非阳历日历』

International Components for Unicode（ICU）是一组为获取 Unicode 支持而广泛使用的 Java 库。它提供的功能部件和可扩展性支持非阳历日历和不同日期格式。使用 ICU 包，您可以轻松地创建自己的日历对象和日期格式化程序。

创建阳历

创建日历控件涉及到创建 CalendarAdapter、DateFormatAdapter 和 DateChooser。以下示例显示如何在英语语言环境中创建日期格式的阳历。当用户单击日历图标并从打开的日历中选择日期时，选择的日期用来填充使用 DateFormat.SHORT 日期格式的文本字段。相同的概念和步骤可以应用到其他类型的日历。

要在 JSP 中创建阳历:

1. 导入以下各包和类:

```
<%@ page import="com.alphablox.blox.uimodel.core.*,
  com.alphablox.blox.uimodel.*,
java.util.Calendar,
java.util.Locale"%>
```

2. 导入 Blox 标记库:

```
<%@ taglib uri="bloxtld" prefix="blox"%>
```

3. 添加 ContainerBlox 以包含 DateChooser 组件:

```
<blox:container id="dateChooserContainer">
```

4. 获取容器的 BloxModel:

```
<%
BloxModel model = dateChooserContainer.getBloxModel();
...
%>
```

5. 设置语言环境:

```
Locale locale = Locale.US;
```

6. 创建 CalendarAdapter:

```
java.util.Calendar calendar = java.util.Calendar.getInstance( locale );
ICalendar calendarAdapter = new CalendarAdapter( calendar );
```

7. 创建 DateFormatAdapter 并应用语言环境:

```
java.text.DateFormat dateFormat =
java.text.DateFormat.getDateInstance( java.text.DateFormat.FULL, locale );
IDateFormat dateFormatAdapter = new DateFormatAdapter( dateFormat );
```

8. 通过使用 `DateChooser.getInstanceWithLocale()` 方法来创建带有 CalendarAdapter、DateFormatAdapter 和 locale 的 DateChooser 对象:

```
<%
DateChooser datechooser =
DateChooser.getInstanceWithLocale( calendarAdapter, dateFormatAdapter, locale );
%>
```

9. 将 DateChooser 添加至容器的模型:

```
<%
...
model.add(datechooser1);
%>
```

以下是完整的示例:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="com.alphablox.blox.uimodel.core.*,
  com.alphablox.blox.uimodel.*,
java.util.Calendar,
java.util.Locale"%>
<%@ taglib uri="bloxtld" prefix="blox"%>
<html>
<head>
<blox:header/>
</head>
<body>
<blox:container id="dateChooserContainer">
<%
BloxModel model = myContainer.getBloxModel();
Locale locale = Locale.US;
java.util.Calendar calendar = java.util.Calendar.getInstance( locale );
ICalendar calendarAdapter = new CalendarAdapter( calendar );
java.text.DateFormat dateFormat =
java.text.DateFormat.getDateInstance( java.text.DateFormat.FULL, locale );
IDateFormat dateFormatAdapter = new DateFormatAdapter( dateFormat );
```

```
DateChooser datechooser =
DateChooser.getInstanceWithLocale( calendarAdapter, dateFormatAdapter, locale );
model.add( datechooser );%>
</blox:container>
</body>
</html>
```

可以很容易地修改语言环境以创建非英语阳历。根据您想要如何设置语言环境或最初选择的日期，还有其他工厂方法用于创建 DateChooser。第 185 页的『在启动日历时指定所选日期』示例演示用于创建 DateChooser 的另一个工厂方法。CalendarAdapter 可以使用一个语言环境，DateFormatAdapter 可以使用另一个语言环境，还有一个语言环境可以供 DateChooser 使用。例如，您可以让日语日历使用不同语言的 DateFormat（在文本框中显示所选日期时要使用的格式）。

可以使用 Java 的日历组件或 International Components for Unicode（ICU）Java 库中的日历组件。要支持多个语言环境，应该使用 ICU，因为它有更好的国际化支持。有关更多信息，请参阅『使用多语言环境支持 ICU 来创建阳历』。

要获取现成的 DateChooser 示例，请参阅『UI 可扩展性』一节的 Blox 样本程序中的“DateChooser 组件”示例。

相关概念

第 185 页的『创建非阳历日历』

International Components for Unicode（ICU）是一组为获取 Unicode 支持而广泛使用的 Java 库。它提供的功能部件和可扩展性支持非阳历日历和不同日期格式。使用 ICU 包，您可以轻松地创建自己的日历对象和日期格式化程序。

相关任务

第 185 页的『在启动日历时指定所选日期』

缺省情况下，在启动日历时，将显示当前月份的日历并选择了当前日期。可以指定除当前日期之外的不同初始日期。

使用多语言环境支持 ICU 来创建阳历

因为 ICU 适合于支持非英语语言，所以本示例使用 ICU。本示例演示如何从 Blox 上下文中获取语言环境，因此日期选择器将影响客户机的语言环境设置。

要在 JSP 中创建阳历:

1. 导入以下各包和类:

```
<%@ page import="com.alphablox.blox.uimodel.core.*,
  com.alphablox.blox.uimodel.*,
java.util.Locale"%>
```

2. 导入 Blox 标记库:

```
<%@ taglib uri="bloxtld" prefix="blox"%>
```

3. 添加 ContainerBlox 以包含 DateChooser 组件:

```
<blox:container id="container">
```

4. 获取容器的 BloxModel:

```
<%
  BloxModel model = container.getBloxModel();
...
%>
```

5. 从 Blox 上下文中获取语言环境:

```
Locale locale = bloxContext.getLocale();
```

6. 创建 CalendarAdapter 并应用语言环境:

```
com.ibm.icu.util.Calendar calendar =
new com.ibm.icu.util.GregorianCalendar( locale );
ICalendar calendarAdapter = new CalendarAdapter( calendar );
```

7. 创建 DateFormatAdapter:

```
com.ibm.icu.text.DateFormat dateFormat =
calendar.getDateTimeFormat( com.ibm.icu.text.DateFormat.FULL, -1, locale );
IDateFormat dateFormatAdapter = new DateFormatAdapter( dateFormat );
```

8. 通过使用 `DateChooser.getInstanceWithLocale()` 方法来创建带有 CalendarAdapter、DateFormatAdapter 和 locale 的 DateChooser 对象:

```
DateChooser datechooser = DateChooser.getInstanceWithLocale( calendarAdapter,
dateFormatAdapter, locale );
datechooser.setName("datechooser");
datechooser.setWidth(300);
```

9. 将 DateChooser 添加至容器的模型:

```
model.add(datechooser1);
```

以下是完整的示例:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="com.alphablox.blox.uimodel.core.*,
  com.alphablox.blox.uimodel.*,
java.util.Locale"%>
<%@ taglib uri="bloxtld" prefix="blox"%>
<html>
<head>
<blox:header/>
</head>
<body>
<blox:container id="container">
<%
  BloxModel model = container.getBloxModel();
Locale locale = bloxContext.getLocale();com.ibm.icu.util.Calendar calendar =
new com.ibm.icu.util.GregorianCalendar( locale );
ICalendar calendarAdapter = new CalendarAdapter( calendar );
com.ibm.icu.text.DateFormat dateFormat =
calendar.getDateTimeFormat( com.ibm.icu.text.DateFormat.FULL, -1, locale );
IDateFormat dateFormatAdapter = new DateFormatAdapter( dateFormat );
DateChooser datechooser = DateChooser.getInstanceWithLocale( calendarAdapter,
dateFormatAdapter, locale );
datechooser.setName("datechooser");
datechooser.setWidth(300);
model.add( datechooser );
%>
</blox:container>
</body>
</html>
```

相关概念

第 185 页的『创建非阳历日历』

International Components for Unicode（ICU）是一组为获取 Unicode 支持而广泛使用的 Java 库。它提供的功能部件和可扩展性支持非阳历日历和不同日期格式。使用 ICU 包，您可以轻松地创建自己的日历对象和日期格式化程序。

相关任务

第 185 页的『在启动日历时指定所选日期』

缺省情况下，在启动日历时，将显示当前月份的日历并选择了当前日期。可以指定除当前日期之外的不同初始日期。

在启动日历时指定所选日期

缺省情况下，在启动日历时，将显示当前月份的日历并选择了当前日期。可以指定除当前日期之外的不同初始日期。

要指定不同的初始日期，请使用可以支持指定所选日期的 DateChooser.getInstanceWithDateLocale() 方法。以下示例创建阳历并将所选日期设置为 2005 年 1 月 1 日。

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="com.alphablox.blox.uimodel.core.*,
  com.alphablox.blox.uimodel.*,
java.text.DateFormat,
java.util.Date,
java.util.Calendar,
java.util.Locale,
java.util.GregorianCalendar"%>
<%@ taglib uri="bloxtld" prefix="blox"%>
<html>
<head>
<blox:header/>
</head>
<body>
<blox:container id="dateChooserContainer">
<%
BloxModel model = dateChooserContainer.getBloxModel();
Locale locale = Locale.US;
java.util.Calendar calendar = java.util.Calendar.getInstance( locale );
ICalendar calendarAdapter = new CalendarAdapter( calendar );
java.text.DateFormat dateFormat =
java.text.DateFormat.getDateInstance( java.text.DateFormat.FULL, locale );
IDateFormat dateFormatAdapter = new DateFormatAdapter( dateFormat );
GregorianCalendar mydate = new GregorianCalendar(2005, Calendar.JANUARY, 01);
Date d = mydate.getTime();
DateChooser datechooser = DateChooser.getInstanceWithDateLocale( d,
calendarAdapter, dateFormatAdapter, locale );
model.add( datechooser );%>
</blox:container>
</body>
</html>
```

创建非阳历日历

International Components for Unicode（ICU）是一组为获取 Unicode 支持而广泛使用的 Java 库。它提供的功能部件和可扩展性支持非阳历日历和不同日期格式。使用 ICU 包，您可以轻松地创建自己的日历对象和日期格式化程序。

如果您的应用程序需要支持多个语言环境，则应使用 ICU 中的 Calendar 和 DateFormat 对象，而不是使用 java.util.Calendar 和 java.text.DateFormat 包。

非阳历日历示例

在创建非阳历日历之前，您应遵循第 181 页的『创建阳历』中所描述的步骤在页面上添加基本日历控件。

以下步骤演示如何在 JSP 文件中创建日语日历控件。您可以修改这些步骤以创建伊斯兰语、希伯来语或中文日历控件。

要创建非阳历日语日历控件:

1. 将下列各包添加到 import 语句:

- java.util.Date
- java.util.Locale

您将使用 ICU 包中的日历和 DateFormat 类。因为 ICU 随 DB2 Alphablox 一起交付，所以不必导入它们。您的 import 语句现在类似于以下内容:

```
<%@ page import="com.alphablox.blox.uimodel.core.*,
  com.alphablox.blox.uimodel.*,
java.util.Date,
java.util.Locale"%>
```

2. 创建日语 CalendarAdapter。CalendarAdapter 对象必须实现 ICalendar 接口。

```
com.ibm.icu.util.Calendar japaneseCalendar =
new com.ibm.icu.util.JapaneseCalendar( Locale.JAPAN);
ICalendar japaneseCalendarAdapter =
new CalendarAdapter( japaneseCalendar );
```

3. 创建 DateFormatAdapter 对象。从日历中获取日语数据格式并将它应用到 DateFormatAdapter。

```
com.ibm.icu.text.DateFormat japaneseDateFormat =
japaneseCalendar.getDateTimeFormat( com.ibm.icu.text.DateFormat.FULL,
-1, Locale.JAPAN );
japaneseDateFormat.setCalendar( japaneseCalendar );
IDateFormat japaneseDateFormatter = new DateFormatAdapter( japaneseDateFormat );
```

如果未将同一日历应用到格式化程序，则文本字段将缺省为使用英语格式。

4. 将日语语言环境应用到 DateChooser。

```
DateChooser datechooserJAPAN =
new DateChooser( japaneseCalendarAdapter,
japaneseDateFormatter, Locale.JAPAN );
```

以下是完整的示例:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="com.alphablox.blox.uimodel.core.*,
  com.alphablox.blox.uimodel.*,
java.text.DateFormat,
java.util.Date,
java.util.Locale"%>
<%@ taglib uri="bloxtld" prefix="blox"%>
<html>
<head>
<blox:header/>
</head>
<body>
<blox:container id="dateChooserContainer">
<%
BloxModel model = dateChooserContainer.getBloxModel();
// Create your Japanese CalendarAdapter
com.ibm.icu.util.Calendar japaneseCalendar =
new com.ibm.icu.util.JapaneseCalendar();
ICalendar japaneseCalendarAdapter =
new CalendarAdapter( japaneseCalendar );
// Apply the Japanese date format to your DateFormatAdapter
com.ibm.icu.text.DateFormat japaneseDateFormat =
japaneseCalendar.getDateTimeFormat( com.ibm.icu.text.DateFormat.FULL,
-1, Locale.JAPAN );
japaneseDateFormat.setCalendar( japaneseCalendar );
IDateFormat japaneseDateFormatter = new DateFormatAdapter( japaneseDateFormat );
// Apply the same locale to the chooser
DateChooser datechooserJAPAN = new DateChooser(japaneseCalendarAdapter,
japaneseDateFormatter, Locale.JAPAN );
datechooserJAPAN.setName("datechooserJAPAN");
model.add(datechooserJAPAN);
```

```
%>
</blox:container>
</body>
</html>
```

重要:

- 如果您使用 ICU 包，则对于中文日历，您必须显式地设置毫秒。否则，字段将不会进行正确的计算。

  ```
  com.ibm.icu.util.Calendar chineseCalendar =
  new com.ibm.icu.util.ChineseCalendar();
  chineseCalendar.setTimeInMillis((new Date()).getTime());
  ICalendar chineseCalendarAdapter =
  new CalendarAdapter(chineseCalendar);
  ```

- 缺省情况下，显示方向被设置为从左到右。对于诸如希伯来语或阿拉伯语的双向语言，您必须设置 DateChooser 的方向，如下所示:

  ```
  myDateChooser.setBidiDirecton(DateChooser.RTL);
  ```

日历控件的字体

日历控件中文本的字体在 DB2 Alphablox 存储库的每个主题的 CSS 文件中指定。指定的字体为 Lucida Sans Unicode、Arial、Helvetica 和 sans-serif。因为 Lucida Sans Unicode 是支持多个语言环境的字体的最佳选择，所以缺省字体设置为 Lucida Sans Unicode。设计它的目的是为了支持在 Unicode 标准的版本 2.0 中定义的最常用字符，它随 Windows 2000 和 Windows XP 一起提供。

如果用户尚未安装此字体（这种情况很少见），并且他们的语言环境设置为一种使用 Arial、Helvetica 和 sans-serif 不支持的 Unicode 代码点的语言，则日历控件中的文本可能不会正确显示。一种解决方案是在其系统上安装 Lucida Sans Unicode 字体。也可以通过搜索 Lucida Sans Unicode 并替换它来更改 CSS 文件中指定的字体。然而，应始终将它设置为支持 Unicode 的字体并确保用户在他们的系统上安装了该字体。

计算的成员

计算的成员是一些数据成员，它们包含动态生成的数据，这些数据是通过对结果集中实际存在的成员执行计算派生的，接着，这些数据显示在新创建的行或列中。某些数据源（如 DB2 OLAP Server、Hyperion Essbase 和 Microsoft Analysis Services）可以使用它们的查询语言、Essbase 报告规范语言和 Microsoft 多维表达式（MDX）语言来生成计算的成员。但是，您通常无法与这些计算的成员进行交互。例如，在使用从查询派生的计算的成员的立方体中，执行上寻或下寻将导致把计算的成员的名称重新提交至数据库。由于这些成员本来并不存在于数据源中，所以这些查询将失败。

DB2 Alphablox 提供了内置的功能来根据数据源返回的结果集创建计算的成员并且对交互进行管理，因此，用户可以与数据进行交互和使用这些计算的成员，就好象它们是真实的成员一样。

在 DB2 Alphablox 中创建计算的成员

在 DB2 Alphablox 应用程序中，使用 DataBlox 的 calculatedMembers 属性来创建计算的成员。使用 DB2 Alphablox 计算的成员的一项重要优点是，它们允许将新数据添加到结果集中，而不必修改实际数据源。当您无法等待数据库更改或者仅仅想对从现有数据派生的新度量进行试验时，使用计算的成员就特别有益。下列示例提供了计算的成员对于用户来说可能很有用的一些情况:

- 计算两个成员（如 Budget 和 Actual）的值之间的差别以及差别百分比
- 计算指定的维（如 DollarSales）上的所有成员的平均值
- 计算销售总量的百分比

要了解有关 DataBlox calculatedMembers 属性的语法和用法详细信息，请参阅 *Developer's Reference* 一书。在下列各节中，提供了几个关于计算的成员的重要主题。

定制计算准则

当使用定制计算时，您应该了解下列准则和限制:

定义定制计算

在定义定制计算时，下列准则适用:

- 计算的成员的定义是一个算术表达式，此算术表达式是按照正常的数学优先顺序规则进行求值的。
- 要对计算的成员进行定位，您可以指定一个参考成员，在网格中，计算的成员应该在该参考成员之前出现。否则，计算的成员将在现有维的末尾出现。
- 当定义了多个要定位在同一个参考成员之前的计算的成员时，计算的成员将具有以下顺序: 定义中的最后一个计算的成员与参考成员最接近。
- 必须根据该维的现有已显示成员来定义计算的成员。
- 当在一个维上定义了多个计算的成员而没有进行定位时，将按照定义顺序将那些维添加至维。
- 计算的成员的定义表达式可以使用先前已定义的计算的成员（向后引用），但不能使用尚未定义的计算的成员（向前引用）。

定制计算限制

- 为了对计算的成员进行制图，必须将"代过滤器"设置为显示所有的代。为此，可以将 totalsFilter 属性设置为 0，也可以在 DHTML 客户机的"图表选项"对话框中将"代过滤器"设置为"显示所有的代"。
- 可以通过正常书签操作来保存和复原计算的成员。
- 成员名不能包含等号（=）、大括号（{}）或双引号（″）。
- 当使用 Microsoft Analysis Services 或 DB2 Alphablox 数据源时，成员名必须是唯一的。
- 对于 DB2 OLAP Server 或 Essbase，强烈建议您在计算的成员表达式中使用唯一的成员名 - 如果 DataBlox useAliases 属性设置为 false，或者用户在 UI 中禁用了别名，计算就可能会失败。
- 对于给定的成员，如果要将缺少的值或空值视为指定的数值，则在表达式中使用 ifNotNumber 函数来提供特殊情况逻辑。

技巧: 当对 ifNotNumber 函数指定值时，您务必了解计算所产生的值。当在计算中使用时，为一些缺少的值或空值替换值可能是没有意义的。

阻止正确显示数据的情况

下表列示了一些导致无法显示有意义数据的情况以及这些情况发生时对网格显示造成的影响。

情况	网格单元格显示:
除零	计算的成员不显示在网格中
引用结果集中不存在的成员	空字符串（或者 missingValueString 属性中指定的值）
计算表达式无效	错误号（在控制台上或日志中将出现错误）。
引用缺少的值或非数字值	空字符串（或者 missingValueString 属性中指定的值）

计算的成员属性语法

要对成员指定一个或多个定制计算，请使用 DataBlox calculatedMembers 属性（或 setCalculatedMembers 方法）。注意，标记属性语法在一个语句中可以包括多个定制计算:

```
calculatedMembers="dim1:calc1{refMember1:gen:missingIsZero}=
   expr1{scopeDim:scopeMember}, dim2:calc2{refMember2} =
   expr2{scopeDim:scopeMember},..., dimn:calcN
      {refMemberN:gen:missingIsZero} =
exprN{scopeDim:scopeMember}"
```

其中:

- dimN 值是创建计算的成员时使用的维的名称。
- calcN 值是计算的成员的名称。
- [可选] refMemberN 值是一个现有成员的名称，在网格中，计算的成员计算将在此成员之前显示。refMemberN 不能是另一个计算的成员。
- [可选] 对于计算所涉及的成员，如果要将所有缺少的值视为零，则可以使用 definitionString 的 missingIsZero 组件。缺省情况下，将把计算中所有缺少的值都视为缺少那些值。[注意: 此关键字仅影响使用成员变量的计算。它对计算函数不起作用。]
- exprN 值是一个算术表达式，它涉及 dim 的成员。可以用函数 ifNotNumber 来替换成员值以提供特殊情况逻辑，以便处理计算所使用的结果集缺少值或包含空值的情况。

  ifNotNumber 函数具有以下语法:

  ```
  ifNotNumber(memberName, value)
  ```

  其中:

  - memberName 是要执行该函数的成员的名称。
  - value 是一个数值，它替换缺少的成员值或空成员值。指定的值一定不能包含逗号。

可以对计算的成员使用的函数

下表概述了可以在您使用的计算表达式中使用的算术函数、搜索函数、特殊计算函数以及与缺少的值相关的函数。要了解有关语法和用法的详细信息，请参阅 *Developer's Reference* 一书的 DataBlox 一节。

算术函数

	描述
Abs	返回一个成员的绝对值。只能对单个数字项（如另一项计算的结果或者单个成员）使用此函数。
Average	返回定义的所有数字的平均值，即总和除以个数。
Count	返回定义中的所有数字的个数。缺少的值将被忽略。如果没有任何值可供计数，则返回零。
Max	返回定义中的所有数字中的最大值。
Median	返回一个集合的中间值。
Min	返回定义中的所有数字中的最小值。
Product	返回定义中的所有值的乘积。
Round	返回一个数字在舍入到最接近的整数后的整数部分；只能对单个数字项（如另一项计算的结果或者单个成员）使用此函数。
Sqrt	返回一个数字的平方根；只能对单个数字项（如另一项计算的结果或者单个成员）使用此函数。
Stdev	返回定义中的所有数字的标准偏差。
Sum	返回定义中的所有数字的累加和；缺少的值被忽略。如果没有任何值可供累加，则将返回零。
Var	返回方差，即集合中的每个数与平均值的偏差的平方的平均值。

搜索函数

	描述
Child	返回指定的成员的所有子代。
Descendants	返回指定的成员的所有后代。
Leaf	返回指定的成员的所有叶级后代。

特殊计算函数

	描述
Rank	按升序或降序返回指定维的指定成员的值。
RunningTotal	返回指定维中的指定成员的数值累积和。

与缺少的值相关的函数

	描述
ifNotNumber	可以用来提供特殊情况逻辑以处理计算所使用的结果集中缺少的值或空值。

可以在 *Developer's Reference* 一书的 DataBlox 一节的 calculatedMembers 属性描述中找到这些算术函数、搜索函数、特殊计算函数以及与缺少的值相关的函数的语法和用法详细信息。

计算的成员示例

下列示例演示计算的成员的一些常见用法:

- 使用单元格值来定义定制计算，此定制计算显示实际值与预算值之间的差别百分比:

```
calculatedMembers = "Scenario:Variance % = (Actual-Budget)/Budget*100"
```

- 使用单元格值来定义定制计算，此定制计算显示实际值与预算值之间的差别百分比，并提供逻辑来为 Actual 替换值 1,000,000，为 Budget 替换值 5,000（在指定数字时，不要使用逗号）:

```
calculatedMembers="Scenario:Variance %=(ifNotNumber(Actual,1000000)-
ifNotNumber(Budget,5000))/ifNotNumber(Budget,5000) * 100"
```

- 定义一个定制计算，此定制计算显示某年份前两季度的销售量:

```
calculatedMembers = "Year: YTD = \"Q1,2000\" + \"Q2,2000\""
```

- 将两个定制计算组合到一个属性中（其中，Scenario 基于一个维，Year 基于另一个维）:

```
calculatedMembers="Scenario:Variance % =(Actual-Budget)/Budget*100,
Year: YTD = \"Q1,2000\" + \"Q2,2000\""
```

- 将两个定制计算组合到一个属性中，并且为不同表达式中使用的同一个成员替换不同的值:

```
calculatedMembers ="Scenario:Variance % =
(Actual-ifNotNumber(Budget, 10000))/ifNotNumber(Budget, 10000) * 100,
Scenario: Difference = Actual-ifNotNumber(Budget, 0)"
```

- 使用单元格值来定义定制计算，此定制计算显示实际值与预算值之间的差别百分比。将计算的成员 Variance % 定位为在网格中位于成员 Actual 之前。

```
calculatedMembers = "Scenario: Variance % {Actual} =
(Actual-Budget)/Budget * 100"
```

- 要在每个组中添加独立的排名，可以使用 Rank 函数，如以下示例所示:

```
calculatedMembers="All Products:Rank =
Rank(All Products,All Locations,2,DESC,GROUPDIM)
```

注: 要清除计算的成员，请将空字符串传递给 setCalculatedMembers。

还可以通过对计算指定计算的成员的作用域或者指定代号来对同一个轴上嵌套的维或者对计算中的计算执行计算，这有助于对计算的成员进行定位。有关 calculatedMembers 属性及相关方法的更多用法详细信息以及示例，请参阅 *Developer's Reference* 一书的 DataBlox 一节。

使用 Essbase 报告脚本命令计算的成员

也可以使用 Essbase 报告脚本命令来创建计算的成员。虽然这在某些情况下可能很有用，但是，如果不生成 DB2 OLAP Server 或 Essbase 错误消息以指示计算的成员不存在（这些成员在 DB2 OLAP Server 或 Essbase 立方体中实际上不存在），就不能与显示的结果数据进行交互。例如，在对包含计算的成员的网格进行钻取时，钻取操作和其他操作将被禁用。作为首选替代方法，您应该尽可能使用 DB2 Alphablox 来创建计算的成员。

要了解关于使用 Essbase 报告脚本命令来创建计算的成员的详细信息，请参阅 DB2 OLAP Server 或 Essbase 文档。并且，务必查阅第 112 页的『DB2 Alphablox 支持的 Essbase 报告脚本命令』。

第 19 章 过滤数据

本主题讨论有关为用户过滤数据的技巧，过滤数据有助于更有效地处理大型结果集、限制对信息的访问或者将用户看到的信息个性化。

隐藏维和成员

通过访问数据，用户就能够创造性地提出问题以及创造性地比较数据，但是，可用信息量有时会显得过多。对于判断力欠佳的最终用户来说，数据过多会导致他们迷失方向。对于精明的用户来说，大量的数据实际上会对他们产生“干扰”，从而使他们无法把注意力集中到关键数据上。作为开发者，您需要时刻考虑面向的用户并开发使用户能够获得适当信息量的分析应用程序。如果用户对“数据布局”面板、“成员过滤器”对话框和完整数据列表具有全面的访问权，最终将会导致他们花费大量的时间来使用隐藏／显示以及保留／除去功能以便在窗口中显示所需的信息。但是，作为开发者，您可以通过使用 DataBlox 属性过滤掉与所执行的任务无关的数据或者很少使用的数据来为他们提供帮助。

借助 DB2 OLAP Server 和 Hyperion Essbase 中的属性维以及 Microsoft Analysis Services 中的虚拟维，企业能够通过许多以前不能采用的方法来将数据切片。例如，在 QCC 样本数据源中，您可以通过每包块数分组或每包盎司数分组来分析巧克力销售情况（而不考虑那些巧克力是否是果仁巧克力）、按产品的推出日期来列示产品或者在考虑占地面积的情况下分析库存。当您想要将这些信息用于分析时，这种情况很理想。但是，如果您不想那样做的话，那些信息就很多余。幸运的是，DB2 Alphablox 提供了两个属性来使您能够方便地隐藏维和成员，这两个属性就是 DataLayout `hiddenDimensionsOnOtherAxis` 属性和 DataBlox `hiddenMembers` 属性。

您可以使用 DataLayoutBlox `hiddenDimensionsOnOtherAxis` 属性来列示不想在 DataLayout 面板中显示的维。要获取示例，请参阅“Blox 样本程序”的“过滤数据”部分。要了解有关 `hiddenDimensionsOnOtherAxis` 的语法和用法详细信息，请参阅 *Developer's Reference* 一书的 DataLayoutBlox Reference 一节。

有时，可以使用 DataBlox `hiddenMembers` 属性来隐藏没有显示意义的成员。例如，在 Scenario 维中，顶级成员是 Scenario，但是，此成员实际上仅仅是一个存储区，它用来存放它下面的一组子成员。并且，在 DB2 OLAP Server 和 Hyperion Essbase 中，Scenario 实际上显示它下方的第一个子代的数据 - 但是，Scenario 成员没有数据。在“Blox 样本程序”的“过滤数据”部分中，一个隐藏 Scenario 成员的示例（用于 DB2 OLAP Server、Essbase 和 Microsoft Analysis Services）显示了 Scenario 的子代。注意，当您对其中一个成员执行上寻时，实际上将显示 Scenario（即使它被列示为隐藏的成员）。但是，当您在 Scenario 中下寻时，它将再次消失。由于钻取操作具有一些执行方式方面的限制，所以此行为是不可避免的。要了解有关 `hiddenMembers` 的语法和用法详细信息，请参阅 *Developer's Reference* 一书的 DataBlox Reference 一节。

使用 dimensionRoot 属性

要过滤信息或者控制对信息进行的访问，其中一种最简单的办法是使用 DataBlox `dimensionRoot` 属性来指定维中的特定成员以将那些成员用作用户的虚拟根。一旦将特

定维根成员定义为新的“虚拟”根，就可以防止用户上寻到位于所定义的根上方的成员。此设置将应用于“成员过滤器”中显示的页面过滤器、行和列以及维成员列表。对于限制他人访问您不想让人访问的数据区域，虚拟根可能非常有用。此属性可以用来提供有限的安全性，或者仅仅用来帮助防止用户迷失在数据中。

您可能正在使用数据库安全性来防止用户看到他们不应该访问的数据值，但是，有时仅仅查看成员名就有可能表示共享的信息过多。在有限的此类情况下，dimensionRoot 可能很有用。例如，如果一个数据库列示了整个国家或地区内的潜在客户，那么，这个数据库就有可能向即将离开贵公司而加入竞争对手并且心怀不满的雇员提供有用的信息。如果最大程度地降低对此类信息的访问对您来说非常重要，则 dimensionRoot 属性可以提供一个有用的选项。

注: 由于您只想防止用户访问私有信息，所以不应该使用 dimensionRoot 属性。此外，对于同一数据源，总是可以使用其他数据库工具来访问您使用 DataBlox 中定义的虚拟根保护的信息。

您的主要目标也可以是通过阻塞与特定用户需求无关的信息路径来改进可用性，从而防止那些用户在信息路径中四处钻取而最终迷失在数据中。如果某些信息与他们需要执行的任务无关，则使用 dimensionRoot 将有助于最大程度地减少此类“迷失在导航空间中”问题。

设置用户的虚拟根

在以下示例中，对用户进行限制以使他们只能够访问关于某国内的特定地区的信息。这些简单的步骤将允许您配置一个 DataBlox，以便为用户创建与他们的工作国家或地区相关的“虚拟”根:

1. 为独立的或嵌套的 DataBlox 添加 dimensionRoot 属性。

   ```
   <blox:data id="myDataBlox"
   dimensionRoot=""
   ...
   </blox:data>
   ```

2. 添加一个维，然后在该维中添加一个要用作用户“虚拟”根的成员。

   例如，如果要限制对 QCC 数据库中 East 地区的访问，则 dimensionRoot 设置应该类似于:

   ```
   dimensionRoot="All Locations:East"
   ```

3. 保存页面并进行测试。

这是一个非常简单的示例，此示例对 dimensionRoot 设置进行了硬编码。通过让 dimensionRoot 设置基于 DB2 Alphablox 存储库中存储的值，也可以在页面装入时动态地设置 dimensionRoot。定制用户属性可以将数据源中应该出现的区域名用作属性值。

注: 在此类情况下，为了安全起见，您有可能想禁止用户进行编辑，以防止用户更改他们自己的地区并获取对其他地区的访问权。

注: 要了解有关如何为用户和应用程序创建定制用户属性的指示信息，请参阅《*管理员指南*》。

一旦使用 DB2 Alphablox 配置了定制用户属性，您就可以使用服务器端 Java 方法来访问该值。在带有 DataBlox 的 JSP 页面上，可以在 DataBlox 标记中或者使用相关的

setDimensionRoot() 方法来动态地配置 dimensionRoot 设置。如果在 JSP 页面上的 DataBlox 标记前面包括 RepositoryBlox，则可以使用 JSP 表达式语句来将值动态地包括在 DataBlox 标记中。代码将类似于以下示例:

```
dimensionRoot="<%= myRB.getUserProperty("userRegion") %>"
```

其中，myRB 表示在此设置前面定义的 RepositoryBlox 的名称，该 RepositoryBlox 可以从 DB2 Alphablox 存储库中获取值。在本示例中，定制用户属性是 userRegion。这种非常简单的方法也适用于其他属性。

固定选项列表

通常，DB2 Alphablox 应用程序中的页面过滤器毫无限制地允许用户对维信息进行上寻和下寻，并且限制了对“成员过滤器”的访问。dimensionRoot 属性允许您通过定义一个根来限制信息访问（用户无法转到这个根的下方）。但是，有的时候，更好的选择是通过设置 PageBlox fixedChoiceLists、moreChoicesEnabledDefault 和 moreChoicesEnabled 属性创建固定选项列表来限制用户可以选择的选项数目。

示例: “Blox 样本程序”应用程序的“过滤数据”部分提供了一个“固定选项列表”示例，该示例显示了如何使用这些 PageBlox 属性中的其中两个属性。

使用 fixedChoiceLists 属性

PageBlox fixedChoiceLists 属性将指定的维和成员放在页面过滤器的下拉列表中以供用户选取。与正常的页面过滤器不同，固定选项列表将用户选项限制为您指定的那些选项。fixedChoiceLists 的缺省值是一个空字符串，这使用户能够访问所有维及其成员。当使用此属性指定了维和特定成员之后，用户就只能访问您定义的成员。例如，要将用户限制为只能查看 Central 和 East 这两个地区，PageBlox fixedChoiceLists 属性将类似于以下示例:

```
fixedChoiceLists="All Locations:Central,East"
```

即使初始查询未包括任何一个固定选项列表成员，该固定选项列表中指定的维的顶级成员最初也将显示在列表中。在用户选择了其中一个固定选项列表成员之后，顶级成员就会从列表中消失。

为了让固定选项列表显示在 PageBlox 中，您还必须记得在 DataBlox 的 selectableSlicerDimensions 属性中指定该列表的维。在我们的示例中，DataBlox（而不是 PageBlox）属性将类似于:

```
selectableSlicerDimensions="All Locations"
```

注: 如果初始查询包含固定选项中某个维的多个成员，则固定选项列表页面过滤器就不会显示在 PageBlox 中。例如，如果初始查询是:

```
query='<SYM <ROW ("All Products") <ICHILD "All Products"
<COL(Scenario) <CHILD Scenario "2001" Central East !'/>
```

则固定选项列表就不会显示在 PageBlox 中。要使此查询能够起作用，在查询中应该只包括缺省情况下要显示在固定选项列表中的成员（即 Central 或 East）。

使用 moreChoicesEnabledDefault 和 moreChoicesEnabled 属性

缺省情况下，DataBlox moreChoicesEnabledDefault 属性允许用户对页面过滤器选择“更多选项”。要禁用此项缺省功能，请将此属性设置为 false，如下所示:

```
moreChoicesEnabledDefault="false"
```

此外，可以使用此属性的更具选择性的版本，即 moreChoicesEnabled 属性。此选项要求您指定一些维，对于这些维，不显示“更多选项”菜单选项。

要了解有关 moreChoicesEnabled 属性的两个变体的详细信息，请参阅 *Developer's Reference* 一书。

使用 MemberSecurityBlox 来过滤成员

可以使用 Blox 逻辑标记库中包括的 MemberSecurityBlox 来根据访问权对维成员列表进行过滤。通过使用 DataBlox suppressNoAccess 属性，MemberSecurityBlox 根据指定的 MemberSecurityFilter 值来禁止对成员进行访问。此属性可以接受多个根成员，并且，它允许指定多对 dimension:member 来进行过滤。

要获取 MemberSecurityBlox 的用法示例，请参阅第 48 页的『使用 MemberSecurityBlox 来列示立方体成员』。要了解关于 MemberSecurityBlox 的语法和用法详细信息，请参阅 *Developer's Reference* 一书的 Business Logic Blox and TimeSchema DTD Reference 一节。

使用 HTML 表单元素和 FormBlox 组件

尽管 DB2 Alphablox 提供了许多属性来对表示 Blox 组件中的信息传递进行配置和过滤，但是，您还可以将部分功能移到这些 Blox 外部的 Web 页面上。可以使用 HTML 表单元素（包括选择列表、复选框和单选按钮）以及 FormBlox 来访问信息和进行选择。通过除去 Blox 菜单栏和工具栏并加入 HTML 表单元素（这些 HTML 表单元素使用 DHTML 客户机 API 的 JavaScript 方法来调用服务器端逻辑），您可以创建功能强大并且简单易用的应用程序来供任何 Web 用户使用。

除了使用标准 HTML 表单元素和 Blox 客户机 API 以外，还可以使用 Blox 表单标记库（包括 FormBlox 组件）来构建分析应用程序。FormBlox 将生成定制的表单元素，这些表单元素具有构建分析应用程序所必需的常用功能。例如，DimensionSelectFormBlox 和 MemberSelectFormBlox 将生成自动填充了维名和成员名的 HTML 选择列表，并且可以用来创建简单列表以供用户选择。FormBlox 组件的另一项好处是，他们保持状态，即使用户离开了页面然后在浏览器会话中返回至该页面亦如此。

通过将标准 HTML 表单元素与 FormBlox 组件配合使用，可以限制用户交互功能以及过滤掉不想向用户显示的选项。要了解有关可用的 FormBlox 组件的详细信息，请参阅第 41 页的第 6 章，『Blox 表单标记库』以及 *Developer's Reference* 一书的 Blox Form Tags Reference 一节。

使用查询

在向用户显示数据之前，最有效的数据过滤方法可能是构造优良的查询语句。通过使用查询来只返回与正在执行的任务相关的数据，可以避免大型结果集（大型结果集在数据库服务器上需要较长的时间才能生成，并且会影响网络流量）。

解释如何为特定数据源优化查询并不属于本指南的范畴。请您查阅数据库文档和其他资源以了解有关使用查询语句来过滤数据的详细信息。并且，本指南第 109 页的第 14 章，『检索数据』提供了有限的查询技术信息，这些信息对于您从受支持的多维数据库和关系数据库中过滤信息可能有用。

使用 Blox 属性来消除数据

DB2 Alphablox 提供了几个能够提高分析应用程序的可用性和性能的属性。下列各节简要描述了如何使用 DataBlox 的 suppressMissingOnRows、suppressMissingOnColunns、suppressZeros、suppressDuplicates 和 suppressNoAccess 属性。要了解有关这些属性的语法和用法的更多详细信息，请参阅 *Developer's Reference* 一书的 DataBlox Reference 一节。

虽然消除数据在大量数据行或列完全包含零或缺少数据的情况下是有意义的，但是，消除此信息也可能会对企业用户造成误导，例如，该用户实际上需要知道缺少数据的情况以便对此采取措施。如果用户能够访问 Blox 上的菜单栏，他们就可以在"数据选项"对话框中手工地更改此设置，但是，他们可能不了解此设置，或者可能不会想到缺少了他们所不了解的未消除数据。如果菜单栏不可用，并且您想消除数据，则请考虑在页面上设置表单元素（如复选框或单选按钮），以便允许用户控制此设置并认识到已消除了数据。

使用 suppressMissingOnRows 和 suppressMissingOnColumns 属性

当返回的行或列中完全没有数据时，suppressMissingOnRows 和 suppressMissingOnColumns 属性将从网格中除去那些行或列。如果数据集中的某行或某列中的任何单元格包含值，则整行或整列都可视。

要启用此功能，请对 DataBlox 添加 suppressMissingOnRows 和 suppressMissingOnColumns 属性并将值设置为 true，如下所示:

```
suppressMissingOnRows="true"
suppressMissingOnColumns="true"
```

通过使用 *Developer's Reference* 一书的 DataBlox Reference 一节中列示的相关 Java 方法，还可以有规划地来控制此功能。

对于 DB2 OLAP Server 和 Essbase 数据源，当通过在 DataBlox 中将属性设置为 true 启用了 suppressMissingOnRows 或 suppressMissingOnColumns 时，将对数据库服务器和 DB2 Alphablox 都执行消除。以下内容概述了使用 DB2 OLAP Server 或 Essbase 时应该预期的行为:

- 如果初始查询是报告脚本（而不是书签），则 DB2 Alphablox 将消除缺少的数据。

- 如果该查询是书签、钻取或旋转的结果，则将请求 DB2 OLAP Server 或 Essbase 服务器消除缺少值的行。

仅仅依赖于 Essbase 报告脚本命令 <SUPPRESSMISSING 通常不是最佳的解决方案，这是因为，DB2 Alphablox 在这种情况下并不会除去由钻取或其他操作产生的缺少的数据。

当最终用户正在使用 DHTML 客户机时，在初始报告脚本命令中添加 <SUPPRESSMISSING 命令很有可能不会对性能产生显著的影响，即使查询返回大型结果集（超过 1000 行）时亦如此。DHTML 客户机将对它检索到的结果集进行优化，从而将其限制为只包含在特定情况下可以被查看的内容。

注: 使用 GridBlox missingValueString 属性或其相关方法来指定在未包含值的单元格中应该显示的内容。当 DataBlox suppressMissing 属性或 Essbase <SUPPRESSMISSING 报告脚本命令未消除整行或整列时，此属性非常有用。

要了解有关 missingValueString 属性的详细信息，请参阅 *Developer's Reference* 一书的 DataBlox Reference 一节。

使用 suppressZeros 属性

当 DataBlox suppressZeros 属性设置为 true 时（缺省值为 false），将消除所有只包含零的行或列。如果数据集中的某行或某列中的任何单元格包含非零值，则将显示整行或整列。

要启用此功能，请对 DataBlox 添加 suppressZeros 属性并将值设置为 true，如下所示:

```
suppressZeros="true"
```

通过使用 *Developer's Reference* 一书的 DataBlox 一节中列示的相关 Java 方法，还可以有规划地来控制此功能。

使用 suppressDuplicates 属性

当设置为 true（缺省设置）时，DataBlox 的 suppressDuplicates 属性将从网格的行或列中除去所有重复的头值。

如果您不想消除重复的头值，则对 DataBlox 添加 suppressDuplicates 属性并将值设置为 false，如下所示:

```
suppressDuplicates="false"
```

通过使用 *Developer's Reference* 一书的 DataBlox Reference 一节中列示的相关 Java 方法，还可以有规划地来控制此功能。

注: 要在初始查询中消除重复的 DB2 OLAP Server 或 Essbase 共享成员，请在查询语句中使用 <SUPSHARE 报告脚本命令。要了解有关消除 DB2 OLAP Server 或 Essbase 共享成员的更多信息，请参阅 DB2 OLAP Server 或 Hyperion Essbase 文档。

第 20 章 使数据持久以及为数据建立书签

数据和视图的持久性对于分析应用程序来说是一个重要的注意事项。虽然应用程序访问的大部分数据都存储在数据库中，但是，可以使用 DB2 Alphablox 来为应用程序状态、书签和定制属性管理数据持久性。下面提供了使用 JavaServer Pages 技术来在会话期间实现数据值持久性的简要讨论和示例。

DB2 Alphablox 中的数据持久状态

DB2 Alphablox 应用程序是资源的集合。DB2 Alphablox 提供了内置的功能来允许用户保存和复原各种书签及应用程序的状态。例如，在进行钻取和旋转或者选择首选图表布局之后，用户可以对当前视图建立书签以便将来取回。并且，根据已定义的应用程序设置的不同，可以保持应用程序状态并在会话结束（用户关闭浏览器窗口或者当前会话超时）时自动存储该状态。

可以公用形式或专用形式保存书签和已保存的应用程序状态。公用书签和应用程序状态可以由所有能够访问该应用程序的用户共享。要了解有关通过 Blox 用户界面使用此功能的信息，请参阅用户帮助页面。下列各节讨论关于书签和应用程序状态的更多详细信息，并列示了可供开发者用来管理书签和应用程序状态的 Blox 属性和方法。

应用程序状态

应用程序状态是 DB2 Alphablox 采用的另一种保存应用程序信息的方法。当用户启动访问应用程序的会话时，DB2 Alphablox 将创建该应用程序的实例。只要该会话处于工作状态，此实例就保持用户的当前应用程序会话状态，包括应用程序资源的状态，如查询结果集、网格和图表外观、排序以及用户所作的其他更改。

在 DB2 Alphablox 应用程序中，**应用程序状态**表示的是该应用程序中所有 Blox 在特定时刻的状态。在使用 DB2 Alphablox 应用程序期间，DB2 Alphablox 将跟踪并保持用户的当前应用程序状态。此状态被定义为**当前应用程序状态**。此外，**保存的应用程序状态**表示的是应用程序中所有 Blox 在保存应用程序状态的特定时刻具有的状态。

虽然书签保存各个 Blox 的状态，但是，应用程序状态保存的是整个应用程序的状态。可以将应用程序状态保存下来，将来可以根据需要将其复原。并且，可以公用形式保存应用程序状态以供所有能够访问应用程序的用户共享，也可以专用形式为每个用户保存应用程序状态。

应用程序状态管理由 DB2 Alphablox 自动处理。DB2 Alphablox 自动地把当前应用程序状态保存在存储库中:

- 用户（通过关闭浏览器）退出应用程序
- 用户会话超时（缺省情况下，在不活动时间达到 15 分钟时，用户会话将超时）

当用户下次访问该应用程序时，如果应用程序定义中的**复原应用程序状态**设置设置为是（缺省设置为否），则 DB2 Alphablox 将复原最近保存的应用程序状态。如果**复原保存的应用程序状态**被设置为否，则用户将访问最初的状态（即缺省应用程序状态）。您可以在应用程序页面上提供一个定制按钮，以允许用户复原上次保存的状态。

注: 在会话超时之后，如果用户尝试在浏览器窗口中使用该会话，则会显示一条消息，该消息指示用户按浏览器的“刷新”（或“重新装入”）按钮以连接至 DB2 Alphablox。

注: DB2 Alphablox 不会保存应用程序状态数据。当保存的应用程序状态被复原时，应用程序将从数据库中检索新数据。

下表列示了与应用程序状态管理相关的 RepositoryBlox 方法:

Java 方法

`delete()`
`deleteApplicationState()`
`exists()`
`getApplicationStateNameAndDescription()`
`list()`
`load()`
`rename()`
`renameApplicationState()`
`restoreApplicationState()`
`save()`
`saveApplicationState()`
`search()`

DB2 Alphablox 存储库中的定制属性

通过使用 DB2 Alphablox 存储库，您可以创建定制用户属性、定制组属性和定制应用程序属性，并且可以使用标准的 JSP 方法来检索或修改这些属性。在用户定义、组定义和应用程序定义中创建这些定制属性之后，就可以使用 RepositoryBlox 来存储和检索这些定制属性。通过使用 JavaServer Pages 技术，可以将这些属性值作为运行时表达式值替换到 Java 代码中以及 Blox 标记属性中。根据 DB2 Alphablox 属性继承层次结构来提供这些定制属性。

注: 如果您要将 Blox 添加至门户网站应用程序，则请记住，portlet 依靠门户网站基础结构来访问用户概要文件信息。因此，您应使用 Portlet API 而不是通过 DB2 Alphablox 来获取用户信息。

创建定制用户属性

要了解如何使用定制属性，请考虑 ChartBlox `chartType` 属性。缺省值是 `3D Bar`，但对于一组财务应用程序来说，CFO 可能希望使用折线图。开发者可以定义定制属性来为此用户指定另一个缺省值。定制属性名将与 Blox 属性名相同（`chartType`），并且将具有 `Line` 值。

基于本示例，下列步骤将创建必需的定制用户属性:

1. 从 DB2 Alphablox 主页中，选择“管理”选项卡。
2. 单击“服务器”链接，然后选择“定制属性”下方的“用户定义”。“用户定义”页面打开。
3. 单击页面底部的“创建”按钮。“创建用户定制属性”页面显示。
4. 填写下列条目:
   - 属性名: `chartType`
   - 缺省值: `Line`

- 值列表：Line，3D Bar

5. 单击“保存”以保存新属性。接着，您将对用户的定义指定这个定制属性。
6. 从 DB2 Alphablox 主页中，选择“管理”选项卡。
7. 单击“用户”链接。“用户定义”页面打开，该页面显示了现有用户定义列表。
8. 要定义新用户，请单击“创建”按钮。“创建用户”页面显示，该页面显示了“一般属性”面板。（要对现有用户定义指定定制属性值，请从列表中选择用户名并单击“编辑”按钮。）
9. 提供（或编辑）“用户名”、“密码”和“确认密码”条目。
10. chartType 属性将显示在“一般属性”面板底部。确保属性值设置为 Line。
11. 单击“保存”。
12. 现在，通过作为刚刚创建的用户登录来测试属性。

JavaServer Pages 技术和数据持久性

JavaServer Pages 技术提供了数个用来在用户会话期内以及在会话之间管理数据值的方法。JavaServer Pages 技术提供了数种不同的数据值存储和检索方法，这些方法包括 URL 重写、隐藏表单值、request 对象方法和 session 对象方法。请参阅 JavaServer Pages 书籍或其他 JSP 资源以获取可以在基于 JSP 的应用程序中使用的技术的描述。

在以下任务中，您将了解一个示例，该示例使用了上述技术中的其中一种技术：request 对象 getParameter 方法。

使用 Request 参数来检索 URL 属性值

在提交 Web 页面时，URL 地址可以传递信息，在链接的页面中，可以检索该信息。在 DB2 Alphablox 应用程序中，一种常见的用法是使用页面上的 Blox 视图来创建定制打印页面。通过使用 DB2 Alphablox URL render 属性，通过类似于以下的行就可以打开新的打印页面：

```
window.open("view-print.jsp?render=printer","_blank");
```

通过使用这个 JavaScript 方法，将打开新的浏览器窗口，该窗口将显示以 HTML 格式显示（以便于打印）的当前 Blox 视图。但是，如果在所显示的页面上使用 HTML 表单元素（按钮和复选框等）以及文本，则所有那些元素和文本都将包括在可打印的页面上。这不是理想的解决方案。

但是，您可以创建定制打印页面，此页面将从 Blox 视图中检索元素并将它们合并到定制打印页面中。下列步骤显示了如何使用 getParameter 方法来在 JSP 页面之间传递值：

1. 在带有要打印的 Blox 视图的页面上，创建一个 JavaScript 函数，此函数将构造一个 URL，此 URL 用来将信息传递给定制打印页面。

   以下是一个 JavaScript 函数示例，此函数创建一个 URL 地址，从而传递时间、地区（基于 HTML 选择列表中选择的值）、显示方式和 HTML 主题：

   ```
   function printPreview() {
     var   region=document.RegionForm.RegionSelectionList.
             options[document.RegionForm.RegionSelectionList,
             selectedIndex].text;
     var timestamp=new Date();
   ```

```
var URL="passingValues-print.jsp?Region="+escape(region)+
"&TimeStamp="+escape(timestamp.toString())+
"&render=printer"+
"&theme=printer";

window.open(URL,"PrintPreviewWindow");
}
```

2. 在定制打印页面中，捕获 URL 查询字符串中的值并根据需要将它们合并到页面中。

   在定制打印页面的主体中，以下示例显示了使用 request 对象的 getParameter 方法来将时间和地区放到页面上:

```
<h1>Sales for <%= request.getParameter("Region") %></h1>
<p>
<blox:display bloxRef="RegionPresentBlox"/>
</p>
<h3><%= request.getParameter("timestamp") %></h3>
```

注: 在“Blox 样本程序”示例集中，“持久化和建立书签”部分中的“在页面之间传递值”示例演示了 request 参数的用法。

书签 - 开发者细节

作为开发者，在处理书签时，您应该了解一些重要的细节:

- 书签是属性集（“名称 - 值”对）的集合，它用来复原 Blox 的状态。以下是 Blox 属性的属性集示例:
  - `dividerLocation = 0.25`
- 书签不仅包括 Blox 属性，还包括书签属性。这些书签属性包括:
  - 应用程序
  - Blox 类型（Present、Chart 和 Grid 等等）
  - 描述
  - 书签名
  - 隐藏（布尔）
  - 对 Blox 属性的引用
- 书签被保存后，它就是初始 Blox 状态与保存书签时 Blox 具有的当前状态之间的**差别**。**初始状态**包括缺省值以及已定义的标记和标记属性特性。在 DB2 Alphablox 存储库中，只有属性已被更改为与初始状态不同的 Blox 才有具有属性集的书签。
- 保存书签时，将根据应用程序名、Blox 名、书签类型（公用、组和专用）以及组名或用户名把书签保存到 DB2 Alphablox 存储库中的特定位置，以后，将从该位置复原它。
- 书签查询文件包括一个序列化的 Query 对象，后者与不带数据的网格结果集（即，成员对象的元组）非常相似。这不是文本查询，文本查询表示的是以初始 Blox 状态定义的查询。
- 通过使用书签过滤器，可以为书签创建上下文过滤器，上下文过滤器对于创建页面上带有单个 Blox 的应用程序以及根据书签菜单项来设置视图来说非常有用。并且，通过使用共享书签，可以创建“已发布的”应用程序和自助应用程序。

可以在 *Developer's Reference* 一书的 Common Blox Reference 和 BookmarksBlox Reference 这两节中找到有关公共 Blox 属性、BookmarksBlox 标记和标记属性以及可用的服务器端 API 的详细信息。

获取所有书签的计数

此示例演示:

- 使用 BookmarksBlox 及其 `listBookmarks()` 方法来访问存储库中存储的所有书签。`listBookmarks()` 方法返回书签对象数组。
- 如何通过获取数组长度来获取书签总数

```
<%@ taglib uri="bloxtld" prefix="blox" %>
<!--import the following package in order to access the
    com.alphablox.blox.repository.Bookmark class-->
<%@ page import="com.alphablox.blox.repository.*" %>
<blox:bookmarks id="myBookmarksBlox" />
<%
   Bookmark bks[] = null;
   bks = myBookmarksBlox.listBookmarks();
%>
There are <%= bks.length %> bookmark(s).
```

获取书签的属性集

此示例演示如何根据书签名、应用程序名、用户名、Blox 名和书签可视性来访问书签以及获取有关它的属性集的信息。特别是，此示例演示了:

- 使用 BookmarksBlox 来访问各个书签（Bookmark 对象）
- 使用 Bookmark 对象的 `getName()`、`getVisibility()`、`getDescription()`、`getBloxType()` 和 `getBinding()` 方法
- 使用 Bookmark 对象的 `getBookmarkProperties()` 方法来访问各个属性（每个嵌套的 Blox 一个属性）

生成的输出类似于:

The bookmark you are looking for exists.

1. The Repository JNDI binding for this bookmark is:
2. The bookmark name is: q2fy02WestSales
3. The type of Blox this bookmark was saved for is: grid
4. The bookmark description is: The Q2 West Sales
5. The bookmark visibility is: private
6. The bookmark contains Blox properties in the repository
7. Types of Blox properties saved in the bookmark:
   - grid
   - data

```
<%@ page import="com.alphablox.blox.repository.*,
                 com.alphablox.blox.ServerBloxMissingResourceException,
     com.alphablox.blox.ServerBloxException,
com.alphablox.blox.BookmarksBlox" %>
<%@ page import="java.util.*" %>
<%@ page import="java.io.*" %>
<%@ taglib uri="bloxtld" prefix="blox"%>
<html>
<head>
  <!-- Blox header tag -->
   <blox:header/>
</head>
```

```
<body>
<!-- Get an SSPM BookmarkBlox -->
<blox:bookmarks id="bookmarks" />
<ol>
<!--getting the bookmark you want-->
<%
String bookmarkName = "q2fy02WestSales";
String applicationName = "SalesApp";
String userName = "admin";
String bloxName = "myGrid";
   String visibility = Bookmark.PRIVATE_VISIBILITY;
   Bookmark bk = null;
      try {
      bk = bookmarks.getBookmark(bookmarkName, applicationName,
         userName, bloxName, visibility);
   } catch (ServerBloxException e){
      if (e instanceof ServerBloxMissingResourceException){
         %><li><b>The bookmark DOES NOT EXIST!</b></li><%
}
  else
        throw e;
}
   if (bk != null)  {
      %><li><b>The bookmark you are looking for exists.</b></li>
   <li><b>The Repository JNDI binding for this bookmark is:<br>
         <%= bk.getBinding() %></b></li>
   <li><b>The bookmark name is: <%= bk.getName() %></b></li>
   <li><b>The type of Blox this bookmark was saved for is: <%=
      bk.getBloxType() %></b></li>
   <li><b>The bookmark description is: <%= bk.getDescription() %>
      </b></li>
   <li><b>The bookmark visibility is: <%= bk.getVisibility() %>
      </b></li>
   <!---Getting the individual BookmarkProperties ---->
<%
      BookmarkProperties props[] = bk.getBookmarkProperties();
      if (props != null) {
         %><li><b>The bookmark contains Blox properties in the
         repository</b><br>
         Types of Blox properties saved in the bookmark:
         <ul>
<%
         for (int i = 0; i < props.length; i++) {
            %><li><%= props[i].getType() %></li><%
}
         %></ul><br></li><%
}
      else {
         %><li><b>The bookmark DOES NOT CONTAIN Blox properties in the
         repository</b></li><%
}
}%>
 </body>
</html>
```

使用服务器端 bookmarkLoad 事件过滤器

此示例演示如何使用服务器端事件过滤器来在 bookmarkLoad 事件被触发时执行定制任务（在本示例中，我们将一条消息添加至控制台）。

1. 要使用服务器端事件过滤器，首先使用公共 Blox 方法 addEventFilter() 来添加特定事件过滤器对象。

   ```
   <% myPresent.addEventFilter(new LoadFilter()); %>
   ```

2. 然后，编写您自己的类以实现相应的事件过滤器对象（BookmarkLoadFilter）以及该事件被触发时将调用的相应方法 bookmarkLoad(BookmarkLoadEvent)。

```
  public class LoadFilter implements BookmarkLoadFilter
{
   public void bookmarkLoad( BookmarkLoadEvent bre )
{
      //actions to take when the event is triggered
}
}
```

代码如下所示:

```
<%@ page import="com.alphablox.blox.filter.*" %>
<%@ page import="com.alphablox.blox.*" %>
<%@ page import="com.alphablox.blox.repository.Bookmark" %>
<%@ taglib uri="bloxtld" prefix="blox"%>

<html>
<head>
  <title>Bookmarks Filter Events</title>
  <!-- Blox header tag -->
   <blox:header/>
</head>

<%!
public class LoadFilter implements BookmarkLoadFilter {
public void bookmarkLoad( BookmarkLoadEvent ble )
throws Exception {
Bookmark bookmark = ble.getBookmark();
String name = bookmark.getName();
System.out.println("A bookmark called " + name + " is
loaded.");
}
}
%>

<body>
<blox:present id="myPresent" >
<blox:data dataSourceName="TBC"
query="<Row(Market) <ICHILD Market <Column(Year) Year !"/>
<%
myPresent.addEventFilter(new LoadFilter());
%>
</blox:present>

 </body>
</html>
```

使用 BookmarksBlox API 来定制应用程序

BookmarksBlox 与其详尽的 API 一起允许您有规划地来创建和管理书签以及动态地设置书签属性。例如，通过动态地修改与书签存储在一起的数据查询，可以创建时序报告或始终访存当前季度数据的报告。可以使用定制书签属性来存储每个用户选择的报告布局或者实现您自己的安全性。在更改了数据源成员名或轮廓的情况下，可以修改与书签存储在一起的查询。您甚至可以创建自己的书签管理用户界面。

要使用 BookmarksBlox API，请在页面中添加 BookmarksBlox。这使您能够以 Bookmark 对象形式访问每个书签。

以下是“Blox 样本程序”中包括的几个有趣的书签定制示例。

使用书签事件

共有 4 个可以在 DB2 Alphablox 应用程序中使用的书签事件: load、save、rename 和 delete。可以对 Blox 注册这些事件的任何组合，包括注册多个具有相似类型的事件。一旦被注册，这些事件都将在实际过程启动前被调用，从而使您有机会定制书签行为。

典型的事件类似于:

```
public class LoadFilter implements BookmarkLoadFilter {
   public void bookmarkLoad( BookmarkLoadEvent ble ) throws Exception {
    Bookmark bookmark = ble.getBookmark();
        String name = bookmark.getName();
System.out.println("Bookmark " + name + " applied");
}
}
```

在此示例中，事件获取正在装入的书签的名称，然后将该名称显示在控制台中。

使用已注册的事件

注册事件的操作是在 Blox 标记内完成的，如下所示:

```
<blox:present id="myPresent3" >
<blox:data
dataSourceName="QCC-Essbase"
query="!" />
<%
   myPresent3.addEventFilter(new LoadFilter());
   myPresent3.addEventFilter(new SaveFilter());
   myPresent3.addEventFilter(new RenameFilter());
   myPresent3.addEventFilter(new DeleteFilter());
%>
</blox:present>
```

以上 Blox 标记注册全部四个书签事件。

将动态查询与书签配合使用

借助新增的书签 API，您可以指示服务器执行一个查询（此查询与最初和该书签保存在一起的查询不同）以及将结果用作结果集。

以下示例（此示例使用 Microsoft Analysis Services MDX 查询语句）显示了如何修改书签以存储参数化的文本查询，当书签被装入时，将使用该查询。强制书签使用另一个文本查询的操作包括保存具有书签属性的查询以及将 textualQueryEnabled 属性设置为 true。

在书签的 save 事件中，可以执行下列操作以保存参数化的查询:

```
// Parameterized query (NOTE: :year and :quarter)

final String PARAM_QUERY = "SELECT {[Products].[Category].[All Products],
   [Products].[Category].[All Products].children} ON ROWS,
   {[Time].[Calendar].[All Time Periods].[:year],
   [Time].[Calendar].[All Time Periods].[:year].[:quarter]}
ON COLUMNS FROM [QCC]";
// get the Bookmark Object from the BookmarkSaveEvent
Bookmark bookmark = bse.getBookmark();
// Find DataBlox properties for this bookmark
```

```
BookmarkProperties data =
   bookmark.getBookmarkPropertiesByType(Bookmark.DATA_BLOX_TYPE);
// If DataBlox properties not found in existing property set, create
if (data == null) {
  data = bookmark.createBookmarkProperties(Bookmark.DATA_BLOX_TYPE);
}
// Set textualQueryEnabled to true, saving the query above to bookmark
data.setProperty("textualQueryEnabled", true);
data.setProperty("query", PARAM_QUERY);
```

装入书签时，可以使用书签 `load` 事件来将参数替换为由用户给定的相关信息，例如:

```
// get the Bookmark Object from the BookmarkLoadEvent
    Bookmark bookmark = ble.getBookmark();
// find a DataBlox properties for this bookmark
BookmarkProperties data =
   bookmark.getBookmarkPropertiesByType(Bookmark.DATA_BLOX_TYPE);
if (data != null) {

// Get the parameterized query from the bookmark
String query = data.getProperty("query");
// Replace the parameters with real information

// NOTE: replaceText simply replaces any references to the 2nd argument
// with the contents of the third argument.

query = replaceText(query, ":year", "2002");
query = replaceText(query, ":quarter", "Qtr2");
// set the new un-parameterized query
data.setProperty("query", query);
}
```

当书签被装入时，将为 `2002` 和 `Qtr2` 交换参数，并且将执行该查询。

获取与指定的条件相符的书签列表

此示例演示:

- 使用 BookmarkMatcher 对象来获取指定用户（在本示例中，是用户“admin”）的书签
- 使用 Bookmark 对象的 `getBinding()` 和 `getBloxType()` 方法及其输出

生成的输出如下所示:

Got 5 Bookmark Object(s) for user admin.

The Bookmarks are:

- users/admin/salesapp/salesgrid/bookmark/salesq1fy03/properties (grid)
- users/admin/salesapp/salespresent/bookmark/eastq2fy03/properties (present)
- users/admin/budgetapp/mypresent/bookmark/eastq3budget/properties (present)
- users/admin/budgetapp/mypresent/bookmark/westq3budget/properties (present)
- users/admin/budgetapp/present2/bookmark/mybudget/properties (present)

代码如下所示:

```
<%@ taglib uri="bloxtld" prefix="blox" %>
<!--import the following package in order to access the
    com.alphablox.blox.repository.BookmarkMatcherUsers class-->
<%@ page import="com.alphablox.blox.repository.*" %>
<html>
<head>
   <blox:header/>
```

```
</head>
<body>
<blox:bookmarks id="myBookmarksBlox" />
<%
   Bookmark bks[] = null;
   BookmarkMatcherUsers matcher = new BookmarkMatcherUsers();
   bks = null;
matcher.setUser("admin");
   bks = myBookmarksBlox.listBookmarks(matcher);
%>
   <div>Got <%= bks.length %> Bookmark Object(s) for
        user <%= matcher.getUser() %></div>
   <div>The Bookmarks are:</div><br>
<%
   for (int i = 0; i < bks.length; i++) {
%><%= bks[i].getBinding() %> (<%= bks[i].getBloxType() %>)<br>
<%
}
   %></div>
 </body>
</html>
```

当书签被装入时，以文本格式获取 DB2 OLAP Server 或 Essbase 序列化的查询

本示例演示如何以文本格式从书签中获取序列化的查询（此查询与 DataBlox 中的查询不相同）。注意，本示例只能对 DB2 OLAP Server 和 Essbase 数据源起作用。要引用 Microsoft Analysis Services，您需要自己保存查询。

1. 将 DataBlox 的 textualQueryEnabled 属性设置为 true:

   ```
   <blox:data...
   textualQueryEnabled="true" />
   ```

2. 使用了服务器端事件过滤器 BookmarkLoadFilter 来在书签被装入时触发定制操作。请参阅第 227 页的『使用远程 PDF 处理器』以获取服务器端事件过滤器示例。

3. 当书签被装入时，将以文本格式检索序列化的查询。

完整的代码如下所示:

```
<%@ page import="com.alphablox.blox.filter.*,
      com.alphablox.blox.repository.BookmarkProperties,
      com.alphablox.blox.repository.SerializedQuery,
      com.alphablox.blox.repository.SerializedTextualQuery,
      com.alphablox.blox.repository.SerializedMDBQuery,
com.alphablox.blox.repository.Bookmark" %>
<%@ taglib uri="bloxtld" prefix="blox"%>
<html>
<head>
   <blox:header/>
<%!
  public class LoadFilter implements BookmarkLoadFilter
{
   public void bookmarkLoad( BookmarkLoadEvent ble ) throws Exception
{
    Bookmark bookmark = ble.getBookmark();
     SerializedQuery sq = bookmark.getSerializedQuery();
     SerializedTextualQuery stq = null;
      SerializedMDBQuery smq = null;
      String query = null;
     if( sq instanceof SerializedTextualQuery )
{
      stq = (SerializedTextualQuery)sq;
```

```
       query = stq.getQuery();
}
        else if( sq instanceof SerializedMDBQuery )
{
           smq = (SerializedMDBQuery)sq;
            query = smq.generateQuery();
}
System.out.println("query=" + query);
}
}
%>
<body>
<blox:present id="myPresent"
width="800"
height="600">
<blox:data
dataSourceName="QCC-Essbase"
query='<ROW ("All Locations") Central East West
<COLUMN ("All Time Periods") 2001 !"
useAliases="true"
textualQueryEnabled="true" />
<%
  myPresent.addEventFilter(new LoadFilter());
%>
</blox:present>
 </body>
</html>
```

使用定制属性来限制访问

定制属性是对现有书签的增强。现在，您可以将附加的键／值信息放到书签中，当该书签被装入时，在书签的 load 事件发生时将使用那些信息。

在书签的 save 事件中，可以在书签中添加定制属性，例如:

```
// get the Bookmark Object from the BookmarkSaveEvent
Bookmark bookmark = bse.getBookmark();
// add username of bookmark owner as a custom property
bookmark.setCustomProperty("Owner", "Admin");
```

注: 可以为 BookmarkSaveEvent 类或者该类的任何其他书签事件创建构造函数，以接收一个参数（如所有者名称）。

装入书签时，可以使用书签 load 事件来获取定制属性并了解所有者是否匹配:

```
// get the Bookmark Object from the BookmarkLoadEvent
Bookmark bookmark = ble.getBookmark();

// get the owner custom property
String owner = bookmark.getCustomProperty("Owner");

// compare this user and the owner
if (!owner.equalsIgnoreCase(currentUser)) {
    // if user and owner do not match, stop bookmark load
    ble.cancelEvent();
}
```

还可以执行同一操作以停止删除不归当前用户所有的书签:

```
// get the Bookmark Object from the BookmarkDeleteEvent
Bookmark bookmark = bde.getBookmark();
// get the owner custom property
String owner = bookmark.getCustomProperty("Owner");
// compare this user and the owner
```

```
if (!owner.equalsIgnoreCase(currentUser)) {
// if user and owner don't match, stop bookmark delete
    bde.cancelEvent();
}
```

第 21 章 分发视图

您可以通过电子邮件和建立书签来共享分析视图。

尽管大部分的分析工作由坐在办公室或书房内的人单独完成，但是，分析信息和结果通常与企业内的别人共享，那些人包括执行官、同事和客户。幸好有了无处不在的因特网和 Web 浏览器，您可以与办公室中的或者远在全球任何位置的公司职员共享 DB2 Alphablox 应用程序。在以下主题中，讨论了用于分发和共享信息的电子邮件、建立书签和打印方法。

使用电子邮件 bean 创建邮件链接

可以使用电子邮件 bean 来通过电子邮件将静态数据视图发送给一个或多个电子邮件收件人。Application Studio 的电子邮件示例提供了这个 bean 以及一组 JSP 支持文件和图像。此示例的核心文件是 EmailBean.class 文件，您需要将此文件包括在您的应用程序中。

要使用这个 bean，需要对 DB2 Alphablox 指定 SMTP 服务器。可以通过 DB2 Alphablox 管理页面的“管理”选项卡的“系统”链接完成此工作。

下面提供了所包括的步骤的一般概述。要了解有关为应用程序配置和定制文件的逐个步骤的详细指示信息，请参阅现成的示例。

1. 第一步是将 Java 类文件复制到应用程序中。特别是，需要将 EmailBean.class 文件以及另外两个支持类文件（HTMLFileParser.class 和 HTMLFile.class）复制到应用程序的 WEB-INF\classes\alphablox\ 目录中。所有 Java 类、servlet、bean 或其他实用程序类都需要在 WEB-INF\classes\ 中。在本实例中，我们在 classes\ 下面创建名为 alphablox\ 的子目录。
2. 下一步是将下列文件复制到应用程序目录中:
   - emailSend.jsp
   - emailError.jsp
   - emailTemplate.jsp
   - emailDialog.html

   本节末尾的表列示了每个文件的用途。您可能希望修改或定制 emailError.jsp、emailTemplate.jsp 和 emailDialog.html。该表还提供了建议的修改操作。
3. 示例实现包括许多图像以及一个样式表。您可能还希望修改和 / 或使用这些文件。如果使用图像，请将它们复制到应用程序目录的 images 子目录中。可以将样式表（styles1.css）直接复制到应用程序目录中。请参阅本节末尾的表。
4. 示例包括一个名为 emailExample.jsp 的文件。您可以使用此文件来了解如何将电子邮件功能合并到应用程序中。在 emailExample.jsp 中，定义了一个名为 openEmailDialog() 的 JavaScript 函数。此函数调用电子邮件对话框。还添加了代码，因此，当您单击示例中的按钮时，将调用 openEmailDialog 函数。

文件	描述	修改
emailExample.jsp	这个 JSP 文件包含: • 要通过电子邮件发送的用户界面 Blox • 电子邮件链接或按钮，它触发电子邮件功能	将此文件中的一个 JavaScript 代码块复制到您的包含 Blox 的 JSP 文件中，该代码块在一个可调整大小的独立浏览器窗口中显示 emailDialog.html。 在您的电子邮件链接或按钮中，指定要调用所复制的 JavaScript 函数。
emailDialog.html	emailExample.jsp 调用的 HTML 文件；它包含一个表单，这个表单用来填写电子邮件消息的发件人、收件人、主题和主体。 在提交表单后，将调用 emailSend.jsp 文件并通过表单发布来传递所有参数。	您可以按原样使用此文件，也可以针对应用程序来修改标题、徽标或样式表引用。
emailSend.jsp	这个 JSP 文件与电子邮件 bean 进行交互以发送电子邮件。	不要修改此文件。
emailTemplate.jsp	返回的页面，它通知用户已发送电子邮件。	您可以按原样使用此文件，也可以针对应用程序来修改标题或文本。
emailError.jsp	emailSend.jsp 的错误页面。如果在尝试发送电子邮件时发生了错误，则将在此页面中显示错误信息。	您可以按原样使用此文件，也可以针对您的环境来定制此文件。
emailBlox.gif	在 emailExample.jsp 中用作电子邮件图标的“邮箱”图像。	您可以按原样使用此文件，也可以进行修改。
required.gif	一个红色的小箭头，它指示电子邮件对话框中的必需字段。	您可以按原样使用此文件，也可以进行修改。
gridlogo-sm.gif	Alphablox 徽标，它显示在电子邮件对话框中的“发送电子邮件”按钮左边。	您可以使用此徽标，也可以将其替换为您自己的图像。
grid-bg.gif	此图像以平铺方式显示以形成电子邮件对话框的背景。	您可以使用此徽标，也可以将其替换为您自己的图像。
style1.css	emailDialog.html 使用的样式表。	您可以按原样使用此文件，也可以进行修改。

书签

可以使用书签来与已定义小组中的别人或者与公众（具有应用程序访问权的人）共享包含新数据的 Blox 视图实例。书签允许分析员和管理员快速地共享定制数据视图，而不要求他们等待开发者有空创建定制应用程序视图。而是，通过对视图建立书签并与别人共享那些书签，小组的所有成员都可以共享信息。

小组也可以使用书签来共享已保存的共享视图，这些共享视图有可能作为完全定制视图被添加到应用程序中。

注: 由于建立了书签的视图将一直使用在几个月后可能无效的成员名，所以，建议不要对长期使用的视图使用书签。并且，在不修改书签的情况下，对数据源添加的新成员可能不会在书签视图中反映出来。

注: 有时，用户会变得关心书签，而误解了实际保存的内容。保存书签时，并不会保存任何数据。每次打开书签视图时，都将把适当的查询重新提交至服务器并检索新数据。并且，建立了书签的视图不会允许别人访问数据库安全性所不允许他们访问的信息。

要了解有关在开发分析应用程序时可以使用的书签功能的更多信息，请参阅第 222 页的『使用 <blox:pdfReport> 标记的定制 PDF 报告属性』。要了解有关与书签相关的可用属性和方法的详细信息，请参阅 *Developer's Reference* 一书的 Common Blox Reference 和 BookmarksBlox Reference 这两节。

打印

DB2 Alphablox 的其中一个优点是，向用户显示的数据是立即可用的，当数据源更新时，那些数据也会更新。并且，您可以共享那些数据，而不必将它们打印出来以及通过公司电子邮件分发它们。但是，用户不可避免地想要打印副本以便与别人共享。请参阅本指南的其他章节以了解更多有关如何有效地通过打印和 PDF 显示功能来传递分析视图的信息:

- 第 138 页的『打印机格式（render=printer）』
- 第 139 页的『PDF 格式（render=pdf）』
- 第 140 页的『打印 Blox 输出』
- 第 218 页的『以 PDF 格式导出』

第 22 章 导出数据

导出数据是一种以电子表格格式或其他格式输出数据以便共享、归档或进行进一步计算的方式。本主题讨论如何创建支持以 Microsoft Excel 或 XML 格式导出 Blox 视图中的数据的应用程序。

以 Excel 格式导出数据

缺省情况下，工具栏包括一个**以 Excel 格式导出**按钮，菜单栏包括**文件 → 以 Excel 格式导出**选项，它们允许用户以本机 Excel 格式导出当前视图中的数据。

要使用此“以 Excel 格式导出”选项:

- 用户必须有 Microsoft Office 2000 或 Office XP。
- 当提示启用宏时，用户必须启用宏。如果未启用宏，则仅显示网格数据。将不会生成图表。

当用户单击**以 Excel 格式导出**按钮时，一个对话框打开并提示用户选择模板。DB2 Alphablox 提供两个模板: **缺省模板**和**使用图表数据**。您可以提供自己的模板并使它们可以通过对话框中的模板选择列表进行选择。用户可以选择其中一个模板，也可以选择不使用任何模板。

Excel 的缺省模板

缺省情况下，DB2 Alphablox 提供两个模板: **缺省模板**和**使用图表数据**。

下表描述了每个模板的行为和优点。用户也可以选择不使用任何模板。还描述了此方案的行为和优点。

模板	行为	优点
缺省值	使用网格中的数据通过宏在 Excel 中生成图表。如果用户在 Excel 中更改网格数据，则该图表将自动更新。然而，Excel 中的结果图表可能与在 DB2 Alphablox 中显示的图表稍有不同。 **注**: 因为没有网格数据可用于生成独立 ChartBlox 中的图表，所以不会导出该图表。	图表始终与网格同步。
使用图表数据	写入图表数据的单独 **chartData** 工作表并使用该数据生成图表。如果 Excel 中的网格数据已更改，则除非用户更改单独的 **chartData** 工作表中的数据，否则该图表不会自动更新。	因为 Excel 中的结果图表是使用 DB2 Alphablox 中的图表数据生成的，所以该图表类似于显示在 DB2 Alphablox 中显示的图表。

模板	行为	优点
不使用模板	在返回的页面上将 MIME 类型设置为 `application/vnd.ms-excel`。`application/vnd.ms-excel` 是 Microsoft Excel 的标准 MIME 类型。它触发浏览器以启动 Excel 应用程序来装入该页面。该页面本身不使用本机 `xls` 格式。 **注:** 未导出图表。它适用于独立的图表和嵌套在 PresentBlox 中的图表。	没有嵌入的宏。

要创建自己的模板并使它们可以供用户在选择以 Excel 格式导出数据时通过模板选择列表进行选择，请参阅『创建定制 Excel 模板』。

创建定制 Excel 模板

您可以创建自己的模板以便以 Excel 格式导出数据。要创建自己的模板，请使用 DB2 Alphablox 提供的属性编写您的宏。例如，您可以定制列标题样式或缺省单元格样式。编写定制 Excel 模板要求了解 Microsoft Visual Basic for Applications。

“以 Excel 格式导出”选项的模板存储在 DB2 Alphablox 存储库的 templates 目录中。每个模板都必须存储在单独的文件夹中。templates 目录具有以下结构:

文件或目录	描述
templates.xml	用于指定缺省所选模板的 XML 文件。
template_dir	每个模板都必须放置在目录中。目录内应该有两个文件: Excel XLT 模板文件和 template.xml 文件。template.xml 文件描述两条信息: • 模板的显示名称，它将显示在用户界面的模板选择列表中。 • 同一目录中要用于此模板的 XLT 文件的名称。

要创建定制模板:

1. 在存储库的 templates 目录下创建一个目录。
2. 编写您自己的模板。DB2 Alphablox 提供了一组您可以在 Excel 宏中使用的属性。提供的这些属性由两个提供的模板使用。您可以使用这两个提供的模板作为示例来创建自己的模板。请参阅第 217 页的『Excel 模板的属性』以获取属性列表。
3. 将 Excel 模板（`.XLT` 文件）存储在 DB2 Alphablox 存储库的 templates 目录下您自己的目录中。
4. 通过将 template.xml 文件从现有的 default 或 useChartData 模板目录复制到您自己的目录中来为您的目录创建 template.xml 文件。您的模板文件夹现在包含两个文件: 模板 XLT 文件和 template.xml。
5. 修改您刚复制到模板文件夹的 template.xml 文件:
   a. 在 `<displayName>` 标记中指定模板显示名称。例如，`<displayName>My Custom Template<displayName>`.
   b. 在 `<file>` 标记中指定用于此模板的 XLT 文件。例如，`<file>myCustomTemplate.xlt</file>`

您的模板现在将出现在“以 PDF 格式导出”对话框的模板选择下拉列表中。要修改提供的联机帮助以包括有关您的定制模板的信息，请参阅第 235 页的『创建定制用户帮助』。要将您的模板设置为缺省所选模板，请参阅『设置用于以 Excel 格式导出的缺省模板』。

设置用于以 Excel 格式导出的缺省模板

当用户选择将其数据以 Excel 格式导出时选择的缺省模板为**缺省模板**。此模板在 DB2 Alphablox 存储库的 templates 目录中 `templates.xml` 文件中指定。

- 要将缺省模板设置为名为 My Template 的定制模板，请将 `<default>` 标记中的模板名称修改为:

  ```
  <default>My Template</default>
  ```

- 要不将任何模板设置为缺省模板，请将 `<default>` 标记中的模板名称修改为:

  ```
  <default>noTemplate</default>
  ```

如果 `templates.xml` 文件中缺少 `<default>` 标记，或者文件本身就不存在，则使用 templates 这个目录中包含 XLT 文件的第一个目录。

Excel 模板的属性

下表列示了 DB2 Alphablox 为了在 Excel 模板中使用而提供的属性。这些属性及其值都已导出到 Excel 中名为“properties”的单独工作表中。

属性名	设置者	描述
`WorkSheet`	应用程序开发者	DB2 Alphablox 导出数据的表索引。缺省值为 0。
`StartCell`	应用程序开发者	DB2 Alphablox 启动导出进程的单元格位置。缺省值为 A1。
`EndCell`	DB2 Alphablox	DB2 Alphablox 写入的最后一个单元格。此属性可以用来设置图表的边界。
`ColumnHeaderStyle`	应用程序开发者	要在列标题上使用的样式。此单元格的格式化、颜色、字体和对齐设置已导出数据的列标题样式。
`RowHeaderStyle`	应用程序开发者	要在行标题上使用的样式。此单元格的格式化、颜色、字体和对齐设置已导出数据的行标题样式。
`DefaultCellStyle`	应用程序开发者	要在单元格上使用的样式。除了在单元格上设置了单独的格式或单元格提醒时之外，此单元格的格式化、颜色、字体和对齐设置已导出数据的单元格样式。
`ExportChartData`	应用程序开发者	此属性告诉 DB2 Alphablox，图表是应使用直接从 Alphablox 图表数据中导出的数据还是应使用网格中的数据。DB2 Alphablox 图表通常是网格数据的子集。如果您想要 Excel 图表的外观类似于 DB2 Alphablox 图表，则将此属性设置为 `true`。然而，图表数据将导出到单独的工作表中。当 Excel 网格中的数据更改时，图表将不会受影响。

属性名	设置者	描述
ChartEndCell	DB2 Alphablox	如果 ExportChartData 设置为 true，则此属性标记图表数据的最后一个单元格并用于设置图表的边界。
ChartTitle	DB2 Alphablox	图表的标题。
ChartFootnote	DB2 Alphablox	图表的脚注。
ChartType	DB2 Alphablox	DB2 Alphablox 图表类型。在宏中使用此属性以将 DB2 Alphablox 图表类型映射至 Excel 图表类型。
ChartY1AxisTitle	DB2 Alphablox	如果在 DB2 Alphablox 中以图表属性的形式设置了一个标题，则它为 y1 轴的标题。
ChartY2AxisTitle	DB2 Alphablox	如果在 DB2 Alphablox 中以图表属性的形式设置了一个标题，则它为 y2 轴的标题。
ChartXAxisTitle	DB2 Alphablox	如果在 DB2 Alphablox 中以图表属性的形式设置了一个标题，则它为 x 轴的标题。
ChartO1AxisTitle	DB2 Alphablox	如果在 DB2 Alphablox 中以图表属性的形式设置了一个标题，则它为 o1 轴的标题。

从 DB2 Alphablox 至 Excel 的图表类型映射

DB2 Alphablox 提供的模板中的宏将 DB2 Alphablox 中的图表映射为 Excel 中最接近的图表类型。

下表显示从 DB2 Alphablox 至 Excel 的图表类型映射。

DB2 Alphablox 中的图表类型	Excel 中的图表类型
条形图	条形图
折线图	折线图
饼图	饼图
区域图	区域图
散点图和泡式图	散点图
雷达图和折线图	雷达图和折线图
雷达图和区域图	雷达图和区域图

所有其他图表将映射为条形图。保留了三维效果。

以 PDF 格式导出

用户可以将分析视图从 Blox 导出到 Adobe Acrobat PDF 文件。PDF 显示格式与标准的基于 Web 的打印相比具有以下优点:

- 应用程序开发者和用户都可以更好地控制布局。可以在多个页面上显示较宽的图表和网格而不会截断它们。
- 可以保存 PDF 文件供将来使用。
- 可以将 PDF 文件通过电子邮件发送给其他用户。

PDF 报告的缺省用户界面选项

缺省情况下，PresentBlox、GridBlox 和 ChartBlox 组件的菜单栏和工具栏提供了“以 PDF 格式导出”选项。当用户选择菜单栏上的**文件 > 以 PDF 格式导出**或单击 Blox 工具栏上的**以 PDF 格式导出**按钮时，缺省的**创建 PDF 报告**对话框窗口打开。在此对话框中，用户可以修改下列设置:

常规设置	选项
方向	横向（缺省值）或纵向
页大小	Letter、Legal、A3 或 A4。缺省值取决于客户机语言环境（例如，对于美国英语，缺省值为 Letter，对于法语，缺省值为 A4）。
主题	服务器上的可用主题的选择列表，并且缺省值与服务器的缺省 HTML 主题（`coleman`）相同。
标题文本	空白文本输入字段
页脚文本	空白文本输入字段

网格设置	选项
所有列的适合页面数	• 网格宽度适合的页面数 • 如果要将数据列分割为多个页面，则它指定是否要重复行标题 • 如果要将数据列分割为多个页面，则它指定在多少列之后应该有一个换页符 根据指定的页面数，将自动在“在后面换页”面板上对用户建议后面应该有换页符的列，以便均匀分割列。用户可以通过在“可用列”面板和“在后面换页”面板之间移动列来更改建议的列。

图表设置	选项
图表大小	**适合页面**（缺省值）或**定制**。当设置为**定制**时，用户可以指定图表的宽度和高度（以像素计）。

您可以定制此对话框，甚至可以选择不让此对话框显示。要了解更多信息，请参阅第 222 页的『使用 <blox:pdfReport> 标记的定制 PDF 报告属性』和第 224 页的『使用 <blox:pdfDialogInput> 标记来定制“PDF 报告对话框”选项』。

创建全局缺省 PDF 报告属性

可以在位于以下目录中的可选 PDF 报告属性文件（`pdfreport.properties`）中定义定制全局缺省 PDF 报告属性:

```
<db2alphablox_dir>/repository/theme/
```

缺省情况下，所有 DB2 Alphablox 应用程序都将使用此文件中的任何设置。在同一目录中提供了示例 PDF 报告属性文件（`example_pdfreport.properties`）。这个示例文件使用的属性与 DB2 Alphablox 中硬编码的属性相同。当您在上面指定的目录中添加 `pdfreport.properties` 文件时，它将覆盖硬编码的值并使用新的全局缺省设置。

要创建缺省的 PDF 报告属性文件，请创建该示例文件的副本，将其重命名为 pdfreport.properties 并修改该文件中的属性以满足您的需求。可以在此文件中指定下列属性:

header 使用以下宏和基于 XML 格式格式定义的页眉（包括文本和布局）类似于 XHTML（请参阅列表下面的注释）。

可用的宏包括:

- 日期: <date/>
- 时间: <time/>
- 页面计数: <totalpages/>
- 当前页: <pagenumber/>
- PDF 对话框输入: <pdfDialogInputN/>（其中 N 是从 1 到 5 的整数）

缺省情况下，<pdfDialogInput1/> 定义页眉，<pdfDialogInput2/> 定义页脚。

示例:

```
header = <table border-bottom='1px' width = '100%'><tr>
<td valign = 'middle'><img src='/AlphabloxServer/theme/i/brand.gif'/></td>
<td align = 'center' style='font: bold 30px Helvetica; color: #333333;'
valign='middle'> <span><pdfDialogInput1/></span></td>
<td align = 'right' style = 'font: 8px Helvetica; color: black;'
valign = 'top'><span/></td>
</tr></table>
```

footer 使用宏和基于 XML 格式格式定义的页脚（包括文本和布局）类似于 XHTML（请参阅列表下面的注释）。

可用的宏包括:

- 日期: <date/>
- 时间: <time/>
- 页面计数: <totalpages/>
- 当前页: <pagenumber/>
- PDF 对话框输入: <pdfDialogInputN/>（其中 N 是从 1 到 5 的整数）。缺省情况下，<pdfDialogInput1/> 定义页眉，<pdfDialogInput2/> 定义页脚。

示例:

```
footer = <table border-top='1px' width='100%'><tr>
<td align='left' style='font: 8px Helvetica; color: black;' valign='bottom'
width='33%'> <span><date/> <time/></span></td>
<td align = 'center' style = 'font: bold 10px Helvetica; color: #333333;'
valign = 'bottom' width = '33%'><span> <pdfDialogInput2/> </span> </td>
<td valign = 'bottom' width='34%'>
<p style = 'font-size:10;align:right;valign:bottom;'>
<pagenumber/> of <totalpages/></p></td>
</tr></table>
```

headerHeight

页眉高度。有效的单位包括: 像素（px）、点（pt）、英寸（in）、毫米（mm）和厘米（cm）。如果未指定此属性，则将使用像素（px）。

示例:

```
headerHeight=50
```

footerHeight
页脚高度。有效的单位包括: 像素（px）、点（pt）、英寸（in）、毫米（mm）和厘米（cm）。如果未指定此属性，则将使用像素（px）。

示例:

```
footerHeight=10
footerHeight=0.5in
```

margin 页边距。有效的单位包括: 像素（px）、点（pt）、英寸（in）、毫米（mm）和厘米（cm）。如果未指定此属性，则将使用像素（px）。

示例:

```
margin=18
```

size 纸张大小，用来定义纸张大小（A3、A4、Letter 和 Legal）以及方向（landscape 和 portrait）。有效属性包括: [A3 | A4 | Letter | Legal | Custom [[Portrait | Landscape] | [width | [height]]]。缺省页面大小是特定于语言环境的: 在美国或加拿大，缺省值为 Letter，否则页面大小缺省值将是 A4。缺省方向为 Landscape。

示例:

```
size=Letter Portrait

size=A4 Landscape

size=Legal

size=Custom 15in 100mm

size=Custom 8in（在这种情况下，使用缺省高度）
```

themeListEnabled
启用主题列表。值可以是 true（缺省值）或 false。

```
themeListEnabled=true
```

pdfDialogInput1
示例: pdfDialogInput1=Header Text

pdfDialogInput2
示例: pdfDialogInput2=Footer Text

repeatPageFilters
对第一页后面的页面重复页面过滤器。

示例: repeatPageFilters=true

theme 主题名，此名称与 DB2 Alphablox 存储库中使用的主题名相同。

示例: theme=my_own_theme

注: 当使用类似 XHTML 的语法来指定页眉和页脚时，格式并不是真正的 XHTML。存在以下局限性。

1. 不支持 。
2. 要产生不中断的间隔，请使用 Unicode 字符（ ），而不是使用 HTML 字符（ ）。
3. CSS 缩写属性需要遵循 W3C CSS 规范。

使用 JSP 标记来定制 PDF 报告

DB2 Alphablox Blox 标记库提供了两个定制 JSP 标记（<blox:pdfReport> 和 <blox:pdfDialogInput>），这两个标记可用来定制 JSP 页面上的 PDF 属性。开发者可以使用 <blox:pdfReport> 标记来指定定制 PDF 报告属性，如 页脚、页眉、页边距和页大小。<blox:pdfDialogInput> 标记用来指定要添加到“创建 PDF 报告”对话框中的输入字段标注和文本字段。

使用 <blox:pdfReport> 标记的定制 PDF 报告属性

开发者可以使用 <blox:pdfReport> 标记来在 Blox 级别或会话级别指定定制 PDF 报告属性（覆盖硬编码的 PDF 报告属性）。要设置仅影响单个 Blox 的 PDF 属性，请在显示为 PDF 时对要应用属性的 Blox 添加嵌套的 <blox:pdfReport> 标记。

要指定应用于同一个 JSP 页面上的所有 Blox 的 PDF 属性，请将 <blox:pdfReport> 标记置于 JSP 页面上的 Blox 外部，这将导致 PDF 属性应用于该页面上的 Blox 的所有 PDF 对话框。

下表描述了在使用 <blox:pdfReport> 标记定义 PDF 属性时可以使用的标记属性:

属性　描述

footer 页脚。它使用 XHTML 标记（请参阅表后面的注解）和宏进行定义。

可用的宏: 日期: <date/>时间: <time/>页面计数: <totalpages/>当前®页面: <pagenumber/>

示例:

```
footer="<table border-top='1px' width='100%'> <tr> <td align='left'
style='font: 8px Helvetica; color: black;' valign='bottom' width='33%'>
<span><date/> <time/></span></td> <td align='center' style='font: bold
10px Helvetica; color: #333333;' valign='bottom' width='33%'> <span>
<pdfDialogInput2/> </span> </td> <td valign='bottom' width='34%'> <p
style='font-size:10;align:right;valign:bottom;'> <pagenumber/> of
<totalpages/> </p> </td> </tr> </table>"
```

footerHeight

页脚高度。有效的单位包括: 像素（px）、点（pt）、英寸（in）、毫米（mm）和厘米（cm）。如果未指定此属性，则将使用像素（px）。

示例:

```
footerHeight="10"
```

```
footerHeight="0.5in"
```

header 页眉。它使用 XHTML 标记（请参阅表后面的注解）和宏进行定义。

可用的宏包括:

- 日期: <date/>
- 时间: <time/>
- 页面计数: <totalpages/>
- 当前页号: <pagenumber/>

示例: header="<table border-bottom='1px' width = '100%'> <tr> <td valign = 'middle'> </td> <td align='center' style='font: bold 30px Helvetica; color: #333333;' valign='middle'> <pdfDialogInput1/> </td> <td align='right' style='font: 8px Helvetica; color: black;' valign='top'> </td> </tr> </table>"

headerHeight

页眉高度。有效的单位包括: 像素（px）、点（pt）、英寸（in）、毫米（mm）和厘米（cm）。如果未指定此属性，则将使用像素（px）。

示例:

```
headerHeight="10"
```

```
headerHeight="1in"
```

margin 页边距。有效的单位包括: 像素（px）、点（pt）、英寸（in）、毫米（mm）和厘米（cm）。如果未指定此属性，则将使用像素（px）。 1in 值将得到具有 1 英寸页边距的页面。

示例:

```
margin="1in"
```

```
margin="40"
```

pageBreak

用于设置换页符的规则。此规则由一组维成员规范组成，后跟关键字 @ before 或 @ after。例如，Dim1: M1, M2; Dim2: M3, M4 @ after。

以上示例指定，只要维 Dim1 的成员 M1 或 M2 或维 Dim2 的成员 M3 或 M4 更改，就会在数据**后面**添加换页符。

使用分号分隔每个维成员规范。使用逗号分隔同一维中的成员。当满足任何一个指定的条件时，将添加一个换页符。关键字 @ before 和 @ after 不区分大小写。如果错误地拼写了关键字，则会抛出异常。

size 纸张大小，用来定义纸张大小（A3、A4、Letter 和 Legal）以及方向（landscape 或 portrait）。有效属性包括: [A3 | A4 | Letter | Legal | Custom [[Portrait | Landscape] | [width | [height]]]。缺省页面大小是特定于语言环境的: 在美国或加拿大，缺省值为: Letter，否则页面大小缺省值将是 A4。缺省方向为 Landscape。

示例:

```
size="Letter Portrait"
```

```
size="A4 Landscape"
```

```
size="Legal"
```

```
size="Custom 15in 100mm"
```

```
size="Custom 8in"（在这种情况下，使用缺省页面大小高度）
```

theme 用于定义布局样式的服务器 HTML 主题。值可以是任何预定义的或定制的 DB2 Alphablox 主题。

示例:

```
theme="coleman"
```

themeListEnabled

启用主题列表。值可以是 true（缺省值）或 false。

示例:

```
themeListEnabled="false"
```

注: XHTML 标记和 CSS 局限性:

1. 不支持 。
2. 要产生不中断的间隔，请使用 Unicode 字符（ ），而不是使用 XHTML 字符（ ）。
3. CSS 缩写属性应该遵循 CSS 规范。

示例:

```
<blox:pdfReport
size="A3 portrait"
margin="30mm" />
---
<blox:pdfReport
size="Letter portrait"
margin="0"
theme="myTheme"
themeListEnabled="false"/>
---
<%
String header="<span style='color:red'>This report has
<totalpages> pages </span>";
%>
<blox:pdfReport
header="<%=header%>"
headerHeight"50px"
footer="<%=some_xhtml_variable%>"
footerHeight"1in"
```

使用 <blox:pdfDialogInput> 标记来定制“PDF 报告对话框”选项

<blox:pdfDialogInput> 标记用来指定要添加到“创建 PDF 报告”对话框中的输入字段标注和文本字段。它只能用作 <blox:pdfReport> 标记中的嵌套标记。

下表描述了 <blox:pdfDialogInput> 标记的可用标记属性以及简要描述:

index 1 到 5 的整数，它定义 5 个字段中要定义的字段。

示例:

```
index="5"
```

displayName

文本的标注。

示例:

```
displayName="Report Header"
```

defaultValue

[可选] 显示在由 displayName 属性定义的文本字段中的缺省字符串。

示例:

```
defaultValue="2004 Revenue Report"
```

示例:

```
<blox:pdfReport>
<blox:pdfDialogInput index="1"
displayName="Report Title"
defaultValue="My Application Name" />
</blox:pdfReport>
<blox:pdfReport>
<blox:pdfDialogInput index="1"
displayName="Report Title"
defaultValue="My Application Report" />
<blox:pdfDialogInput index="2"
displayName="Footer"
defaultValue="My Application Report" />
</blox:pdfReport>
```

为多个 Blox 组件创建 PDF 文件

在某些一个页面上显示了多个 Blox 的 Web 页面上，您可能想向用户提供将所有 Blox 导出到单个 PDF 文件的选项。

可以使用下列步骤来创建一个按钮，用户可以按下这个按钮来根据页面上的多个 Blox 生成单个 PDF 文件:

1. 在页面顶部添加必需的 page 伪指令和 taglib 伪指令。

   ```
   <%@ taglib uri="bloxtld" prefix="blox" %>
   <%@ taglib uri="bloxuitld" prefix="bloxui" %>
   <%@ page import="com.alphablox.blox.Blox,
   com.alphablox.blox.pdfreport.PDFReport" %>
   ```

   在此示例中，使用了标准的 Blox 标记库和 Blox UI 标记库来定义那些 Blox 的表示 Blox 和嵌套标题。该 page 伪指令提供对一些 Java 类的访问权，对于创建用来在单个 PDF 文件中生成多个 Blox 的按钮来说，这些类是必需的。

2. [可选] 使用嵌套在表示 Blox 中的 `<bloxui:title>` 标记来向各个 Blox 添加标题。

   ```
   <blox:present id="myPresentBlox"> ...
   <bloxui:title title="PresentBlox View"
   style="padding:10;font-weight:bold;"
   alignment="left" />
   ...
   </blox:present>
   ```

   在此示例中，`<bloxui:title>` 标记创建一个标题，该标题将显示在 JSP 页面中此 PresentBlox 的正上方。并且，由于此标记嵌套在 PresentBlox 标记中，因此它还将在 PDF 文件中出现。注意，放在 JSP 页面上的任何标题或其他文本都不会在 PDF 文件中出现。

3. 添加用于将多个 Blox 显示为单个 PDF 文件的按钮。

   ```
   <blox:container id="containerName" visible='true'>
   <%
   String bloxNames="myGridBlox,myChartBlox,myPresentBlox";
   PDFReport.addButton(containerName,"buttonName",
   "Create PDF Report",bloxRequest,bloxNames);
   %>
   </blox:container>
   ```

   在此示例中，`bloxName` 字符串按照 Blox 在 PDF 文件中应该具有的出现顺序来定义将被显示为 PDF 的 Blox 列表。

4. 在将这些内容添加至 JSP 文件之后，用户将能够生成显示了上面定义的所有表示 Blox 的单个 PDF 文件。

指定 PDF 存储位置和文件名

缺省情况下，DB2 Alphablox 在应用程序服务器上生成的 PDF 文件仅暂时地存储在服务器上。在某些情况下，您需要能够指定永久存储位置和唯一文件名，这样就允许用户或您自己创建 HTML 页面来提供指向那些特定文件的链接，或者通过电子邮件发送指向这些已存储的文档的链接。

为了将会话中 Blox 的 PDF 报告存储到指定位置，com.alphablox.blox.pdfreport 包中的 PDFReport 对象有一个 writePDFToFile(AbstractBlox[] *bloxList*, HttpServletRequest *request*, String *path*, Printable *printJob*) 方法。

1. 在 JSP 文件中添加下列 import 语句:

```
<%@ page import="com.alphablox.blox.AbstractBlox"%>
<%@ page import="com.alphablox.blox.pdfreport.Printable"%>
<%@ page import="com.alphablox.blox.pdfreport.PDFReport"%>
<%@ page import="com.alphablox.blox.*" %>
```

2. 照常创建 Blox。在以下代码中，我们将创建一个非常简单的 PresentBlox。

```
<%@ taglib uri="bloxtld" prefix="blox"%>
<blox:data id="dataBlox" dataSourceName="Qcc-Essbase"
useAliases="true" visible="false"
query="!">
<html>
<body>
<blox:present id="myPresentBlox"
width="700" height="500">
<blox:data bloxRef="dataBlox" />
</blox:present>
```

3. 使用 writePDFToFile() 方法将 PDF 报告写入指定的位置。以下示例显示了在会话中调用 JSP 页面和创建 Blox 时保存 PDF 报告所需的基本代码。

```
<%
BloxSession bloxSession = bloxRequest.getBloxSession();
Printable printJob = PDFReport.getPrintable(bloxSession);
AbstractBlox[] bloxList = new AbstractBlox[1];
bloxList[0] =  myPresentBlox;
PDFReport.writePDFToFile(bloxList, request, "c:\\temp\\sales.pdf", printJob);
%>
```

PDF 显示引擎不需要浏览器会话，可以在 Java 代码中触发以上 JSP 以将 PDF 报告直接保存到指定的位置。您可以很容易地修改示例以将多个 Blox 保存到 PDF 报告。您还可以使用 com.alphablox.blox.pdfreport 包中提供的 Printable 接口中的方法来指定布局、页脚、页眉和报告的页数。

或者，您也可以创建两个 JSP 会话属性，一个用于 PDF 文件（使用 PDF_FILE_NAME），另一个用于 PDF 文件存储目录（使用 PDF_DIRECTORY_NAME）:

```
<%
session.setAttribute("PDF_DIRECTORY_NAME", "c:\\temp");
session.setAttribute("PDF_FILE_NAME", "analysis.pdf");
%>
```

这些属性是可选的，根据特定应用程序的不同，可以使用这两个会话属性设置中的任何一个会话属性设置，也可以同时使用这两个会话属性设置。

使用远程 PDF 处理器

为了提高性能、进行内存管理或者让多台 DB2 Alphablox 主机共享 PDF 处理，您可能会决定在远程专用服务器上运行 PDF 引擎。要了解关于配置远程 PDF 服务器的详细信息，请参阅使用远程 PDF 处理器。

以 XML 格式导出

应用程序开发者可以使用以 XML 格式导出的数据来将信息传递给其他应用程序，也可以将该数据与 Java、JavaScript 和 JavaServer Pages 技术配合使用以创建定制数据视图。以下任务说明如何以 XML 格式导出查询结果。

以 XML 格式显示结果集

以 XML 格式显示从应用程序数据源返回的查询结果集包括下列步骤:

1. 定义带有标准 DataBlox 的 HTML 页面。

   注: DB2 Alphablox XML 立方体只能访问显式定义的 DataBlox 的结果集。它不能访问 PresentBlox、GridBlox 或 ChartBlox 的隐式定义的底层 DataBlox 的结果集。

2. 使用 DataBlox 属性或方法来指定它的数据源和查询字符串。
3. 对 DB2 Alphablox 定义应用程序和数据源。
4. 调用应用程序，并确保将 render 属性添加到应用程序的 URL 中:

   ```
   .../AppName.jsp?render=XML
   ```

   render 属性的 XML 值将导致 DB2 Alphablox 执行下列处理:

   - 访问 DB2 Alphablox XML 立方体 DTD（文档类型定义）
   - 以 XML 格式显示 DataBlox 结果集（替换页面上的 DataBlox）
   - 使 XML 文档可用于进一步的处理

注: 当使用独立的 DataBlox 来以 XML 格式显示时，在应用程序页面上不需要 Blox header 标记（`<blox:header/>`），并且该标记可能会导致页面无法正确地显示。此外，您可能想使用具有公共 `render` 属性（将它的值设置为 `xml`）的 DataBlox，而不是使用 `render=xml` URL 属性。

下一节显示上一页的示例结果集的 XML 格式显示结果。

以 XML 格式显示结果集: 样本 DB2 Alphablox XML 文档

以下是以 XML 文档形式显示的示例结果集。在某些情况下，为了提高可读性，添加了换行符。

```
<?xml version="1.0"?>
<!DOCTYPE cube SYSTEM '/AlphabloxServer/xml/dtd/cube.dtd'>
<cube>
  <bloxInfo>
    <bloxID>15</bloxID>
    <bloxName>MyDataBlox</bloxName>
    <appName>MyXMLDoc</appName>
  </bloxInfo>
  <data>
```

```
    <slicer>
<slicerDimension name="Period">Period</slicerDimension>
<slicerMember name="Period" gen="1"
leaf="false">Period</slicerMember>
    </slicer>
    <slicer>
<slicerDimension name="Accounts">Accounts
      </slicerDimension>
<slicerMember name="Accounts" gen="1"
leaf="false">Accounts
      </slicerMember>
    </slicer>
    <slicer>
<slicerDimension name="Scenario">Scenario
      </slicerDimension>
<slicerMember name="Scenario" gen="1"
leaf="false">Scenario
      </slicerMember>
    </slicer>
<axis name="columns" index="0">
      <dimensions>
<dimension name="Market" index="0">Market</dimension>
      </dimensions>
<tuple index="0">
<member name="East" index="0" gen="2" span="1"
spanIndex="0" leaf="false">East
      </member>
      </tuple>
<tuple index="1">
<member name="West" index="0" gen="2" span="1"
spanIndex="0" leaf="false">West
      </member>
      </tuple>
<tuple index="2">
<member name="South" index="0" gen="2" span="1"
spanIndex="0" leaf="false">South
      </member>
      </tuple>
<tuple index="3">
<member name="Market" index="0" gen="1" span="1"
spanIndex="0" leaf="false">Market
      </member>
      </tuple>
    </axis>
<axis name="rows" index="1">
      <dimensions>
<dimension name="Product" index="0">Product
        </dimension>
      </dimensions>
<tuple index="0">
<member name="Audio" index="0" gen="2" span="1"
spanIndex="0" leaf="false">Audio
      </member>
      </tuple>
<tuple index="1">
<member name="Visual" index="0" gen="2" span="1"
spanIndex="0" leaf="false">Visual
      </member>
      </tuple>
<tuple index="2">
<member name="Product" index="0" gen="1" span="1"
spanIndex="0" leaf="false">Product
      </member>
      </tuple>
    </axis>
    <cells>
      <row>
```

```
        <column>
          <cell>13438.0</cell>
        </column>
        <column>
          <cell>22488.0</cell>
        </column>
        <column>
          <cell>0.0</cell>
        </column>
        <column>
          <cell>35926.0</cell>
        </column>
      </row>
      <row>
        <column>
          <cell>33138.0</cell>
        </column>
        <column>
          <cell>40351.0</cell>
        </column>
        <column>
          <cell>24565.0</cell>
        </column>
        <column>
          <cell>98054.0</cell>
        </column>
      </row>
      <row>
        <column>
          <cell>46576.0</cell>
        </column>
        <column>
          <cell>62839.0</cell>
        </column>
        <column>
          <cell>24565.0</cell>
        </column>
        <column>
          <cell>133980.0</cell>
        </column>
      </row>
    </cells>
  </data>
</cube>
```

第 23 章 错误处理

不幸的是，在软件问题中会发生错误。但是，作为开发者，您具有一定程度的管理错误处理方式的能力。本主题提供有关如何使用 Blox 异常、属性和方法来处理可能发生的错误的信息。

异常

虽然软件会发生错误（如果运气好的话，很少出错），但对最终用户来说重要的是在发生错误之后发生的情况。程序仅仅是停止工作还是平稳地恢复？异常是一个事件，它中断程序的正常执行。Java 语言允许您捕获或尝试捕获所发生的异常，以便以受控方式平稳地处理它们。要了解有关异常以及如何处理它们的更多信息，请参阅优秀的 Java 或 JavaServer Pages 参考。

许多 Blox Java 类都会抛出异常，从而允许您使用 Java 语言的错误处理功能。可以在 DB2 Alphablox 的以下目录中提供的 Javadoc 文档中找到可用的 Blox 异常:

```
<db2alphablox_dir>/system/documentation/javadoc/index.html
```

其中 `<db2alphablox_dir>` 是 DB2 Alphablox 的安装目录。

定制错误页面

当 JSP 引擎尝试编译页面但失败时，将生成错误消息和堆栈跟踪并将它们显示给最终用户。这些消息中的大部分消息对最终用户来说没有多大意义，对他们了解所发生的情况也没有什么帮助。作为开发者，您可以选择创建要向用户显示的定制错误消息，而不是显示缺省的标准 JSP 错误页面。page 伪指令的 `errorPage` 和 `isErrorPage` 这两个属性允许您定义 JSP 页面应该在什么位置查找定制错误页面，并允许您将特定 JSP 页面定义为定制错误页面。下面提供了这些属性的简要描述以及使用它们来创建您自己的定制错误页面的步骤。要了解有关错误页面伪指令的用法的更多信息，请参阅初级 JSP 书籍。

errorPage 属性

page 伪指令的 `errorPage` 属性指定要用作错误页面的备用页面，此属性的定义方式如下所示:

```
<% page errorPage="/errorPage.jsp" %>
```

`errorPage` 值指定同一个 Web 应用程序中的一个相对 URL，在该位置，可以找到 JSP 页面。在以上示例中，值在指定的页面前面包括正斜杠（“/”）。正斜杠（尽管不是必需的）通知应用程序服务器: 后面的 URL 是相对于 Web 应用程序的根目录的。通过在这个 page 伪指令中使用正斜杠，可以将定制错误页面放在应用程序目录中的一个位置，然后在应用程序的所有 JSP 文件中使用这个 page 伪指令，即使它们位于子目录中亦如此。

isErrorPage 属性

定制错误页面必须包括将 isErrorPage 属性设置为 true 的 page 伪指令。将布尔 isErrorPage 值设置为 true 的 page 伪指令如下所示:

```
<%@ page isErrorPage="true" %>
```

这个伪指令使页面能够访问 exception 隐式对象中的信息，并允许您控制用户所看到的信息的显示。

创建简单的定制错误页面

下列步骤将指导您完成创建定制错误页面的过程:

1. 创建一个将用作定制错误页面的基本 JSP 文件并将其保存为 errorPage.jsp。
2. 在该页面顶部添加将 isErrorPage 属性设置为 true 的 page 伪指令。例如:

   ```
   <%@ page isErrorPage="true" %>
   <html>
   ...
   </html>
   ```

3. 创建错误页面的主体布局，显示当发生错误时要让最终用户看到的内容。

   例如，您可能想显示错误页面标题并包括发生错误的页面的 URL。并且，您可能会决定显示异常的顶级错误消息而不显示堆栈跟踪，这是因为用户不大可能知道如何解释堆栈跟踪。

   以下是一个简单的示例:

   ```
   <%@ page isErrorPage="true" %>
   <html>
   <head>
      <title>Error Page</title>
   </head>
   <body>
   <h2>Your application has generated an error</h2>
   <h3>Please notify your help desk.</h3>
   <b>Exception:</b><br>
   <%= exception.toString() %>
    </body>
   </html>
   ```

4. 要测试定制错误页面，请添加以下 page 伪指令并将 errorPage 属性值设置为指向定制页面的位置:

   ```
   <%@ page errorPage="errorPage.jsp" %>
   ```

   在此示例中，定制错误页面与测试 JSP 页面在同一个目录中。

5. 通过生成错误来测试错误页面。

   一种生成错误的方法是包括以下 scriptlet，该 scriptlet 将导致发生“除零”运行时错误:

   ```
   <%
      int i = 10;
      i = i / 0;
   %>
   ```

示例: 此定制错误页面示例包括在“Blox 样本程序”示例集的“错误处理”部分中。

注: Application Studio 中的各种示例和基本模板全都包括错误页面，您可以检查并复制该页面以供您自己使用。

Blox 属性和错误处理方法

在定制应用程序时，可以使用下列 Blox 属性和方法来处理错误条件。要了解有关这些属性和方法的详细信息，请参阅 *Developer's Reference* 一书。

noDataMessage

noDataMessage 属性定义当没有数据可用时要在 Blox 中显示的字符串，这是一种将可能的应用程序错误通知用户的方法。这个公共的 Blox 属性适用于 ChartBlox、GridBlox 和 PresentBlox，当这些 Blox 中的其中一个 Blox 被实例化但由于尚未接收到数据或由于发生错误而没有数据可用时，将显示此属性。缺省消息是"没有数据可用"，它主要显示在网格和图表中。

如果缺省的"没有数据可用"消息在网格或图表中显示几秒钟以上的时间，则用户可能会将此消息视为指示没有数据将变为可用的错误消息。在大多数情况下，这是合理的假定，并且不应该修改此消息。

有时，数据检索操作花费的时间比预期的时间要长。这可能是由于查询比较复杂、返回了大型数据集或者连接速度较慢而导致的。在诸如此类的情况下，某些开发者将 noDataMessage 字符串修改为"请等待"或某些其他替代消息。尽管这是此属性的一种合理用法，但是您应该了解，更改显示的消息有时会在实际上没有数据可用时令最终用户感到疑惑。当实际上没有数据可用时，此消息可能仍显示"请等待"。如果您考虑更改此消息，则初始消息时常准确的好处比起当存在真实的数据可用性问题时用户实际上被告知等待的小风险，可能利益大于风险。

另一种方法是在程序中使用相关联的 setNoDataMessage 方法来根据所发生的事件返回另一条消息。尽管比仅仅使用 noDataMessage 属性更为复杂，但您可能会想探索此选项。

onErrorClearResultset

DataBlox 的 onErrorClearResultSet 布尔属性指定当后续数据库操作失败时是否应该从 DataBlox 中清除现有的结果集。要了解关于此属性及其相关方法的更多信息，请参阅 *Developer's Reference* 的 DataBlox 一节。

第 24 章 添加用户帮助

本主题讨论在使用 DB2 Alphablox 创建的应用程序中提供用户帮助时涉及的一些问题。

在理想的世界里，应用程序是直观的，并且用户不需要任何帮助来指示如何使用应用程序。不幸的是，这种情况非常罕见，并且，应用程序越复杂，您就越有可能需要提供用户帮助。

在应用程序的设计和开发过程中，忘记考虑在应用程序中添加帮助是一项常见的疏忽。如果应用程序的用户将是频繁的熟练用户并且将接受培训，则在整个应用程序中添加帮助可能并不关键，但也会有帮助。然而，如果用户将是未经培训的或临时的（不频繁的）用户，则提供用户帮助是确保应用程序成功的重要因素。您的设计小组应该考虑用户帮助并且（如果有必要的话）在应用程序开发周期中安排时间和资源来开发帮助。

下列各节讨论 DB2 Alphablox 应用程序中的用户帮助的可用性和行为以及您的设计决策的影响。

使用现有的 DB2 Alphablox 用户帮助

当应用程序包括 GridBlox、ChartBlox 或 PresentBlox 时，可以向用户提供一个工具栏。根据所使用的 Blox 的不同，该工具栏上的**帮助**按钮将打开 DB2 Alphablox 用户帮助系统并提供有关该特定 Blox 的帮助页面。例如，单击 PresentBlox 工具栏上的**帮助**按钮将打开标题为“使用 PresentBlox”的页面，该页面描述 PresentBlox 并提供了其他指向更多帮助的链接。

此外，如果没有工具栏，则从菜单栏的**帮助**菜单中选择**帮助**将打开用户界面 Blox 的帮助页面，这取决于该 Blox 是 PresentBlox 还是独立的 GridBlox 或 ChartBlox。例如，对于独立的 GridBlox，选择**帮助**菜单选项将打开标题为“使用网格”的帮助页面。

您可能会经常决定不在应用程序中的分析视图上提供工具栏。在这些情况下，应该确保菜单栏可用。缺省情况下，在这些用户界面 Blox 中提供了菜单栏。如果由于任何原因而需要同时关闭菜单栏和工具栏，则您可能想考虑提供定制用户帮助。

创建定制用户帮助

如果您决定不在分析视图上提供工具栏，或者决定要为用户提供目标定制用户帮助，则可以添加适当的帮助链接或按钮。特别是，在下列情况下，您可能想考虑提供定制用户帮助:

- 工具栏或菜单栏不可用
- 使用定制按钮和菜单选项定制了工具栏或菜单栏
- 页面使用定制 HTML 表单元素而不是内置的 Blox 用户界面元素（如页面过滤器和工具栏按钮）来管理 Blox 视图的交互和分析
- 视图包括能够从词汇表或其他帮助信息中受益的成员或其他标注

如果定制了工具栏和菜单栏，则可能还需要定制现有的用户帮助。帮助文件位于文档目录中:

```
<db2alphablox_dir>/system/documentation/help/dhtml/<locale>
```

在修改文件之前，首先应该备份该目录。并且，请记住，在升级服务器时，将除去此目录中的文件并将它们替换为 DB2 Alphablox 用户帮助文件。

如果彻底关闭工具栏和菜单栏，除了使用标准的 HTML 技术来提供定制用户帮助以外，您可能还想考虑使用 DB2 Alphablox 信息链接（在下一节中讨论）来作为一种提供目标和可视帮助信息的方法。

使用信息链接来提供帮助

如第 160 页的『信息链接』所述，在 Blox 上有三种类型的信息链接: 头链接、单元格链接和单元格提醒链接。这些链接（缺省情况下，由蓝色圆圈内的白色“i”表示）可以用于许多用途，包括链接至与行头或列头相关的信息或者链接至有关指定的数据单元格的信息。注意，这些链接也可以用于目标用户帮助，该帮助可能定义了特定成员所代表的内容或者特定数据单元格的评价方式。信息链接的其中一项好处是它们高度可视并且难以被忽略。当然，这也可能是不提供它们或者至少是审慎地使用它们的原因。

第 25 章 使用 DB2 Alphablox FastForward

DB2 Alphablox FastForward 应用程序框架允许应用程序管理员（OLAP 管理员）复制框架、配置报告模板以及快速地部署分析应用程序以供行业用户使用。本节提供 FastForward 框架的概述，您可以添加新的报告模板以定制应用程序框架。

注: 当安装 DB2 Alphablox 以在 WebSphere Portal Server 上运行时，DB2 Alphablox FastForward 不可用。

DB2 Alphablox FastForward 概述

DB2 Alphablox FastForward 是预安装在 DB2 Alphablox 上的样本应用程序框架，它用来在业务机构内快速地开发、部署和共享定制分析视图。FastForward 框架提供了现成的公共应用程序服务，这些服务包括安全性、协作、定制和个性化。应用程序管理员（通常是 OLAP 管理员）可以创建新版本的 FastForward 应用程序，通过选择报告模板和配置报告参数来发布报告，然后部署新的应用程序，而不必查看任何代码。并且，由于它十分灵活并且可以扩展，所以 JSP 开发者可以修改或扩展应用程序框架，并且可以添加新的定制报告模板以供应用程序管理员配置和部署。

报告应用程序和分析应用程序中的常见功能已被构建到 FastForward 应用程序框架中，这些功能包括:

- 以 Microsoft Excel 格式导出
- 生成可打印的视图
- 方便地保存和共享个人数据视图
- 通过电子邮件将视图发送给别人
- 方便地在不同视图之间进行导航

通过提供这些功能，DB2 Alphablox 使您可以方便地加快此类通用报告和分析应用程序的开发速度。您不需要创建导航系统、工具栏、安全性以及用于保存和共享报告的机制 – 我们已经为您预先编码了这些功能。通过将开发任务和配置任务分隔出来，应用程序管理员就可以把注意力放在配置和部署现有的报告模板上，从而使您能够把注意力放在更具挑战性的需求上。

FastForward 用户的角色

DB2 Alphablox FastForward 用户的三种主要角色包括应用程序管理员角色、模板开发者角色和最终用户角色。这三个小组之间的良好配合将有助于确保基于 FastForward 的应用程序获得成功。下面简要描述关于这三种角色的更多信息。

应用程序管理员

应用程序管理员（通常是 OLAP 管理员）应该能够通过定义几项设置来创建新版本的 FastForward 应用程序、根据可用的报告模板来创建报告并接着快速地将解决方案部署给最终用户。如果使用现有的报告模板无法满足最终用户的需求，则应用程序管理员将与模板开发者配合工作以创建新的报告模板。应用程序管理员应该能够利用他们的

OLAP 数据库经验、有关管理 Alphablox FastForward 应用程序的文档（请参阅《*管理员指南*》）以及联机管理帮助（在 FastForward 应用程序的“管理任务”方式下，将提供此帮助）来完成他们的工作。

模板开发者

模板开发者通常是 JSP 开发者，他们主要负责在应用程序管理员无法使用现有模板来配置所请求的报告时创建定制报告模板。通过与应用程序管理员和最终用户协商，模板开发者应该能够通过修改现有报告模板或根据需要创建新的报告模板来创建新的报告模板。

通过使用 Blox 标记库、服务器端 Java API 和 DHTML 客户机 API 以及 Web 编程经验，模板开发者应该能够创建模板来满足几乎每一项能够想象到的需求。除了熟悉构建 DB2 Alphablox 应用程序和视图以外，开发者还应该熟悉 FastForward 用户帮助（在用户方式下，可通过“帮助”按钮获得此帮助）、管理员帮助（在“管理任务”方式下，当作为管理员登录时，可通过“帮助”按钮获得此帮助）和《*管理员指南*》中的『管理 FastForward 应用程序』。

最终用户

最终用户（通常是机构中的业务分析员和其他业务用户）应该能够登录到 FastForward 应用程序中并使用已发布的报告来分析业务问题。根据基于 FastForward 的特定应用程序所提供的交互功能的不同，最终用户可以处理数据、对数据层次结构进行钻取、更改图表类型、添加注释和执行其他操作。在修改视图以回答特定的业务问题之后，用户可以保留他们的当前视图，这可以通过在“专用”选项卡中创建已保存的报告以供将来使用来完成，也可以通过在“组”选项卡中与已定义的应用程序用户组共享那些报告来完成。

对于每个报告，用户通常可以从位于报告上方的应用程序工具栏中选择几个其他选项。除了保存报告以便进行联机分析以外，“以 Excel 格式导出”选项还允许用户将视图导出到 Microsoft Excel 电子表格以便将来进行脱机分析。用户还可以使用“打印预览”选项来打印特定视图的副本。并且，如果他们愿意的话，他们可以打开一条包含当前视图链接的电子邮件消息、添加注释并将该消息发送给其他应用程序用户。

如果最终用户的应用程序未提供所需的报告，他们通常直接向应用程序管理员请求提供新报告。

定制 Alphablox FastForward

虽然 DB2 Alphablox FastForward 提供了一些现成的样本报告模板以使您能够使用它们来立即开始创建应用程序，但是，报告应用程序和分析应用程序很有可能要求您为应用程序管理员创建定制报告模板。为了帮助您开始使用 DB2 Alphablox FastForward 应用程序，本主题提供 FastForward 体系结构和报告模板的概述。接着，您将学习如何创建新的报告模板。

FastForward 应用程序体系结构

FastForward 应用程序由两个主要的组件组成，即框架和在该框架内使用的报告模板。FastForward 框架包括 JSP 页面、JavaBeans 组件以及许多其他用于定义应用程序框架的文件，包括公共应用程序服务（如应用程序配置、导航和安全性）。

在以下目录中提供了样本 DB2 Alphablox FastForward 应用程序:

<alphabloxDirectory>/system/ApplicationStudio/FastForward/

要点: 请不要删除或修改此目录 – 此目录包含样本应用程序使用的文件。当升级 DB2 Alphablox 时，此目录将被覆盖。

当（使用 AlphabloxAdministrator 或其他已定义的管理员角色）作为 DB2 Alphablox 管理员登录时，您可以通过单击“管理任务”按钮来创建新的 FastForward 应用程序，然后通过单击对话框中的“创建”来创建样本应用程序的新版本。当您单击“创建”选项时，将有一个对话框窗口提示您定义新应用程序的上下文名称（目录名）、显示名称（此名称将显示在“应用程序”页面上）、简要的应用程序描述以及被允许编辑该应用程序的管理员角色。

注: 在创建新的 FastForward 应用程序时，需要指定一个管理员角色。只有隶属于该指定角色（缺省角色为 AlphabloxAdministrator）的用户才可以管理该特定应用程序。

在单击“确定”之后，将会自动创建一个新的 J2EE 应用程序并将 FastForward 应用程序框架和样本报告模板的副本复制到该应用程序的目录中，该目录通常位于以下位置:

<alphabloxDirectory>/webapps/<applicationDirectory>

注: 要了解关于配置 FastForward 应用程序的详细信息，请参阅《*管理员指南*》中的『管理 Alphablox FastForward 应用程序』一节。

在创建 FastForward 应用程序的副本时，将把下列目录和文件添加到所创建的 Web 应用程序中。以下是 FastForward 框架中包括的目录结构和文件的概述。

目录　描述

admin　包含在“管理”方式下使用 FastForward 应用程序时使用的大多数文件，包括管理帮助目录。

admin/help
专门用来帮助您管理应用程序的帮助文件。

admin/images
包含在“管理”方式下使用的图像。

help　包含用户帮助文件。应用程序开发者可以根据需要对此目录进行定制。

images　包含在“用户”方式下使用的图像文件。

模板　包含可供应用程序使用的报告模板集合。下面的『报告模板』一节描述了此目录的子组件。

WEB-INF
标准的 J2EE 应用程序目录，此目录包含运行应用程序时所需的文件。

WEB-INF/classes
包含应用程序中使用的 JavaBeans 组件的已编译 Java 类文件。

WEB-INF/src
JavaBeans 组件的源文件，这允许您根据需要进行定制或扩展。

WEB-INF/tlds
Blox 标记库描述符文件的副本，包括 blox.tld、bloxform.tld、bloxlogic.tld、bloxreport.tld 和 bloxui.tld。

WEB-INF/ui

包含用来定义应用程序外观的 XML 文件，这些文件包括 buttons.xml、toolbar.xml 和 toolbarhelp.xml。您可以根据需要来编辑这些文件以添加或除去按钮、更改按钮图像以及编辑工具提示描述。应用程序开发者不应该更改其余的 XML 文件。

对于大多数应用程序来说，您将主要对 templates 目录感兴趣，报告模板就存储在此目录中。下一节更详细地描述报告模板和 templates 目录。

报告模板

报告模板是 FastForward 应用程序的心脏，它为应用程序管理员提供编辑页面，这些页面用来快速地为最终用户配置报告。报告模板至少由 4 个文件组成：编辑页面、模板参数文件、报告页面和帮助页面。通过从用于定义报告选项和数据选项的选择列表、单选按钮和复选框中进行选择，应用程序管理员使用**编辑页面**来创建新报告。编辑页面中提供的选项或参数是在**模板参数文件** 中定义的。最终用户看到的内容是由**报告页面**的布局确定的。还包括了与特定报告相关的**帮助页面**。（可选）可以将打印页面和 Excel 页面包括在报告模板中。模板开发者可以创建新的报告模板，这些报告模板将添加到可用报告模板的集合中，从而减少所需进行的常用报告类型的定制报告开发工作。因此，最终用户可以访问和修改报告并保存那些报告的副本以供将来重用和共享。

对于每个 FastForward 应用程序，可供该应用程序使用的报告模板集合位于以下目录中：

<applicationDirectory>/templates/

在 templates 目录中有一组报告模板子目录，在这些子目录中，每个子目录都构成一个独立的报告模板。当创建 FastForward 应用程序的新版本时，将复制样本报告模板并将它们包括在此目录中。还可以通过简单地将模板放到包含可用报告模板集的目录中来快速地添加新报告模板。除了 DB2 Alphablox 附带包括的样本报告模板以外，还可以从 Alphablox 下载其他模板、在机构中的模板开发者之间共享模板以及由第三方开发者创建模板。当将新的报告模板添加至 templates 目录时，它将立即可供应用程序管理员使用（显示在他们的可用报告模板列表中）。并且，放到 templates 目录中的报告模板的压缩副本将自动地被解压缩并显示在菜单中。您可以根据需要继续添加新的报告模板，从而快速地使它们可供应用程序管理员使用，而不必停止和启动服务器。

每个报告模板目录都至少包含 3 个必需的文件，概述如下：

文件名 描述

edit.jsp

编辑页面，管理员通过查看和配置此页面来创建新报告或编辑现有报告。

template.xml

模板参数文件，它包含可以由应用程序管理员设置的所有参数（属性）的列表。

report.jsp

报告页面，当与管理员定义的值相组合时，此页面将生成可以供用户查看和处理的报告。

下列模板文件是可选的：

文件名 描述

help.jsp

帮助页面向用户提供有关这个特定报告的有用信息。样本模板包括一个帮助页面原型。

excel.jsp

Excel 页面，它提供发送至 Microsoft Excel 的定制 HTML 页面。

print.jsp

打印页面提供可供打印的定制 HTML 页面。

报告模板的复杂程度是由用户期望看到的内容（report.jsp 文件中显示的内容）、由参数的数目和类型（那些参数是在 template.xml 文件中指定）以及用于配置那些参数的用户界面（使用 edit.jsp 生成的界面）确定的。借助简单的报告模板，就有可能限制参数设置的数目，并且许多设置已被预定义。模板越灵活，提供的配置选项就越多，但也会要求提供大量的参数。

以下是 DB2 Alphablox 附带提供的样本报告模板的描述，第 242 页的『创建定制报告模板』一节将在描述新报告模板的创建步骤时阐述有关这三个文件的更多详细信息。

样本报告模板

DB2 Alphablox 附带提供的样本报告模板包括大部分企业通常会遇到的各种常用报告类型。虽然这些样本中的其中一些样本本身就很有用，但这些模板还可以用来帮助您了解如何开发自己的定制报告模板。

DB2 Alphablox 附带包括的样本模板面向不同类型的报告需求，概述如下:

样本模板	目录名	描述
交互式显示 Blox	InteractiveBlox	样本报告: Interactive Analysis
样本分配	SampleAllocation	样本报告: Sales by Store (Sales Analysis)
样本报告	SampleReport	样本报告: Sample Report (Sales Analysis)
样本趋势	SampleTrending	样本报告: Sales Trend by Region (Sales Analysis)
销售方差	SampleVariance	样本报告: Sales Variance (Variance Analysis)
交互式差异	VarianceQCC	样本报告: Ad-Hoc Variance Analysis (Variance Analysis)
样本报告 Blox	SampleReportBlox	样本报告 Blox 模板

虽然样本模板无法满足您的所有特定用户需求，但是，它们对于您迅速了解报告模板的强大功能和灵活性以及作为优秀的示例以使您了解如何针对特定问题编码解决方案来说非常有用。虽然您有可能在只进行少量修改的情况下部署样本 Alphablox FastForward 应用程序的副本，但是，当您开始添加针对独特用户需求进行定制的定制报告模板时，就可以实现应用程序框架和报告模板的强大功能。下一节说明如何创建一个简单的报告模板。

创建定制报告模板

为了帮助您开始开发自己的报告模板，本节将指导您完成在开始创建定制报告模板前需要了解的最重要步骤。如上所述，每个报告模板都包含 3 个重要的文件，即报告页面（report.jsp）、模板参数文件（template.xml）和编辑页面（edit.jsp）。下列步骤描述如何为一个简单的分配报告模板创建这些必需文件中的每个文件。

创建或修改报告页面（report.jsp）

报告页面（report.jsp）生成视图，在应用程序管理员配置报告后，FastForward 应用程序用户就能够查看这些视图。此页面通常包含下列信息：报告标题、数据视图和用户控件。报告页面既可以是静态的查看页面也可以是指导型的分析页面，并且，高级的报告页面可以包含许多用户控件。

要创建报告页面，请首先创建一个具有代表性的 JSP 页面，它包含最终用户将要使用的功能。

在本示例中，报告在装入后将显示标题和基本的饼图视图，该视图显示了指定的产品分组的各个子类别的百分比。

要创建可以用来生成此视图的报告模板，您需要创建在生成报告时需要读取的参数。这些参数（以后，您将在模板参数文件 template.xml 中定义它们）包括应用程序管理员在使用编辑页面（edit.jsp）配置应用程序时能够设置的属性。在我们完成创建报告页面和模板参数文件的操作后，您将学习如何创建编辑页面。

使用报告原型文件，我们将下面这一行添加至文件顶部:

```
<%@ include file="../../reportdata.jsp" %>
```

这个 JSP include 伪指令在被编译时将导致把 reportdata.jsp（此文件位于模板的根目录中）的内容添加到报告页面中。reportdata.jsp 文件导入必需的类文件，为 bloxtld 和 bloxformtld 添加 taglib 伪指令，并使下列对象可供 report.jsp 使用:

对象　用途

report 提供对报告参数（即属性）的访问

user 提供对用户参数的访问

appContext
提供对应用程序参数的访问

template
提供对模板及其参数的访问

savedState
提供对已保存的专用报告和组报告的访问

这些对象实现了下列方法:

方法
String getParameter(String name) - 返回指定的参数的参数值，或者返回 NULL。如果指定的参数尚未定义，则返回 NULL。

方法
String getParameter(String name, String default) - 返回指定的参数的参数值，或者返回 defaultValue 自变量。如果指定的参数尚未定义，则返回给定的 defaultValue。
String[] getParameterValues(String name) - 返回为指定的参数定义的参数值。如果未定义任何参数值，则返回空数组。
String[] getParameterNames() - 返回一个数组，该数组包含为此对象定义的所有指定参数。

为了将报告原型参数化，您现在需要在报告页面顶部定义一个 JSP scriptlet 以指定要使用的参数。您需要指定饼图切片父成员以及要使用的量度，然后生成一个查询字符串来读取这两个参数，如下所示（用于 DB2 OLAP Server 或 Essbase 数据源）：

```
<%
  String pieSliceParent =
report.getParameter("pieSliceParent","Specialties");
String measure = user.getParameter("preferredMeasure", "Sales");
String query = "&lt;SYM &lt;COLUMN (\"Measures\") \""+measure+"\"
&lt;ROW (\"All Products\") &lt;CHILD \""+pieSliceParent+"\" !";
%>
```

如果您正在使用 Microsoft Analysis Services，则第三行（用于指定查询字符串）将如下所示:

```
String query = "SELECT DISTINCT( {[Measures].["+measure+"]} ) ON COLUMNS,
{"+pieSliceParent+".children} ON ROWS FROM [qcc]";
```

注意，本示例中使用的 getParameter 方法允许在参数名后面使用可选的第二个自变量来设置这两个属性的缺省值。

接着，将页面中每个要使用参数值的位置的内容替换为 JSP 表达式。在 DataBlox 标记中，需要使用 <%= query %> 来读取用户的查询:

```
<blox:data id="dataBlox"
dataSourceName="QCC-Essbase"
useAliases="true"
query="<%= query %>"/>
```

还需要在报告标题中添加 <%= measure %> 和 <%= pieSliceParent %> 这两个 JSP 表达式来替换 measure 和 pieSliceParent 值，如下所示:

```
<h3>Comparing <%=measure%> for subcategories of <%= pieSliceParent %> </h3>
```

这将得到适合于所显示的饼图的标题。

在页面顶部添加 JSP include 伪指令以及添加将读取保存的值的 JSP 表达式之后，完整的 report.jsp 文件应该类似于:

```
<%@ include file="../../reportdata.jsp" %>
<%@ taglib uri="bloxlogictld" prefix="bloxlogic"%>
<%@ taglib uri="bloxuitld" prefix="bloxui"%>
<%
  String pieSliceParent =
report.getParameter("pieSliceParent","Specialties");
String measure = user.getParameter("preferredMeasure", "Sales");
String query = "<SYM <COLUMN (\"Measures\") \""+measure+"\"
<ROW (\"All Products\") <CHILD \""+pieSliceParent+"\" !";
%>
   <blox:header/>
<body>
```

```
<blox:data id="dataBlox"
dataSourceName="QCC-Essbase"
useAliases="true"
query="<%=query%>"/>
<h3>Comparing <%=measure%> for product code subcategories:
   <%=pieSliceParent%></h3>
<blox:chart id="chartBlox"
bloxName="chart"
height="100%"
width="100%"
chartType="Pie"
totalsFilter="0" >
<blox:data bloxRef="dataBlox" />
</blox:chart>
</body>
```

创建或修改模板参数文件（template.xml）

接着，您需要定义将由 FastForward 应用程序管理员配置的参数（即属性）。要完成此任务，您需要创建新的 template.xml 文件或修改示例文件（示例文件仅包含报告模板所必需的参数）。这应该是一个相对较短的过程。

template.xml 文件顶部包含这个 XML 文件的必需 DTD 说明:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE template PUBLIC "-//Sun Microsystems, Inc.//DTD
Web Application 2.2//EN" "../template.dtd">
```

在此说明之后，包括了 template 元素，此元素包含这个特定模板的参数。下表列示了 template 元素的可用嵌套元素:

元素　描述

display-name
包含一个名称，此名称将显示在编辑页面中的报告模板选项选择列表中。可以添加一个可选的 **lang** 属性，该属性是使用标准语言代码和国家或地区子代码（例如 pt_BR 或 fr）定义的。

description
[可选] 将显示在编辑页面中的简要报告描述。可以添加一个可选的 **lang** 属性，该属性是使用标准语言代码和国家或地区子代码（例如 pt_BR 或 fr）定义的。

report-page
指定将用来生成报告的报告页面。

edit-page
指定一个编辑页面的文件名，该编辑页面创建视图，应用程序管理员使用该视图来定义报告

report-params
[可选] 定义可以由管理员设置的报告参数集合。每个 param 元素都嵌套在此元素中。

param　[可选] 指定此元素的内容定义的是一个参数。

param-name
[可选] 此标记嵌套在 param 元素中，它指定编码模板时使用的参数的名称。

param-label
[可选] 此标记嵌套在 param 元素中，它指定参数的显示名称，应用程序管理员

将在报告模板的编辑页面中看到此名称。可以添加一个可选的 **lang** 属性，该属性是使用标准语言代码和国家或地区子代码（例如 en-GB 或 fr）定义的。

default-value
[可选] 参数的缺省值。当报告未提供值时，将使用此值。

print-page
[可选] 指定报告中使用的打印页面。

excel-page
[可选] 指定报告中使用的“以 Excel 格式导出”页面。

help-page
[可选] 指定报告中使用的帮助页面。

注: 必须按照模板参数在编辑页面中应该具有的出现顺序来定义它们。并且，编辑页面文件名和报告页面文件名可以是任何合理的名称。以下示例使用的文件名遵循 FastForward 附带包括的样本报告模板的做法，并且，当您继续完成下面的操作时，请进行合理的命名。

在本示例中，您需要修改 display-name、description 和 report-params 元素。report-page 和 edit-page 元素定义那些页面的实际文件名。

阐述本示例之后，下面给出整个 template.xml 文件的内容（包含已定义的参数）：

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE template PUBLIC "-//Sun Microsystems, Inc.//DTD
Web Application 2.2//EN" "../template.dtd">

<template>
   <display-name>Allocation (Essbase Version)</display-name>
   <description>First Allocation Template (Essbase Version)
   </description>
   <report-page>report.jsp</report-page>
   <edit-page>edit.jsp</edit-page>
   <report-params>
      <param>
         <param-name>pieSliceParent</param-name>
         <param-label>Parent Member of Pie Slices:</param-label>
      </param>
   </report-params>
</template>
```

注意，report-params 元素定义此模板将要使用的报告参数集合。在上面的内容中，嵌套的 param 元素的 pieSliceParent 定义了一个参数，该参数指定要显示的饼图切片的父成员。在创建报告页面和模板参数文件之后，最后一项任务是创建编辑页面，该页面显示供应用程序管理员配置的可选择选项。

创建或修改编辑页面（edit.jsp）

应用程序管理员使用编辑页面来定义报告参数（即属性），这些参数通常显示在选择列表、单选按钮和复选框中。样本 FastForward 应用程序中包括的样本报告模板中的编辑页面比本示例所需的编辑页面要复杂，但是，包括的基本步骤是相同的。

让我们开始创建编辑页面。在 edit.jsp 文件顶部，添加一个 JSP page 伪指令，该伪指令指定需要导入的必需类:

```
<%@ page import="com.alphablox.blox.form.FormEventListener,
                 com.alphablox.blox.DataBlox,
                 com.alphablox.blox.form.TimePeriodSelectFormBlox,
                 com.alphablox.blox.logic.timeschema.TimeSchemaBlox,
                 com.alphablox.blox.form.FormEvent,
     com.alphablox.blox.ServerBloxException,
     fastforward.*,
                 com.alphablox.blox.form.MemberSelectFormBlox,
com.alphablox.blox.data.mdb.Member" %>
```

接着，添加 JSP taglib 伪指令以访问页面上使用的 Blox 标记库:

```
<%@ taglib uri="bloxtld" prefix="blox" %>
<%@ taglib uri="bloxformtld" prefix="bloxform" %>
<%@ taglib uri="bloxlogictld" prefix="bloxlogic" %>
```

并且，所有访问 Blox 标记的 JSP 页面都需要 <blox:header> 标记:

```
<blox:header />
```

接着，指定一个 DataBlox，编辑页面将使用该 DataBlox 来生成选择列表选项。以下 <blox:data> 标记定义了一个 DataBlox，该 DataBlox 已被预先配置为使用 QCC-Essbase 数据源（应该已经在 DB2 Alphablox 应用程序定义页面中指定了此数据源）:

```
<blox:data id="dataBlox"
useAliases="true"
dataSourceName="QCC-Essbase" />
```

并且，指定可能需要的任何标记属性。在本示例中，useAliases 标记属性值 true 告诉服务器您想要查看显示成员名，而不想查看由已定义的 DB2 OLAP Server 或 Hyperion Essbase 数据源提供的唯一成员名。如果您使用的是 Microsoft Analysis Services，则本示例的数据源名称将是 QCC-MSAS，并且不应该添加 useAliases 标记属性，这是因为此属性仅适用于 DB2 OLAP Server 和 Essbase 数据源。

接着，指定 MemberSelectFormBlox 并将 id 设置为 pieSliceParent，以便使用从数据源返回的饼图切片父代选项来生成选择列表:

```
<formblox:memberSelect id="pieSliceParent"
visible="false"
dataBloxRef="dataBlox"
dimensionName="All Products"/>
```

注: 以下是几个 FormBlox 用法要点:

- edit.jsp 页面中定义的 FormBlox 的 id 属性名必须与 template.xml 文件中使用的参数名完全匹配。注意，在这里的示例中，MemberSelectFormBlox 的 id 是 pieSliceParent，它与 template.xml 文件中定义的 pieSliceParent 参数匹配。
- 请记住将 visible 标记属性设置为 false，以防 Blox 在任何处理逻辑完成前显示。在本示例中，在处理逻辑完成后，以下 TemplateHelper 类的 renderControls 方法将在页面上显示该 Blox。如果您忘记添加这个 visible 属性（visible="false"），或者意外地将其设置为 true，则您在页面上将出乎意料地看到重复的 Blox。

现在，您已经完成了定义编辑页面将要显示的选择列表的操作，接下来，可以构建页面了。

以下 JSP scriptlet 生成要显示的页面，从而应用先前保存的参数值并显示页面控件（那些控件先前被设置为不可视）。它还建立一个验证器，该验证器将用来确保管理员输入所需的信息。验证步骤是可选的，但它能够使应用程序更加健壮，从而帮助确保用户输入所需的值:

```
<%
    TemplateHelper.applySavedParameters(pageContext);
    TemplateHelper.renderControls(pageContext);
Template template=(Template)request.getAttribute("template");
    template.setValidator(new Validator());
%>
```

接下来，定义了 Validator，它实现 ReportValidator。如果要实现 ReportValidator，类就需要定义一个函数 validate(ReportData data)，该函数执行关键的工作。验证器访问已定义参数的方式与报告访问参数的方式相同 – 通过对需要检查的任何参数的数据对象调用 getParameterValue()。

在本示例中，执行的检查操作验证管理员是否未选择作为叶成员的 pieSliceParent（即，它没有子代）。并且，添加了适当的错误消息，当用户尝试使用不允许的值时，将显示该消息。

```
<%!
    public class Validator implements ReportValidator {
      public void validate(ReportData data) throws ServerBloxException {
String pieSliceParent=data.getParameterValue("pieSliceParent");
          if (pieSliceParent == null){
data.addError("Please select a member from the products.");
return;
}
          // There is a FormBlox associated with pieSliceParent
// Use this FormBlox to get the selected member object and validate it
          MemberSelectFormBlox select =
(MemberSelectFormBlox)data.getFormBlox("pieSliceParent");
          Member members[] = select.getSelectedMembers();
// there is only one selected member -- it cannot be a leaf
          if (members[0].isLeaf() == true) {
data.addError("The selected member must have some children");
}
}
}
%>
```

编辑页面现已完成。以下是整个 edit.jsp 文件的完整副本，供您参考:

```
<%@ page import="com.alphablox.blox.form.FormEventListener,
                 com.alphablox.blox.DataBlox,
                 com.alphablox.blox.form.TimePeriodSelectFormBlox,
                 com.alphablox.blox.logic.timeschema.TimeSchemaBlox,
                 com.alphablox.blox.form.FormEvent,
                 com.alphablox.blox.ServerBloxException, fastforward.*,
                 com.alphablox.blox.form.MemberSelectFormBlox,
com.alphablox.blox.data.mdb.Member"%>
<%@ taglib uri="bloxtld" prefix="blox"%>
<%@ taglib uri="bloxformtld" prefix="formblox"%>
<%@ taglib uri="bloxlogictld" prefix="bloxlogic"%>
   <blox:header />
<blox:data id="dataBlox"
useAliases="true"
dataSourceName="QCC-Essbase" />
<formblox:memberSelect id="pieSliceParent"
visible="false"
dataBloxRef="dataBlox"
dimensionName="All Products" />
<%
    TemplateHelper.applySavedParameters(pageContext);
    TemplateHelper.renderControls(pageContext);
Template template=(Template)request.getAttribute("template");
    template.setValidator(new Validator());
%>
<%!
```

```
    public class Validator implements ReportValidator {
      public void validate(ReportData data) throws ServerBloxException {
String pieSliceParent=data.getParameterValue("pieSliceParent");
          if (pieSliceParent == null){
data.addError("Please select a member from the products.");
return;
}
          // There is a FormBlox associated with pieSliceParent
          // You can use this FormBlox to get the selected member
          // object and validate it,
          MemberSelectFormBlox select =
(MemberSelectFormBlox)data.getFormBlox("pieSliceParent");
          Member members[] = select.getSelectedMembers();
          // there is only one selected member -- it should not be a leaf
          if (members[0].isLeaf() == true) {
data.addError("The selected member must have some children");
}
}
}
%>
```

在创建有效模板的过程中，编辑页面是您需要创建的最后一个页面。在完成这个简单的模板之后，您可以使用样本报告模板中的编辑页面来帮助您开始构建更复杂的模板。但是，在创建功能更强大的报告模板之前，您应该对刚刚创建的报告模板进行测试。

创建可选模板页面

如上所述，模板可以包括帮助页面、打印页面和 Excel 页面。您可以对这些页面进行定制以满足特定的应用程序需求。

对于创建的每个报告模板，建议您包括一个帮助页面来阐述提供给用户的报告的用法。样本报告模板包括一个名为“报告帮助”的链接，该链接指向示例报告帮助页面。帮助页面的文件名是在 `template.xml` 文件的 `help-page` 参数中指定的。

在 FastForward 应用程序中，应用程序工具栏上的“打印预览”按钮导致在将 `render` URL 属性设置为 `printer` 的情况下显示当前报告。要了解 Printer 格式的详细信息，请参阅第 138 页的『打印机格式（render=printer）』。第 140 页的『通过基于 HTML 的打印功能进行打印』 提供了有关创建定制打印页面的其他详细信息。

在应用程序工具栏上，还有一个适用于 FastForward 报告的“以 Excel 格式导出”选项。此选项生成当前报告中的结果并在将 `render` URL 属性设置为 `xls` 的情况下显示该结果。要了解 Excel 格式的详细信息，请参阅第 139 页的『以 Excel 格式导出（render=xls）』。

本地化 FastForward 应用程序

DB2 Alphablox 在服务器级支持全球本地化。因此，将根据服务器的语言环境而不是根据输入用户请求来将 Blox 本地化。JSP 文件中的或 Java 类中的所有字符串都是从消息束文件 `FastForwardBundle_<lang>.properties` 中抽取的，其中 `<lang>` 是语言代码，包括任何国家或地区子代码（例如 `en-GB` 或 `fr`）。FastForward 应用程序附带的所有 JSP 示例文件都使用标准的国际化（i18n）标记库，该标记库由 Apache 项目提供。正如前面的『模板参数文件』一节所述，可以对 `display-name`、`description` 和 `param-label` 元素应用可选的 `lang` 属性。可以使用这些元素的多个实例以支持多种语言。

测试报告模板

假定您已经正确地创建了报告页面（report.jsp）、参数定义页面（template.xml）和编辑页面（edit.jsp），现在，您已经有了一个简单的报告模板，可以在 FastForward 应用程序的管理页面中使用该报告模板。要测试该模板，请将整个模板放在 FastForward 测试应用程序的 templates 目录中。接着，在 Microsoft Internet Explorer 中打开应用程序，然后单击“管理任务”按钮。继续并创建一个新报告，从可用报告模板的列表中选择新模板，并查看它是如何工作的。在编辑页面上，可以通过单击“预览”按钮来预览报告，也可以单击“返回至应用程序”以作为最终用户来测试该报告。

保存报告模板

要使用报告模板，应该将创建的所有模板文件（包括 report.jsp、template.xml 和 edit.jsp）都存储在应用程序的 template 目录的一个子目录中。编辑页面中显示的模板名是从 template.xml 文件中定义的 display-name 元素读取的。

共享报告模板

现在，您已经有了一个有效模板，请记住，您可能想与其他模板开发者共享工作成果。交换模板有助于满足其他应用程序用户的需求，您的模板也可能成为供别人学习的优秀范例。记住，如果将模板目录压缩为一个文件并将其交给别人，他们只需将压缩文件放到 FastForward 应用程序的模板目录中，应用程序管理员就立即可以开始使用该模板了。在新的浏览器会话打开 FastForward 应用程序之后，新的报告模板就会出现。

使用 savedState 对象来保存状态

当管理员创建报告并将它们提供给用户使用时，那些报告就会变为“已发布”的报告，“已发布”的报告仅仅是定义了参数的 JSP 页面。当用户保存报告以供自己个人访问或供小组访问时，FastForward 就会尝试保存该报告，以便以后能够复原该报告并恢复用户当前所作的更改。以下是它的工作方式概述:

1. 当用户单击“保存报告”按钮时，FastForward 将存储下列信息:
   a. 保存的新报告所基于的模板和报告模板的名称
   b. 与该报告相关联的参数
   c. 页面上每个 FormBlox 的 FormValues 属性
   d. 页面上每个可建立书签的 Blox（GridBlox、ChartBlox、PresentBlox 和 DataBlox）的书签
2. 以后，当用户重新装入保存的报告时，将基本上按照同一顺序重新构建该报告:
   a. 根据保存的参数来装入并编译页面
   b. 使用保存的 FormValues 属性来调用 setFormValues()
   c. 对每个保存了书签的 Blox 调用 restoreBookmark() 方法

通常，这种工作方式在大多数情况下都是可接受的。但是，有时您可能不想使用这个标准复原过程。例如，您可能想更改报告中显示的数据以使该数据基于当前日历天／季度／月份，或者，对于供小组访问的报告，您可能想装入个性化的报告，即根据装入报告的用户的不同来设置某些值。

为了便于完成此工作，FastForward 提供了 savedState 对象，在复原专用的或供小组访问的报告之后，就可以从 JSP 页面中使用此对象。如果报告已被“发布”，则 savedState 对象不可用，对其进行的所有引用都将返回 null。

savedState 对象提供了下列功能:

- 能够关闭缺省复原行为
- 能够获取页面上的任何 FormBlox 或其他标准 Blox
- 能够获取与任何可建立书签的 Blox 相关联的书签
- 能够按照所需的任何顺序复原各种 Blox 的状态。

要了解有关 savedState 对象的这些功能以及其他功能的详细信息，请参阅 FastForward Javadoc 文档，可以从 DB2 Alphablox 管理页面中的“帮助”菜单获得那些文档。

为了在创建新报告时使用此对象，您需要将定制报告复原逻辑添加至 JSP 文件末尾，从而在受影响的 Bolx 被实例化之后但在页面上显示之前应用该逻辑。

在以下示例中，对复原的报告修改了 GridBlox 以便在网格中禁用行色带:

```
<blox:present id="myBlox" visible="false"
width="100%" height="100%"
dataLayoutAvailable="<%=dataLayoutVisible%>"
menubarVisible="<%= menubarVisible %>">
<blox:grid visible="<%=gridVisible%>"
bandingEnabled="<%=gridBanding%>"/>
<blox:chart visible="<%=chartVisible%>"
chartType="<%=chartType%>"
totalsFilter="0"/>
<blox:data bloxRef="dataBlox" />
<blox:toolbar visible="<%=toolbarVisible%>"
removeButton="Save,Load" />
combineToolbars(myBlox.getBloxModel()); %>
</blox:present>
<%
if (savedState != null) { Bookmark bookmark =
savedState.getBloxBookmark("myBlox");
BookmarkProperties gridProps =
bookmark.getBookmarkPropertiesByType(Bookmark.GRID_BLOX_TYPE);
gridProps.setProperty("bandingEnabled", "false"); }
%>
<blox:display bloxRef="myBlox"/>
```

后续步骤

一旦您有了一些简单报告模板，您就可以开始开发更具挑战性的报告模板了。当您构造自己的模板时，样本 Alphablox FastForward 应用程序中包括的样本报告模板能够为您提供丰富的创意和代码源泉。

并且，您可以使用所有的可用开发者资源，包括 *Developer's Reference*、本指南、*FastForward API Javadoc* 文档以及 *Blox API Javadoc*文档。这些文档也可以从 DB2 Alphablox 管理页面上的“帮助”菜单中获得。

第 26 章 DHTML 客户机 DOM API

由于实现将有可能发生更改，所以，除非使用下列各节描述的已发布的 DHTML 客户机 DOM API，否则开发者不应该编写处理或遍历由 DHTML DB2 Alphablox 客户机生成的 DOM 的客户机端代码。

GridBlox 客户机 API

以下主题描述在 JSP 页面上显示时 GridBlox 组件上可用的客户机 API。

Blox 定义

```
<blox:grid id="myBlox"
width="80%"
height="30%"
bandingEnabled="true"
visible="false">
<blox:data bloxRef="dataBlox" />
</blox:grid>
```

DHTML 客户机将返回文档名称空间中的 JavaScript 对象。

要获取对 Blox 的 JavaScript 对象的引用，请使用:

```
var gridBlox = document.myBlox;
```

或者

```
var gridBlox = myBlox;
```

网格

GridBlox 使用基于零的网格数组来提供对它所包含的网格的访问。

要获取对先前定义的 GridBlox 所包含的网格的引用，请使用:

```
var myGrid = myBlox.grids[n];
```

其中 n 是网格的基于零的编号。

这将返回引用该网格的元素。此外，该网格包含总行数和总列数这两个属性。

要返回网格中的可滚动行数或列数:

```
var numRows=myGrid.getRowCount(); var numCols=myGrid.getColumnCount();
```

这些值表示可滚动元素的总数，并且对头行或列与数据行或列不加以区别（如果它们可滚动的话）。如果可滚动元素的数目未填满可用的区域，则不需要进行滚动。如果网格区域的大小发生更改，则将根据需要自动添加或除去滚动条。

要将网格滚动至指示的行和列，请使用 scrollTo 方法:

```
myGrid.scrollTo(row,column)
```

scrollTo 方法将网格滚动到指示的行和列。

要确定网格是否支持滚动:

```
var enabled = myGrid.isScrollingEnabled()
```

如果网格支持滚动，则 isScrollingEnabled 方法将返回 true。

选择

网格使最终用户能够选择一个或多个单元格。然后，他们可以对选择的单元格执行操作。DHTML 客户机允许使用下列方法来对那些选择的单元格进行程序访问。

选择对象

要访问代表当前在该网格中选择的单元格的选择对象，请使用:

```
var myGrid=myBlox.grids[0]; var select = myGrid.selection;
```

检索可视的所选单元格标识值

选择对象提供了一种检索基于零的字符串数组的方法，在这个数组中，每个字符串都是在相关联的网格中当前选择的单元格的标识。

```
var selectedCellIds = select.getCellIds();
```

此数组将包含已选择的和当前可视的单元格的标识。如果需要整一组已选择的单元格，则应该访问模型。

要确定单元格是否已被选中，请使用:

```
var selected = selection.isSelected( cellID)
```

如果单元格已被选中，则返回 true。选择的单元格在客户机上可能可视也可能不可视，但它们的选中状态将被客户机保留。

要控制单元格的选择，请使用:

```
selection.selectCell(cellID,selected)
```

cellID 必须是有效的网格单元格标识。将 selected 设置为 true 以选择单元格，设置为 false 以取消选择单元格。

要清除所有已选择的单元格:

```
selection.clearSelection()
```

所有单元格选择都将被清除。

声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing,
IBM Corporation,
North Castle Drive, Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation, Licensing,
2-31 Roppongi 3-chome, Minato-ku,
Tokyo 106-0032, Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation “按现状” 提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本资料中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Corporation,
J46A/G4, 555 Bailey Avenue,
San Jose, CA 95141-1003
U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本文档中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名称都是虚构的，与实际商业企业所用的名称和地址的任何雷同纯属巧合。

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。用户如果是为了按照 IBM 应用程序编程接口开发、使用、经销或分发应用程序，则可以任何形式复制、修改和分发这些样本程序，而无须向 IBM 付费。

商标

下列各项是 International Business Machines Corporation 在美国和／或其他国家或地区的商标或注册商标:

DB2	DB2 OLAP Server	DB2 Universal Database™
IBM	WebSphere	

Alphablox 和 Blox 是 Alphablox Corporation 在美国和／或其他国家或地区的商标或注册商标。

Java 和所有基于 Java 的商标是 Sun Microsystems, Inc. 在美国和／或其他国家或地区的商标。

Linux® 是 Linus Torvalds 在美国和／或其他国家或地区的商标。

其他公司、产品或服务名称可能是其他公司的商标或服务标记。

索引

D

E

F

G

H

I

J

程序号: 5724-L14

中国印刷

S151-0144-01

Spine information:

IBM DB2 Alphablox

DB2 Alphablox 开发者指南

版本 8.4