为包含数组的 IMS 事务创建 J2C 应用程序

目录

为包含数组的 IMS 事务创建 J2C 应用程序

本教程描述如何使用 J2C Java™ Bean 向导构建简单的 Web 应用程序来处理包含数组的 IMS™ 事务的输入和输出数据。

学习目标

本教程使您能够完成下列任务:

- 使用 J2C Bean 向导来创建运行 IMS 事务的 J2C Java bean。
- 为 Bean 创建 Java 方法以运行 IMS 事务。
- 创建测试代理 Java 类以测试应用程序

所需时间

30 分钟

相关信息

查看 PDF 版本

输入和输出数组样本

简介: 为包含数组的 IMS 事务创建 J2C 应用程序

本教程描述如何使用 J2C Java Bean 向导构建简单的 Web 应用程序来处理包含数组的 IMS 事务的输入和输出数据。

本教程可能需要一些可选可安装组件。要确保您已安装了适当的可选组件，请参阅“系统要求”列表。

本教程分为一些练习，必须按顺序完成这些练习，教程才有意义。本教程教您如何使用 J2C Java Bean 向导来创建运行 IMS 事务的 Java bean。在完成这些练习的过程中，您将执行下列操作:

- 使用 J2C Bean 向导来创建运行 IMS 事务的 J2C Java bean。
- 为该 Bean 创建 Java 方法 `runInOut.java` 以运行 IMS 事务。
- 创建测试代理 Java 类 `TestInOutProxy.class` 来为 IMS 事务构建输入消息，调用运行 IMS 事务的 J2C Java bean 方法，然后显示由 IMS 事务返回的输出数据。

  注: 已经为英语语言环境创建了 `TestInOutProxy.java` Java 类，您可能必须根据其他语言环境对此代码进行修改。

所需时间

完成本教程大约需要 30 分钟。如果研究其他与本教程相关的概念，则完成本教程可能需要更长时间。

技能级别

熟练

用户

本教程面向熟悉企业信息系统（EIS），特别是熟悉 IMS 的用户。

系统要求

为了完成本教程，需要安装下列工具和组件:

- IBM® WebSphere® Application Server V6.1
- J2EE 连接器（J2C）工具
- **与 IMS 环境有关的信息**: 在本教程中，您的应用程序与 IMS 中的应用程序交互。需要获取一些信息，例如，IMS Connect 的主机名和端口号以及将运行事务的 IMS 数据存储器的名称。请与 IMS 系统管理员联系以获取此信息。另外，如果想要运行此样本，则需要在 IMS 中执行一些设置工作。
- **COBOL 文件 `InEqualsOut.cbl` 的一个副本**: 可在以下产品安装目录中找到此文件: <installdir>\IBM\SDP70Shared\plugins\com.ibm.j2c.cheatsheet.content_7.0.0\samples\IMS\InOutArray。如果想将它存储到本地，可从以下链接复制其代码: ../resources/inequalsout.dita
- **一个干净的工作空间**。

要使用本教程，必须安装并配置了应用程序服务器。要验证是否提供了服务器运行时环境，请单击**窗口 → 首选项**，展开**服务器**，然后单击**已安装的运行时**。可以使用此窗格来添加、除去或编辑已安装的服务器运行时定义。还可以下载并安装对新服务器的支持。

必备知识

为了能够彻底完成本教程，您应该熟悉下列内容:

- J2EE 和 Java 编程
- 基本 IMS Transaction Manager（IMS TM）概念

课程 1.1: 选择资源适配器

本课程指导您完成一些详细的步骤来选择并配置资源适配器以连接至 IMS 服务器。

本教程使用了 COBOL 数据结构来描述 IMS 事务输入和输出消息。输入和输出消息是相同的，而且它们包含一个客户元素数组，后跟包含函数代码的单个字段。此数组最多可包含八个元素，但在本教程中输入到 IMS 应用程序中的只有三个元素，IMS 应用程序返回的只有四个元素。

本教程中使用的 IMS 事务不是其中一个 IMS 安装验证程序。它使用了 DFSDDLT0: 一个根据控制语句信息发出对 IMS 的调用的 IMS 应用程序。以下提供了用于本教程的 DFSDDLT0 控制语句。但是，要运行本教程，您必须为 DFSDDLT0 配置环境并提供必需的 JCL。本教程将 SKS2 用作此 DFSDDLT0 应用程序的事务代码。

DFSDDLT0 控制语句

```
S11 1 1 1 1    TP     1
L        GU
E      OK
E  Z0088 DATA  SKS2 03CN001Cathy Tang          CN002Haley Fung         X
                               CN003Steve Kuo           123456
WTO IC4JINOU: Single segment received from JITOC
L        GN
E      QD
WTO IC4JINOU: End of input segments from JITOC
L        ISRT  JITOC53
L  Z0113 DATA  TRNCD04CN001Cathy T.            CN002Haley F.           X
                               CN003Steve K.            CN004Kevin F.            65432X
                             1
E      OK
WTO IC4JINOU: Single segment inserted - 3 elements !!!!!!!!!!!!!

L        GU
```

连接至 IMS 服务器

1. 如果工作空间的右上角选项卡中未出现 J2EE 图标  ，则您需要切换至 J2EE 透视图。从菜单栏中，选择**窗口 > 打开透视图 > 其他**。将打开“选择透视图”窗口。
2. 选择 **J2EE**。
3. 单击**确定**。将打开 J2EE 透视图。
4. 在 J2EE 透视图中，选择**文件 > 新建 > 其他**。
5. 在“新建”页面中，选择 **J2C > J2C Java Bean**。单击**下一步**。
6. 在“选择资源适配器”页面中，选择 J2C 1.5 IMS 资源适配器。对于本教程，请选择 **IMS Connector for Java（IBM: 9.1.0.2.3）**。单击**下一步**。
7. 在“连接属性”页面中，取消选择**受管连接**并选择**非受管连接**。（在本教程中，将使用非受管连接来直接访问 IMS。）接受缺省的连接类名 `com.ibm.connector2.ims.ico.IMSManagedConnectionFactory`。在空白字段中，输入所有必需的连接信息。用星号（*）指示的必填字段有:
   a. **对于 TCP/IP 连接**:
      1) **主机名**: （必填）IMS Connect 的 IP 地址或主机名。
      2) **端口号**: （必填）目标 IMS Connect 使用的端口号。
   b. **对于本地选项连接**:
      1) **IMS Connect 名称**: （必填）目标 IMS Connect 的名称。
   c. **对于两者**:
      1) **数据存储器名称**: （必填）目标 IMS 数据存储器的名称。
8. 可从您的 IMS 系统管理员处获取连接信息。当提供了必需的连接信息后，单击**下一步**。

课程 1.2: 设置 Web 项目以及 Java 接口和实现

课程 1.2 将指导您完成创建 Web 项目以及 Java 接口和实现。

在开始之前，必须先完成课程 1.1。在本课程中，您将完成下列任务:

- 创建 J2C Java bean
- 创建动态 Web 项目

1. 在工作台中完成的所有工作都必须与一个项目相关联。项目为工作文件和目录提供了一个有组织的视图，并根据项目的类型对视图功能进行了优化。在工作台中，所有文件都必须位于一个项目中，因此在创建 J2C Java bean 之前，首先需要创建放置此 bean 的项目。在“新建 J2C Java Bean”页面的**项目名称**字段中，输入值 `InOutArray`。
2. 单击**项目名称**字段旁边的**新建**按钮以创建新的项目。
3. 现在，您将创建动态 Web 项目。在“创建新的源项目”页面中，选择 **Web 项目**，并单击**下一步**。
4. 在“新建动态 Web 项目”页面中，确保选择了下列值:
   a. **项目名称**: InOutArray
   b. **项目内容**: 接受缺省值
   c. **目标运行时**: WebSphere Application Server V6.1
   d. **配置**: 接受缺省值
   e. **将项目添加至 EAR**: 已选中
   f. **EAR 项目名称**: InOutArrayEAR
5. 单击**下一步**。

6. 在“项目构面”页面上，接受缺省值。
7. 单击**下一步**。
8. 在“Web 模块”页面上，接受缺省值。
9. 单击**完成**。
10. 可能会出现一个对话框询问是否要切换到动态 Web 透视图。单击**是**。
11. 在“J2C Java Bean 输出属性”页面上:
    a. 在**包名**字段中，单击**浏览**并选择 `InOutArray` 项目。单击**确定**。
    b. 在**包名**字段中输入 `sample.ims`。
    c. 在**接口名称**字段中输入 `InOut`。
    d. 在**实现名称**字段中输入 `InOutImpl`。
    e. 不要选中**将会话另存为 Ant 脚本**。
12. 单击**完成**。

课程 1.3: 创建 Java 方法

课程 1.3 将指导您完成创建 Java 方法。

在开始之前，必须先完成课程 1.2。在本课程中，您将完成下列任务:

- 创建 Java 方法
- 在 COBOL 与 Java 之间创建输入和输出数据映射

1. **首先，将创建 Java 方法**: 在“项目资源管理器”视图的“动态 Web 项目”中，展开项目 **InOutArray**。
2. 右键单击 JavaSource 中的 **InOutImpl.java**，然后选择**源 > 将方法添加至 J2C Java bean**。
3. 在“Java 方法”页面中，单击**添加**。
4. 在 **Java 方法名称**字段中，输入 `runInOut` 作为方法的名称。
5. 单击**下一步**。
6. **接下来，将在 COBOL 与 Java 之间创建输入数据映射**: 在“Java 方法”页面的**输入类型**字段旁边，单击**新建**。
7. 在“数据导入”页面中，确保**选择映射**字段是 **COBOL 到 Java**。单击 **COBOL 文件**字段旁边的**浏览**。
8. 浏览以查找 InEqualsOut.cbl 文件的文件位置。(可以在产品安装文件夹中找到此文件的副本: <installdir>\IBM\SDP70Shared\plugins\com.ibm.j2c.cheatsheet.content_6.0.0\Samples\IMS\InOutArray)。
9. 单击**打开**。
10. 单击**下一步**。
11. 在“COBOL 导入器”页面中，单击**显示高级**。
    a. 选择下列选项:

表 1. COBOL 导入器参数设置

参数	值
平台名称	Z/OS
代码页	IBM-037
浮点格式名称	IBM 十六进制
外部十进制符号	EBCDIC
字节存储次序名称	大尾数法

表 1. COBOL 导入器参数设置 (续)

参数	值
远程整数字节存储次序名称	大尾数法
带引号的名称	DOUBLE
Trunc 名称	STD
Nsymbol 名称	DBCS

b. 单击**查询**来装入数据。

c. 显示了一列数据结构。选择**数据结构**字段中的 **IN-OUT-MSG**。

d. 单击**下一步**。

12. 在“保存属性”页面中,

a. 对**生成样式**选择**缺省值**。

b. 单击**项目名称**旁边的**浏览**并选择 Web 项目 **InOutArray**。

c. 在**包名**字段中，输入 `sample.ims.data`。

d. 在**类名**字段中，接受缺省值 **INOUTMSG**。单击**完成**。

e. 不要选中**将会话另存为 Ant 脚本**。

13. 在“Java 方法”页面中，选择**将输入类型用于输出**。

14. 单击**完成**。

15. 单击**完成**以完成方法的定义。

课程 1.4: 创建 Java 测试类以测试应用程序

课程 1.4 将指导您完成创建 Java 测试类以测试应用程序。

在开始之前，必须先完成课程 1.3。在本课程中，您将完成下列任务:

- 创建 Java 测试类。
- 使用下面提供的代码编辑该类。
- 运行测试类来测试您的应用程序。

1. **首先，将创建 Java 测试类:** 展开 **InOutArray** 项目，展开 **Java 资源**，然后选择 **sample.ims** 包。
2. 右键单击并选择**新建**。选择 类选项以创建新的 Java 类。
3. 在 Java 类名中，输入 `TestInOutProxy`。注意，TestInOutProxy 类仅作为示例提供。根据您的 IMS 机器规范，可能需要更改事务代码。请咨询您的 IMS 管理员以获取事务代码。可以在代码中找到此语句 `input.setWs__trcd("SKS7 ");` 以进行更改。
4. 确保**源文件夹**字段包含 **InOutArray/JavaSource** 并且**包名**字段包含 **sample.ims.data**。
5. 单击**完成**。
6. 双击 **TestInOutProxy** 以便在 Java 编辑器中打开该文件。
7. 复制下面提供的所有代码，并将其粘贴到 **TestInOutProxy.java** 类中。替换编辑器中的任何现有代码:

   注: 已经为英语语言环境创建了 `TestInOutProxy.java` Java 类，您可能必须根据其他语言环境对此代码进行修改。

```
/*
  * Created on 4-Oct-2004
 *
  * TODO To change the template for this generated file go to
```

```
  * Window - Preferences - Java - Code Style - Code Templates
 */
package sample.ims;
import sample.ims.data.*;
import com.ibm.connector2.ims.ico.IMSDFSMessageException;

/**
  * @author ivyho
 *
  * TODO To change the template for this generated type comment go to
  * Window - Preferences - Java - Code Style - Code Templates
 */
public class TestInOutProxy
{
 public static void main (String[] args)
 {
  try
  {
   // ---------------------------------------------------
   // Create the formatHandler, then create the input
   // message bean from the formatHandler.
   // ---------------------------------------------------
   INOUTMSG input = new INOUTMSG();

   int sz = input.getSize();
   System.out.println("\nInitial size of input message is: " + sz);

   // ---------------------------------------------------
   // Don't set the length (LL) field yet... wait until
   // input message has been adjusted to reflect only
   // the number of array elements actually sent.
   // ---------------------------------------------------
   input.setWs__zz((short) 0);
   input.setWs__trcd("SKS7 ");

   // ---------------------------------------------------
   // Construct an array and populate it with the elements
   // to be sent to the IMS application program.  In this
   // case three elements are sent.
   // ---------------------------------------------------
   Inoutmsg_ws__customer[] customers = new Inoutmsg_ws__customer[3];

   Inoutmsg_ws__customer aCustomer1 = new Inoutmsg_ws__customer();
   aCustomer1.setWs__cust__name("Cathy Tang");
   aCustomer1.setWs__cust__number("CN001");
   customers[0] = aCustomer1;

   Inoutmsg_ws__customer aCustomer2 = new Inoutmsg_ws__customer();
   aCustomer2.setWs__cust__name("Haley Fung");
   aCustomer2.setWs__cust__number("CN002");
   customers[1] = aCustomer2;

   Inoutmsg_ws__customer aCustomer3 = new Inoutmsg_ws__customer();
   aCustomer3.setWs__cust__name("Steve Kuo");
   aCustomer3.setWs__cust__number("CN003");
   customers[2] = aCustomer3;

   // ---------------------------------------------------
   // Set the array on the input message.
   // ---------------------------------------------------
   input.setWs__customer(customers);
   input.setIndx((short) 3);

   System.out.println("\nInitial value of INDX is: " + input.getIndx());

   // ---------------------------------------------------
   // Reallocate the buffer to the actual size
```

```
        // --------------------------------------------------
        byte[] bytes = input.getBytes();
        int size = input.getSize();
        byte[] newBytes = new byte[size];
        System.arraycopy(bytes, 0, newBytes, 0, size);

        // --------------------------------------------------
        // Set the bytes back into the format handler and set
        // the length field of the input message, now that
        // we know the actual size.
        // --------------------------------------------------
        input.setBytes(newBytes);
        input.setWs__ll((short) size);
        System.out.println("\nAdjusted size of input message is: " + size);
        System.out.println("\nAdjusted size of INDX is: " + input.getIndx());

        // --------------------------------------------------
        // Set fields that follow the array after the input
        // message has been adjusted.
        // --------------------------------------------------
        input.setWs__func__code("123456");

        InOutImpl proxy = new InOutImpl();

        INOUTMSG output = new sample.ims.data.INOUTMSG();
        output = proxy.runInOut(input);

        short outndx = output.getIndx();
        System.out.println("\nOutput value of INDX is: " + outndx);

        Inoutmsg_ws__customer outArray[] = output.getWs__customer();

        for (int i = 0; i < outndx; i++)
        {
         System.out.println(
          "\n"
           + outArray[i].getWs__cust__name()
           + outArray[i].getWs__cust__number());
        }
       }
       catch (Exception e)
       {
        if (e instanceof IMSDFSMessageException)
        {
         System.out.println(
          "\nIMS returned message: "
           + ((IMSDFSMessageException) e).getDFSMessage());
        }
        else
        {
         System.out.println(
          "\nIMS Connector exception is: " + e);
        }
       }
      }
     }
```

8. 按 **Ctrl-S** 键以保存更改。
9. **接下来，将测试此应用程序**：展开 **InOutArray** 项目和 **sample.ims** 包。
10. 右键单击 **TestInOutProxy.java** 类并展开**运行**，然后选择**运行方式 > Java 应用程序**。
11. 您应该会在控制台上看到以下输出:

```
Initial size of input message is: 217
Initial value of INDX is: 3
Adjusted size of input message is: 92
Adjusted size of INDX is: 3
Output value of INDX is: 4
Cathy T.            CN001
Haley F.            CN002
Steve K.            CN003
Kevin F.            CN004
```

祝贺您！您已完成“输入输出数组”教程。

总结: 为包含数组的 IMS 事务创建 J2C 应用程序

本教程已指导您完成了为包含数组的 IMS 事务创建 J2C 应用程序所需执行的详细步骤。

在本教程中，您已经学习了如何使用 J2C Java Bean 向导构建简单的 Web 应用程序来处理包含数组的 IMS 事务的输入和输出数据。

已学习的课程

在完成这些练习的过程中，您学习了如何完成下列任务:

- 使用 J2C Bean 向导来创建运行 IMS 事务的 J2C Java bean。
- 为该 Bean 创建 Java 方法 `runInOut.java` 以运行 IMS 事务。
- 创建测试代理 Java 类 `TestInOutProxy.java` 来为 IMS 事务构建输入消息，调用运行 IMS 事务的 J2C Java bean 方法，然后显示由 IMS 事务返回的输出数据。

  注: 已经为英语语言环境创建了 `TestInOutProxy.java` Java 类，您可能必须根据其他语言环境对此代码进行修改。