Rational. software

Rapid Java and J2EE Development with IBM WebSphere Studio and IBM Rational Developer

Stephanie Parkin
Information Architect
IBM Software Group

Contents

Introduction

With increasingly complex IT environments, incredible pressure to deliver timely solutions and new security threats looming daily, companies have changed their view of application development. Organizations are moving from simple, separate systems to a holistic view of their overall computing environment that includes suppliers, partners and customers. IBM has articulated this shift as moving to an on demand world. The on demand organization can respond with flexibility to any customer demand, market opportunity or external threat.

To achieve this flexibility, organizations are creating application development solutions to solve their key business problems. As a result, software development has become a core business process. Integrated development environments (IDEs) can speed the transition to an on demand model by enabling rapid development of Web, Java™ and Java 2 Enterprise Edition (J2EE™) solutions. IDEs that contain rapid application development tools automate many time-consuming development tasks. Organizations want IDEs that are easy to learn and those that can maximize the productivity of developers with diverse backgrounds.

This paper describes how IBM WebSphere® Studio and IBM Rational® Developer products help speed up both the adoption of Java and J2EE for IT shops new to the technologies and the development process for shops already well-versed in Java technologies. It includes references to the IBM WebSphere Studio and the IBM Rational Developer families of products, but focuses on IBM Rational Web Developer for WebSphere Software Version 6 (formerly known as IBM WebSphere Studio Site Developer) and IBM Rational Application Developer for WebSphere Software Version 6 (formerly known as IBM WebSphere Studio Application Developer). Rational Web Developer and Rational Application Developer are collectively referred to as "Rational Developer" in this paper.

The importance of Java and J2EE technologies

The Java platform has several basic benefits as outlined below.

Multiplatform

The true benefits of Java technology lie in its deployment architecture. Since the Java platform relies on a single Java Virtual Machine (JVM), all Java programs can run on any system that has a JVM version. This JVM provides unparalleled portability across platforms and operating systems. Furthermore, existing Java applications can be easily adapted for devices with limited-memory resources. In essence, the Java platform extends users' computing power from the desktop to the resources of the Web. Java technology can help bridge different computing environments and use the power of the Web in your business applications.

Open standards

The Java platform has been developed through a unique Java Community Process that ensures that the language is based on open standards. An open consortium of companies (with IBM as a major contributor) defines the architectures and standards for the Java platform. Open standards allow other companies to code their own Java extensions and class libraries and to help shape the evolution of new Java technologies. IBM has proven its strong commitment to open standards through its active role in the Java Community Process and in the creation of the Eclipse platform. By using a language based on open standards, your company will not get locked into a proprietary environment that might suddenly go in a direction that does not suit your company's needs.

Object-oriented

From a programming viewpoint, Java is an object-oriented language, which means that it models real-world objects and describes their properties, interactions and behaviors. The object-oriented model lends itself well to reuse, because each component can be easily shared in other programs as long as its interactions with other external objects are well defined and follow some standard design patterns.

J2EE technology

For companies that are grappling with the need to develop multi-tier, multi-user, transactional applications that encapsulate complex business logic or access legacy data and systems, the Java platform has an enterprise version. J2EE simplifies the construction and deployment of multi-tier enterprise applications by basing them on standardized modular components. It also provides a complete set of services to those components and handles many details of application behavior—automatically—without complex programming.

The challenges of Java application development

Most IT shops recognize the need for Java technology, but they often are hampered by a lack of Java skills or daunted by the complexities of multiplatform applications that access heterogeneous systems. The majority of developers today fall into three basic sets: novice Java developers, legacy procedural developers, and experienced Java and J2EE developers. This section describes these users in detail and the challenges that each user set faces when developing Java applications.

Challenges for novice Java developers

Many companies today are experiencing a shortage of skilled Java and J2EE developers. Because these professionals are high priced, most companies need their existing employees to learn Java skills quickly; therefore, many developers trained in client/server programming technologies such as Microsoft® Visual Basic are scrambling to get up to speed on the Java language, especially J2EE technology.

Visual Basic was quickly adopted by masses of developers because of the simplicity and ease of use offered by these development tools based on intuitive point-and-click visual composition and object instantiation. Traditionally, Java technology has not offered an equivalent development approach, making the transition to the Java language difficult for Visual Basic® developers.

Rational Developer provides the ideal environment for novice Java developers, especially those developers familiar with Visual Basic. As we will explore in this paper, the new JavaServer™ Faces (JSF) technology in Rational Developer, coupled with its visual and diagram editors, and a data access framework based on Service Data Objects (SDOs) make it easy to develop Web applications without writing code.

Rational Developer uses perspectives and progressive disclosure of capabilities to tailor the user interface to the task of the developer. While it contains a wealth of features for expert developers, it effectively hides the complexity from less experienced users. The wizards, code assist features, integrated best practice guidance and interactive guides (cheat sheets) all guide new users through the application development process. Rational Developer is the ideal tool to get your staff up to speed on Java technology.

Challenges for legacy procedural developers

Many companies have procedural programmers who possess a wealth of knowledge about the company's business, legacy systems and databases, but who lack the object-oriented skills required to develop new e-business solutions. These developers are proficient in COBOL, RPG, C or other procedural languages, and might also be mainframe developers well-versed in subsystems like IBM CICS® or IBM IMS™.

Fourth-generation (4GL) programming languages bridge the gap between procedural programming and object-oriented programming. They let developers use a comfortable programming paradigm and then generate the required deployment code in a different language. IBM provides a 4GL called Enterprise Generation Language (EGL) that generates Java code for execution on IBM WebSphere Application Server. It is a simple procedural language—easy to learn for any programmer proficient in business-oriented languages. This language is available for the whole spectrum of e-business solutions supported by the Java language, including JavaServer Faces components, Struts-based Web applications, the creation and consumption of Web services, access to message queues and access to databases and legacy systems.

The EGL language in Rational Developer can help companies deliver innovative e-business systems without requiring programming teams trained in procedural languages to master the complexity of the J2EE platform.

Challenges for expert Java and J2EE developers

The third set of developers is already proficient in Java, J2EE and Web services technologies. These developers are hindered by the tedious and repetitive coding of low-level programming interfaces that have nothing to do with the application business requirements. These developers need a tool that automates much of the administrative programming so that they can concentrate on implementing business logic that solves unique problems.

Expert J2EE developers often spend the bulk of their time in more critical areas, such as ensuring a sound application design, verifying application performance and throughput requirements or resolving the most elusive and obscure application failures. A tool that automates these tasks can significantly boost the productivity of expert developers.

Rational Developer brings together tools for rapidly developing and deploying all the various components that comprise J2EE applications: JavaServer Pages™ (JSPs), servlets, Enterprise JavaBeans™ (EJB™) components, portlets, Web services and SQL queries. It provides Unified Modeling Language (UML) tools to help visualize and understand the structure of complex applications and a comprehensive set of testing tools to help with every step of quality assurance, from unit and remote system testing and automated test case generation to dynamic application performance analysis.

Large organizations with a mix of the three skill sets discussed need a development environment that the entire team can use for collaborating on development projects. They also need a tool that allows for specialization so non-programmers, such as user interface (UI) designers and information developers, can quickly develop their artifacts in the same environment as the rest of the development team. Because it provides tooling for all these different roles, Rational Developer speeds up the entire application development cycle.

IBM Rational Developer overview

Now that you know the benefits of the Java platform and understand the different types of users trying to develop Java applications, let's take a closer look at what makes Rational Developer unique in the world of Java IDEs.

Part of the IBM Software Development Platform

The IBM Software Development Platform is a set of integrated tools, best practices and services that support a proven end-to-end process for the application development life cycle. Rational Developer fits into a tools framework that supports structured application development, including modeling, proven design practices and patterns and an iterative development process that helps ensure that applications meet user requirements.

Based on the Eclipse platform

Eclipse is an open source, Java-based, extensible development platform for tools integration. Eclipse-based tools give developers the freedom of choice in an environment supporting multiple languages, platforms and vendors. Eclipse delivers a plug-in–based framework that makes it easy for your team to create, integrate and use software tools together.

Rational Developer is IBM's core application development offering based on Eclipse 3.0. It provides a comprehensive and productive application development environment for creating and maintaining J2EE-compliant enterprise application systems. It also includes many features not available in Eclipse. And because Rational Developer is built on Eclipse, development teams can adapt and extend the development environment with best-of-breed plug-in tools from IBM, IBM Business Partners and the Eclipse community to match their needs and maximize developer productivity.

While the Eclipse platform does provide an open source IDE that can be used to code applications, it also serves as a platform for building application development tools. Table 1 gives a quick overview of the features that Rational Developer provides beyond those included in Eclipse 3.0.

Feature	Usage
Page Designer	Create the visual layout and design of dynamic HTML and JSP pages
Web Site Designer	Manage the structure and navigate entire Web sites
JavaServer Faces support	Quickly develop rich Web user interfaces
Service Data Objects support	Provide a single API for interacting with multiple data sources and visual tools for quickly developing data-driven Web applications
Web Diagram Editor	Map and construct Struts-based Web applications visually
Enterprise Generation Language	Generate Web applications without coding in Java
Portal tools	Visually develop portlets, portals, themes and skins using JSF and Struts
Integrated Web services tools and wizards	Discover and use existing Web services and build, test and publish new Web services
Visual Editor for Java	Extend Eclipse Visual Editor by providing visual tools for binding data sources to controls. Data sources can be EJBs, Web services or JavaBeans
J2EE and EJB wizards and editors	Extend Eclipse by providing wizards, editors and builders to automate creation, testing and deployment of J2EE apps and EJB components. Also supports Xdoclet annotations for rapid deployment
Integrated IBM WebSphere Application Server and IBM WebSphere Portal unit test environments	Provide for unit testing of J2EE and portlet applications
Crystal Reports framework	Design graphical data reports and embed them in Web-based applications
Performance profiling tools	Extend Eclipse by providing thread analysis, additional execution performance views, custom runtime analysis probes and advanced memory leak detection
Component test tools automation	Extend Eclipse by providing structural, complexity and coverage metrics to help decide what to test next, data-pool driven testing, and Web service test generation from Web Services Description Language (WSDL) files
XML tools	Create, edit and transform XML documents
Relational Schema Center	Manage and access databases
Integrated UML Visual Editor for Java, EJB and data tables	Visualize and manage complex code
Code analysis tools	Continuously help ensure code quality and completeness
IBM Rational ClearCase LT	Provide version control and manage team programming

IBM Rational Developer provides features beyond Eclipse

Provides a complete family of tools

Various configurations of WebSphere Studio and Rational Developer help ensure that your IDE grows with your company's needs. As your e-business application requirements grow from simple Web applications to complex, integrated, cross-enterprise business solutions, your developers' skills are preserved and your investment remains protected. The core Rational Developer configurations provide more functionality as you move up the ladder; for example, Rational Application Developer contains all of the functionality of Rational Web Developer.

- ***Rational Web Developer for WebSphere Software*** *(formerly WebSphere Studio Site Developer) is an entry-level IDE for Web and Java developers and primarily used for building JSP and servlet-based Web applications, Java applications and Web services. It provides visual development with JavaServer Faces components and EGL for generating Java code.*

- ***Rational Application Developer for WebSphere Software*** *(formerly WebSphere Studio Application Developer) allows for more advanced J2EE development, including Enterprise JavaBeans (EJB) components. It supports portal and UML-based development, and contains IBM Rational ClearCase® LT for version control. Another version of Rational Application Developer is the WebSphere Studio Application Developer Integration Edition, which helps enable accelerated development and integration of composite business applications that deploy to the IBM WebSphere Business Integration Server Foundation. It provides a broad portfolio of rich application and technology adapters and J2EE-based visual workflow tools.*

- ***WebSphere Studio Enterprise Developer*** *adds support for COBOL and PL/1 development and for the development of applications that target legacy back-end systems such as CICS and the IBM @*server® *zSeries® family of servers. It also provides EGL code generation that outputs COBOL source code.*

The capabilities of Rational Web Developer and Rational Application Developer are also incorporated into Rational Software Architect. Rational Software Architect adds support for UML 2 modeling, patterns, model transforms and code generation, C/C++ development and Java application structural review and control.

This paper focuses on the rapid application development features of Rational Web Developer and Rational Application Developer. These products are referred to collectively as Rational Developer.

IBM Rational Developer accelerates Java and J2EE development

Rational Developer accelerates Java and J2EE development. Two basic Rational Developer design principles decrease the Java learning curve and increase programmer productivity:

- ***Hides complexity from novice users***
  Rational Developer, using a series of perspectives, organizes the users' interaction with various tools based on their role in the development team, project or task at hand. All perspectives share a common look and feel, which reduces the learning curve and helps developers organize complex projects and focus on the task at hand. However, these perspectives are particularly beneficial to novice Java developers, who may only need to access a subset of the many capabilities available in the rich Rational Developer environment.

  Since it is based on Eclipse 3.0, Rational Developer takes this idea a step further by showing features in a view only when a user actually needs them (progressive disclosure). Advanced users can also customize their perspectives to show the tools they use most often or other plug-in tools not included with the base product.

- ***Speeds development by eliminating many tedious tasks***
  For more experienced Java and J2EE developers, Rational Developer automates many tedious tasks that developers routinely perform. It automatically creates a program structure that conforms to J2EE standards and creates configuration files for specific types of projects like Struts and portlets. Rational Developer can also create skeleton entities for business logic, such as Web services and EJB components. Wherever possible, Rational Developer wizards and editors generate page-handling and component interaction code, significantly reducing developer work effort and allowing them to focus on application functionality rather than low-level "plumbing".

Rapidly develop Web applications

Rational Developer provides many features to speed up the development of interactive, form-driven Web applications. Companies that are creating lightweight, dynamic Web applications need a tool that can help them provide marketing information online, basic forms for gathering registration information or even an online catalog.

The roles involved for basic Web site development usually include an interface designer or Web master, a content creator and a Web developer (smaller shops might have a single person performing all these roles).

Rational Developer supports all of these roles in the Web perspective. The Web master can design the overall Web site flow in Site Designer, prototype the Web page layout with the Page Designer tool and add controls without writing Java code. Web developers with minimum Java expertise can add dynamic elements to Web pages, such as a news feed or a registration application. They can rapidly lay out dynamic Web pages as JSPs and connect to existing databases, all without writing code. If the application will run on a portal server, Web developers also can create portal applications to improve usability and provide customized interfaces for end users. Additionally, they can exploit existing frameworks for Web development, such as the Struts framework, which enforces good Web application design principles that make it easier to subsequently modify applications.

Easily create, import and manage Web sites

The Web Site Designer tool helps developers create and manage the organization and layout of the individual pages in Web sites. They can quickly develop a Web site structure, create a navigation bar and create a site map. The tool provides page templates that enforce a consistent design for the Web site. Web Site Designer also gives a graphical view of the Web site to help developers understand how their users navigate through the site.

The Navigation view shows the site's pages as a hierarchy and represents their static navigational relationships. From this view, developers add, delete and move pages in the Web site. This view also displays a list of pages in the Web project that are not currently in Site Designer, which helps developers keep track of pages yet to be added.

The navigation information is used to automatically generate navigation menus, links and site maps. When pages are moved, the corresponding navigation links are automatically regenerated to reflect changes in the site's structure. The resulting menus, links and maps are standard HTML and can be deployed to any HTTP server. Rational Developer also lets developers import existing Web sites for redesign with Web Site Designer.

Without this capability, developers would have to create each navigation element by hand with a Web page editing tool, manually creating page templates without the ability to easily enforce their use.

After designing the static structure of the site, developers can create a Web diagram that shows the relationships between the entities that comprise the site and maps the logic-driven navigational flows. The Web Diagram Editor provides an intuitive facility for laying out the Web pages, business logic and data that passes through the application. It can contain any combination of JSF and Struts elements in addition to regular HTML pages, JSPs and JavaBeans. Designers can visually code navigation between pages and specify simple text links or JSF or Struts logic-driven flows. The Web Diagram Editor speeds up development of new Web sites, but it can also help designers to understand the structure of existing sites. Figure 1 shows the Web Diagram Editor.

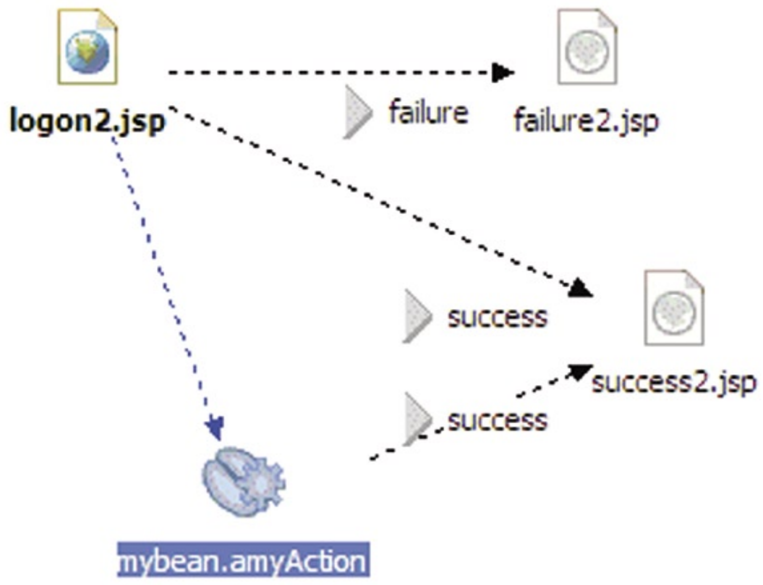


Figure 1. The Web Diagram editor helps speed development of new Web sites

Without the Web Diagram Editor, developers would have to manually update the navigation elements on each page and manually code the XML configuration files that manage the navigation flows when using Struts or JSF.

Build dynamic Web pages without coding
Page Designer lets Web designers quickly lay out a Web page using an interface similar to other what-you-see-is-what-you-get (WYSIWYG) Web editors. It provides tabs for viewing the page in Design mode (WYSIWYG composition), Source mode (code-level composition) and Preview (page displayed as it will appear in the browser at runtime).

Designers can quickly add images, links and forms to the page by simply dragging and dropping from a palette of components. They can work with a library of provided images, create their own and even create animated images—all from a single workspace. They also can add JSP tags, Java applets or embedded scripts. These tools let your Web designer work with elements that might have been created by a Java developer or a graphic designer in a single environment. They can also add a JavaBean to the page and access its properties and methods through a simple wizard-driven interface.

The Source Code view provides syntax highlighting and formatting and code assist, a feature that lists valid alternatives for completing the current tag so that developers do not need to look up the syntax. Rational Developer also provides context-sensitive help for the current tag being edited.

Many sites use JSP include files to encapsulate common elements used in multiple pages. On each page that calls the include files, Page Designer displays the contents of the include file to enable quick editing.

To quickly design a new Web page, designers can start with one of the many sample Web sites shipped with Rational Developer and modify as needed. They can also use predefined page templates.

The tight integration with the other views and perspectives helps developers lay out a Web page in Page Designer, switch to another view to create dynamic elements like servlets or portlets and then switch back to Page Designer to add the elements onto the graphical use interface (GUI). Since Rational Developer provides an integrated WebSphere Application Server runtime environment, designers can test the page immediately within Rational Developer, without having to deploy the artifacts to a server. Without Rational Developer, developers would have to design Web pages in one editor and then create dynamic elements in a different code editor. They would only be able to test them on the server after deploying all of the elements.

Rapidly develop interactive Web pages with JavaServer Faces

JavaServer Faces is a powerful new presentation standard and framework which, when combined with appropriate tools, significantly simplifies the construction of Web interfaces and Web applications. Prior to the introduction of JSF, developers had to manually write code to handle almost all user interactions with the application, such as validating input, checking for errors, and validating and converting strings. The JSF framework consists of user interface (UI) components and a runtime that renders these components to the client and manages the page life cycle (errors, validators and navigation, for example). Using the visual JSF tools in Rational Developer, you can create rich, dynamic Web sites without writing a single line of code.

The new JavaServer Faces tools provide an easy way to rapidly develop user interfaces for J2EE applications and to integrate relational databases, eliminating a lot of the hand-coding we used to do, and making our Lotus® workplace development faster and more efficient."

*—Dr. Rolf Kremer
Chief Development Officer
PAVONE AG
Germany*

Rational Developer provides intuitive wizards and visual UI components for creating JSF files and the Web Diagram Editor to help organize the JSF application. The wizards do all the set-up work needed to use JSF in an application. Rational Developer provides the base JSF components, plus over 20 additional objects created by IBM that conform to the JSF specification. These components include a file upload feature, a rich text editor, charts, tabbed panels and an enhanced data grid. IBM has further extended some JSF components to download client-side data for a more interactive user experience without time-consuming trips to the server.

Novice developers can use this functionality without working with the generated code, but more advanced Java developers will want to view the source that is generated by the JSF editor. Rational Developer creates code-behind files that contain the page source code. The source is dynamically generated as you edit the JSF design surface—there is no additional generate step, so the code and UI design are always in sync. Developers can control whether the code-behind files are displayed in the Web Diagram.

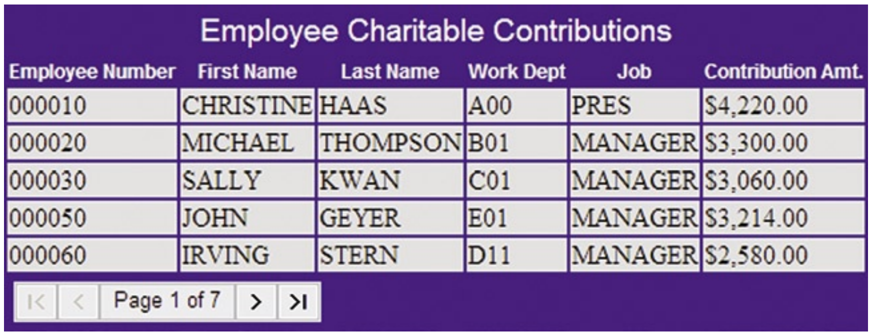
Quickly add dynamic data to Web pages

Rational Developer makes it easy to access, create, display, update and delete dynamic data in Web pages. Data objects can encapsulate a data source located in a relational database, a JavaBean, an EJB or a Web service. The Service Data Object (previously known as the WebSphere Data Object) sits between the Web application and the data source, and provides a consistent interface to any relational database, EJB, JavaBean or Web service. Developers do not have to write the data-access code; the Rational Developer tools for the Service Data Object write it for them.

Rational Web Developer for WebSphere Software (formerly WebSphere Studio SiteDeveloper) makes it easy to build data-driven Web applications. Using JavaServer Faces components and Service Data Objects, a simple CRUD (create, read, update and delete) database application can be built with eight clicks and no coding.

This gives Web developers the option of not creating a data bean; they can simply drag and drop the data object on the page, fill in the fields in the wizard and then design how they want the user to interact with the data (or even let Rational Developer generate the UI). They can work with a tabular view of several rows of data (a data grid) or work in detail with a single data record.

They can then use the JSF data grid to make a row selectable, add columns to rows, set widths and alignments and even handle paging through data (for example showing five rows at a time). The data column can contain images, hyperlinks and more. The generated code helps to ensure that the interactions and data caching with the database are optimized, which alleviates the need to roundtrip to the server each time another row of the table is accessed. Figure 2 shows a data grid created from a Service Data Object (SDO) object. Remember, no coding was needed to create this application, not even for the data connection:


Employee Charitable Contributions

Employee Number	First Name	Last Name	Work Dept	Job	Contribution Amt.
000010	CHRISTINE	HAAS	A00	PRES	$4,220.00
000020	MICHAEL	THOMPSON	B01	MANAGER	$3,300.00
000030	SALLY	KWAN	C01	MANAGER	$3,060.00
000050	JOHN	GEYER	E01	MANAGER	$3,214.00
000060	IRVING	STERN	D11	MANAGER	$2,580.00

Figure 2. Data grid created from a Service Data Object with no coding needed

EJB session JavaBeans can also be directly accessed from a Web page, including EJBs that call SAP and Siebel entities. When an EJB is added to the page, Rational Developer can automatically generate the input and output forms, or developers can design their own through a set of wizards. Rational Developer generates all of the access code and descriptors.

Rapidly develop structured Web sites with the Struts framework

The Struts open source framework helps separate business logic from user interface elements—making both easier to maintain—and provides a proven architecture that performs well for the majority of interactive Web sites. The Struts tools in Rational Developer let novice developers create well-architected Web applications without coding the layout and connecting code. You can even combine the power of the Struts architecture with a JSF user interface.

These Struts tools include a Struts Explorer and the Web Diagram Editor. The Struts Explorer helps developers understand and navigate the various pieces of a Struts application. When a new Struts project is created, Rational Developer builds a Web diagram that shows the structure of the application. The developer then draws connections that indicate how the application flow proceeds. As the diagram is edited, Rational Developer updates the Struts XML configuration files and generates Java code.

The Web Diagram editor can obviously save time in helping organize and visualize dynamic Web applications, but the real time-saver is the code generated when a Struts application is defined. Developers only need to write the code to perform the action—all other code for managing the data and passing it between forms is created by Rational Developer and encapsulated in reusable JavaBeans.

Without this feature, developers would have to create each of the individual elements by hand, manage their interactions, manually create the Struts configuration file and ensure that they have separated business logic from interface elements.

Generate Java and Web applications with EGL
The Enterprise Generation Language (EGL) provides a 4GL rapid application development environment for developing both Web and traditional character-based applications. EGL evolved from IBM VisualAge® Generator to help enable the movement of applications to the Web. Rational Developer extends EGL by incorporating IBM Informix® 4GL constructs and functionality. EGL is the strategic IBM language for providing procedural developers an evolutionary path to Java.

The EGL language is a simple procedural language that is comfortable for COBOL, Informix 4GL, Oracle Forms, report program generator and other 4GL programmers. Using the EGL language in Rational Developer, your procedural developers can rapidly develop Web and J2EE applications using familiar programming constructs.

EGL is a fully supported language in Rational Developer, enabling procedural developers to become immediately productive in J2EE projects. For example, EGL is integrated into the new JavaServer Faces implementation. This linkage allows for rapid development of JSF-based applications with all of the page logic written in EGL rather than in the Java language. Developers can drag and drop records and data items defined in EGL onto a JSP while the JSP is being defined, automatically establishing the linkage between the JSP and the EGL data item.

Rational Developer provides a special server-side event (called onPageLoad) that lets developers retrieve data and manipulate the UI component tree after it is created, but before it is displayed (similar to GUI “aboutToOpen” events), which enables more intuitive controller logic. This event helps procedural programmers map their traditional text-based applications to more GUI-driven designs.

All productivity features of the Rational Developer context-sensitive editors and debuggers also apply to EGL. Developers can debug entire applications seamlessly without viewing the generated Java code.

Rational Developer EGL supports rich data, which lets developers define just once all the validations, formatting and display information for data items. These rules apply wherever the data item surfaces (TUI, Page and business logic, for example). The EGL runtime automatically runs the application and displays appropriate messages.

Because the Informix 4GL language only supports text-based interfaces on the UNIX® platform, Informix developers have not been able to connect browser-based GUIs to Informix databases. EGL offers legacy Informix customers an evolutionary path from the Informix Data Server (IDS) to the WebSphere Application Server. Rational Developer provides a new migration tool that automates the conversion of Informix 4GL code to the EGL language in Rational Developer 6.0.0.1, a patch that will ship shortly after the 6.0 release.

Rational Web Developer for WebSphere Software and Rational Application Developer for WebSphere Software can generate Java code to deploy to Microsoft Windows®, AIX®, Linux® and IBM i5/OS™ environments. Enterprise Developer can also generate COBOL code to deploy to the IBM @server iSeries™ and IBM @server zSeries families of servers (CICS and batch). EGL also provides connectivity to IBM Informix Dynamic Server, IBM DB2® database management and other relational database servers through Java database connectivity.

Quickly debug and test Web applications
Because developers spend so much time debugging applications, Rational Developer is stocked with a very powerful set of debugging facilities. All of the editors mentioned above provide a Task view that gives visual cues to any possible problems, including broken links. The Link view also identifies broken links, and the Link Utilities feature can either repair broken links or globally convert links to a new root when an application's directory structure changes. The Source Code view also provides visual cues to identify possible problems.

When running the application, developers can easily set breakpoints and step through all types of code, including JSPs, servlets and even JavaScript. They can run snippets of code to view the results before doing full testing, and can even change the code while it is running, updating it on the fly.

Rational Developer comes with an integrated instance of WebSphere Application Server. With a single click, a developer can run an application on a test server without the overhead of publishing the application or installing it onto a separate server. They also can view the application running exactly as it would in the target deployment environment.

As a 15-year Informix 4GL developer, we are excited about the possibility of reusing our existing skills to create new Web applications using Enterprise Generation Language."

— Patrick van Dorsselaer
Senior 4GL Developer
Informa
Netherlands

Rational Developer includes multiple versions of the application server to allow testing of applications that target an older version of the runtime. The Rational Developer Server Test Environment can also be configured to test a Web application on a Tomcat or BEA WebLogic server via a toolkit. Rational Application Developer adds a WebSphere Portal 4.2 and 5.0 test environment.

Rational Developer also supports publishing to remote servers. Developers can configure, start and interact with WebSphere Application Server defined on a remote system (Windows, Linux/Intel, AIX and iSeries). The remote server can be either a physically separate machine or another server instance on the same machine running outside Rational Developer. This lets your team use a single dedicated test machine to simulate a target production environment with multiple servers.

Rapidly deploy a Web application

Deploying a Web application consists of simply configuring a server instance and then publishing the application to that server. Static Web applications can be published to any HTTP server and dynamic applications can be published to WebSphere Application Server, Tomcat or BEA WebLogic. New Java Runtime Environment definitions can also be added to Rational Developer and can target different Java Development Kit levels. Since the robust Server Test Environment lets developers do most of the server-side testing on their client, deploying the application becomes almost trivial.

Rapidly develop Java and J2EE applications

In addition to Web application development tools, Rational Developer provides many rapid application development tools for programmers involved in creating more complex Java applications that require more Java programming knowledge and most of the logic to run on the server. They usually are working with more complex coding structures and need more control over data access mechanisms. In addition to the tools used by Web application developers, Java application developers need tools to help them quickly write JSPs, servlets, portlets and Web services and that let them test on servers. They need tools to help design the overall application architecture and keep track of all the various artifacts that make up their application. And, of course, advanced J2EE developers need tools to help them quickly code business logic using the EJB component model.

Visually program Java applications
Rational Developer provides Java editors, including a visual editor and a traditional text-based editor. The Rational Developer Visual Editor for Java is based on the JavaBeans components model, which defines the interface to a JavaBean through methods, properties and events. This editor lets developers visually design the GUI components for an application and then implement the necessary JavaBeans to add application functionality.

Unlike other visual editors, the Visual Editor for Java is a code-centric editor that performs round-tripping of changes between the JavaBeans and the source code. Developers can modify the JavaBeans and see the modifications reflected in the source, and can also change the source code and see the changes applied to the JavaBeans. The Visual Editor for Java also displays the JavaBeans and the source code side by side using a split pane. When a JavaBean is selected, the source code shows the method that initializes it; when an individual property is selected, it shows the statement that sets its value.

The Visual Editor for Java helps programmers quickly create Java applications by providing an interface similar to other visual construction tools. The layout tools help them quickly prototype an application for user validation and reuse JavaBeans from Swing, SWT or AWT component libraries or import them from other libraries. Visual binding tools let developers quickly connect visual components to J2EE artifacts to bind components to data sources (including EJBs, Web services or JavaBeans). Although developers do not need to understand the details of the binding code, they can modify the generated code to optimize performance.

Speed up coding of Java and J2EE applications
The text-based Rational Developer Java editor provides several features that accelerate Java development. The editor provides visual aides, such as syntax highlighting and code assist, for completing the current language element.

Rational Developer provides automatic incremental compilation, which means that the code editor can give immediate feedback about problems in code, helping eliminate many bugs before the program is compiled.

Coding is also accelerated through the ability to reorganize code without having to manually fix all references to specific components. Re-factoring code means changing the code while preserving the behavior. The re-factoring tools enable reorganization of program elements from any editor, even the renaming of elements. Rational Developer will dynamically update all references to the code and preview the impact of the changes before making them permanent. This process saves developers a huge amount of time and effort spent locating and changing references in code.

For EJB development, Rational Developer eliminates much of the handwork and helps ensure that the application complies with J2EE standards. It contains several wizards and guides that lead developers through the development process and generates wrappers for most code artifacts.

Rational Developer also simplifies EJB client access by generating common code patterns for creating or finding EJBs with the EJB Snippet Wizard. The Session Bean Façade pattern simplifies client access to container-managed persistent (CMP) EJBs. The Session Bean Façade is a Session EJB that is created as a client interface and uses Service Data Objects to exchange data with the persistent EJB.

The Java editor recognizes EJB components and offers several extensions, such as a task list, that provide immediate feedback on the correctness of the EJB components and other J2EE artifacts. The code editor verifies that different types of Enterprise JavaBeans are constructed correctly and that they are consistent with other associated EJBs. Rational Developer also provides special editors for viewing and modifying EJB deployment descriptors as well as tools to create, edit and validate Enterprise Archive (EAR) files.

To further automate deployment, for most EJB patterns, developers can specify deployment information through Xdoclet metadata tags in the Java classes. Rational Developer supports the standard Xdoclet tag set. These tags let Rational Developer generate all the necessary deployment artifacts at runtime, which lets developers maintain a single EJB artifact rather than a collection of classes and deployment descriptors.

Rational Developer can generate cross-EAR EJB references and generate EJB components that tie into transaction processing systems, and developers can quickly turn their EJB components into Web services. These EJB features work together to provide a J2EE structure for developers so that they can concentrate on coding business logic.

The UML Visual Editor for Java and EJB also increases developer productivity by giving a graphical view of the structure of an application. Developers can implement classes visually and manage the complexity of their programs through intuitive UML diagrams.

Rapidly develop data-access code

One of the biggest challenges for developers has always been the integration of their Java applications with back-end data systems. Usually developers need access to existing data that has a predefined structure that they need to bring into their programs. Occasionally, developers who are not familiar with data modeling must develop the data structures themselves. Rational Developer provides tools that automate much of the coding for both scenarios by importing a data structure from an existing data source or creating a new data model within Rational Developer and exporting it to the actual data source. It also provides a Data view to organize all these tools.

This section describes just a few of the data tools that speed up the process of managing data in J2EE programs. These tools are not tied to a particular database; most of them support the major database vendors.

Rational Developer delivers tools for visual or programmatic access to EJB data via the Service Data Object. It provides a wizard that generates connection code to let developers visually bind the data to controls on a JSP. It also generates an EJB Session Bean façade that lets developers write code to access the SDO datagraph for EJB data.

Additional wizards that help quickly create certain kinds of EJB components are provided by Rational Developer. In particular, the EJB-to-RDB (relational database) wizard creates entity EJB components from a predefined data structure in a relational database, a process called bottom-up mapping. The tools in Rational Developer import the target data structure and handle all the mappings, which eliminates most of the coding in this type of EJB development.

Rational Developer also provides wizards for top-down mapping for EJB development, for both CMP JavaBeans and bean-managed persistent (BMP) JavaBeans. After defining the data structure, developers can map it to the data source using a simple map browser that shows the Enterprise JavaBeans and the database tables side by side. They can drag and drop entities from the EJB table onto the database tables, and then Rational Developer creates all the underlying mapping code.

These wizards provide incredible time savings in creating and mapping EJB components. Without them, the developer would have to manually code the structure of the EJB component to match the data source or vice versa. Figure 3 shows the Map Browser.

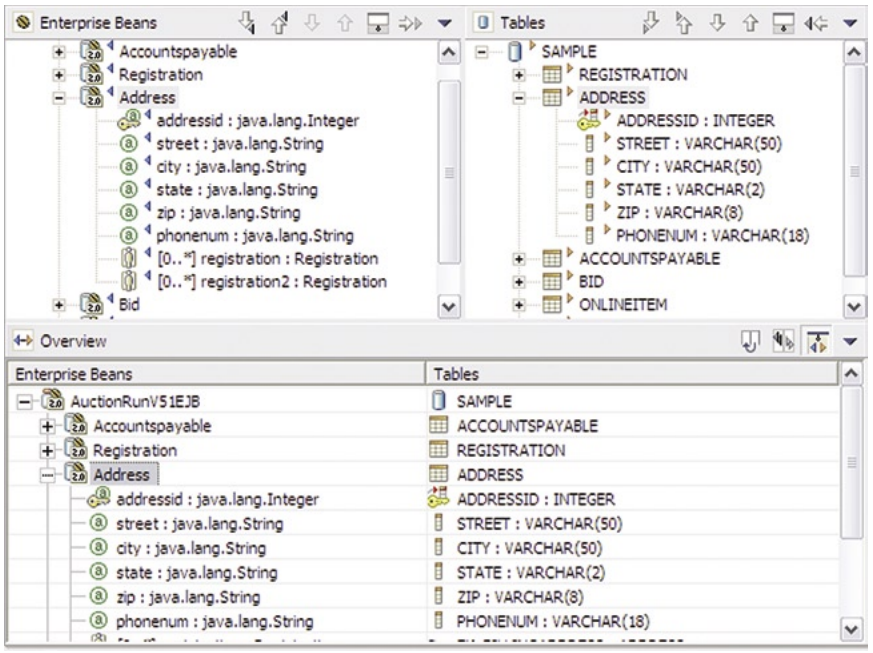


Figure 3. The Map Browser provides time savings in creating and mapping EJB components

The SQL query builder provides a visual interface for creating and executing SQL statements—from very simple statements to complex expressions and groupings. It also provides an SQL-to-XML wizard to generate XML artifacts for use in other applications, such as a servlet or JSP.

The query builder provides a graphical view of the tables and columns in the database. It lets developers select the columns to work with, link them to specific Join operations and build expressions to qualify the data to be retrieved. Developers can also execute the query directly in the editor to verify that it works as intended, define host variables that the user enters and call stored procedures from the query.

Quickly create high-performance data reports
Rational Developer provides Crystal Reports integration for creating high-quality reports. Crystal Reports is a powerful reporting framework for designing flexible, feature-rich reports that can access virtually any data source, including XML, JDBC and enterprise data sources. Extensive formatting options provide complete control over data presentation and make it easy to deliver information in a way that is consumable for end-users.

The Crystal Reports feature lets developers create dynamic, visual data reports up to 180 times faster than by hand-coding. They can design reports within Rational Developer or use predefined report templates that provide a wealth of common chart formats, such as pie, bar, Gantt and many more. Also included is a JSF Report viewing component that developers can use to incorporate reports into their Java applications. This viewer allows them to render reports within Web pages running on their application server.

For larger enterprise reporting requirements, Rational Developer provides a test and development license of the Crystal Enterprise servers (Embedded and Professional). This feature provides a rich set of JSF components, enabling developers to deploy enhanced Enterprise Reporting applications with more advanced features like security and scheduling. These reporting environments give developers a highly scalable reporting framework to manage report deployment on an enterprise scale.

Quickly edit and debug SQL stored procedures

For expert developers who require more control over the SQL query, the SQL editor provides all the features of the code editor (such as syntax highlighting and code assist) to help them quickly write their own queries by hand. Rational Developer has special code editors for writing SQL stored procedures and user-defined functions (UDFs) for DB2. The source-level debugger for SQL stored procedures supports all standard Rational Developer debugging tools, including running on a server without packaging and deploying the stored procedures.

Visualize and edit code with UML

The UML Visual Editor for Java and EJB tool provided with Rational Application Developer gives a graphical view of the code and data objects. It displays code artifacts as Unified Modeling Language class and sequence diagrams, which show the structure and relationships of Java classes, data objects, interfaces and EJB components. UML notation is an industry standard for object modeling, ratified by the Object Management Group. This view of an application helps with organization and documentation for other users of your code. Even for developers who are not familiar with UML notation, the intuitive visual design can help keep track of complex applications that contain many artifacts.

The UML Visual Editor now provides three categories of diagrams:

- ***Browse diagrams****—help developers navigate their code without needing to create and manage their own diagrams. These diagrams are analogous to a Web browser that lets developers explore the web of implementation artifacts in their workspace to discover the underlying structure.*
- ***Topic diagrams****—help developers document their code and ensure their documentation stays current. These diagrams query the current state of the application to show relationships between classes and program hierarchy.*
- ***Edit diagrams****—help developers create diagrams of their classes, which can be used to design and implement aspects of their applications (including Java classes, EJBs and data objects). The diagrams illustrate the developing relationships between classes as they code them.*

All of these diagrams help developers document their implementation and keep that documentation up to date. Both the Edit and Topic diagrams can be inserted into Javadoc files to provide a visual map of their program (clicking on any class in the diagram takes the user to the Javadoc for that class). Figure 4 shows a browse diagram:

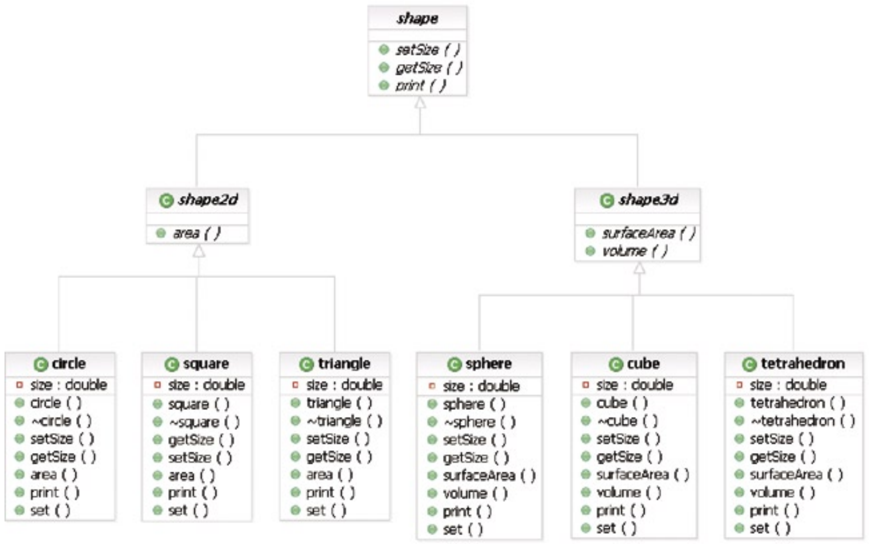


Figure 4. The Browse diagram helps developers document their implementation

The UML Visual Editor also aids with component implementation; for example, it provides powerful graphical editing capabilities for EJB components.
The object model is saved along with the rest of the J2EE project and is used throughout the object's life cycle, including during testing and deployment. Because the visualization is derived dynamically, it is always synchronized with the underlying code. Any changes made in other editors, like the deployment descriptor editor or the Java editor, are immediately reflected in the class diagram and vice versa. The UML Visual Editor is an easy way to re-factor code; as objects are dragged and dropped in the model, the references are automatically updated in the underlying code.

The UML Visual Editor supports both IE and IDEF diagrams, helping developers visualize data objects; they can simply drag and drop a data object onto the UML editor to visually model their data. These UML visualization tools provide an easy-to-read, up-to-date UML view of the program's implementation.

Automate the build process

One extremely tedious task for programmers is bringing together all the various pieces of an application into a build for system integration testing. This task has to be performed frequently, as often as weekly or even daily. Rational Developer allows automation of this and other frequent tasks through its support of the open source Ant tool from the Jakarta Apache project.

Ant is a Java-based build tool that has become a de facto standard for building Java projects. The build scripts are XML files containing targets and specifying the tasks (operations) required for each. Ant comes with a large number of built-in tasks that can perform many common build operations.

Running Ant scripts is a built-in feature in Rational Developer. It supports running any XML file as an Ant script simply by clicking Run Ant. It displays Ant messages in an Ant Console view to help debug Ant scripts.

Quickly test J2EE applications

In addition to the unit test environment described above, Rational Developer provides testing features for finding and fixing defects early in the development cycle and automating regression testing. When unit testing parts of a J2EE application, such as EJB components, Rational Developer can automatically configure the server to access data sources and generate EJB deployment code. Developers can even modify the application on the fly—the server will detect the updated files and immediately reflect the changes. This tight coupling provides an extremely productive testing environment due to the reduced overhead of republishing and change deployment.

The universal test client (UTC) is a Web application that makes it easy to test server-side components like servlet and EJB modules without creating a custom application to drive the component. Developers interact with the UTC via a browser, either the one built into Rational Developer or an external one. This client lets developers set breakpoints, define parameters to pass to JavaBeans and fully test the functionality of the component.

Any Java class can be loaded via the UTC, which makes it perfect for invoking multiple application paths, including the Java proxy generated for Web services. Without this feature, developers would have to manually create a test client that covers all test cases for an application and deploy the test client to the target server.

Rational Developer also provides component test tools for building, managing and automatically generating test cases for unit and system testing. The Component Test view lets testers manage manual test cases and automate testing of Web sites and Java applications running locally or remotely. The Component Test view is based on the Junit framework and the open source Hyades project. The test creation wizards offer a set of test patterns to automatically generate tests from code analysis and user selection. Rational Developer provides proven patterns for testing various components, including testing the life cycle of EJBs. It also supports data-driven testing, which allows for reuse of test scenarios with multiple sets of data. The tests can be automatically deployed and run on the application server, and the results are captured for later analysis.

This wealth of features automates much of the component test cycle; without them, the test team would need to create and manage test cases manually, hand-coding programs to call the components that require testing.

Continuously ensure quality

While testing an application for errors is important, developers today need to ensure that their code meets user requirements and adheres to architectural guidelines and standards. In addition to the component testing tools, Rational Developer provides a suite of tools to automate code reviews that include and extend the runtime analysis capabilities of IBM Rational PurifyPlus™ for the Java language.

The Code Review and Architectural Review tools let developers test their programs against a set of rules or patterns to improve performance, maintainability and readability. The tools come with a set of predefined rules that test for J2EE best practices in areas such as globalization, performance and scalability. Developers and architects can also define additional rules so that a team develops to the same standards and conventions.

Rational Developer provides a simple interface for configuring and running code reviews. After an architect or team lead configures the rules for a project, the developer simply selects the rules to run against the application, presses a button and then reviews the results, which provide detailed information on how to fix the code—the Quick Fix option can even automatically fix the code.

Architectural reviews can be automated in a similar way, which helps teams ensure that their programs meet the original business requirements and manage the complex relationships between program interfaces. The architect defines architectural constraints for the components and their relationships and records them as rules based on predefined templates. Developers then check their components against these rules to verify the integrity of the architecture and report all violations of the architectural constraints.

This suite of Code Review tools can significantly improve the productivity of a team by helping to ensure program quality and correctness throughout the development cycle rather than at the end. These tools help architects and team leaders enforce a coherent architecture, consistent coding guidelines and development best practices without time-consuming manual code reviews.

Rapidly deploy J2EE applications

Through annotation-based programming, Rational Developer can generate EJB and J2EE deployment information needed at runtime, so that developers create and maintain a single artifact. Developers specify deployment information through Xdoclet metadata tags in the Java classes. This feature helps enable the automatic installation of applications and modules onto a running WebSphere Server and supports deployment to a staging server rather than just a unit test server.

The tight integration between the development and runtime environments means that developers do not need to waste time reconfiguring the server when components change. Through incremental updating, Rational Developer only redeploys the parts of the application that have changed, which makes the test cycle much more efficient. Developers can spend less time restarting the server and more time focused on program quality.

Rapidly develop portal applications

A portal provides a mechanism for aggregating information and accessing enterprise services via a single consolidated view. A portlet provides access to a specific application or function that is available via the portal. The IBM Portal Toolkit is now fully integrated into Rational Application Developer, which helps developers to create, edit and test portals and portlets just as they would any other type of programming artifact. They can use the portal tools in combination with JSF and Struts elements, EJBs and even Web services.

Rational Application Developer supports development of projects targeted to WebSphere Portal Server 5.0.

For example, developers can visually create and edit portal applications and visually edit the themes and skins that control their appearance. They can also manage complex portlet projects that render portlet JSPs, locate and use Web services and manage data. Rational Application Developer also includes a Portal Test environment for unit testing both portal and portlet behaviors and system testing for interactions with other programming artifacts.

A very common portal application is to provide a managerial dashboard to enterprise applications such as SAP and Siebel. Rational Developer now provides portal-based mediators that make it easy to connect to these back-end systems. Developers can drag and drop connectors onto the portlets and fill in simple wizards—Rational Developer will then generate the code needed to access data stored in these systems.

Generate portlets with wizards

As with other types of applications, developers create a new portlet using Rational Developer wizards, which generate a portlet project structure that conforms to J2EE standards and create a complete portlet. Rational Developer also automates the creation of the portal deployment descriptors and EAR files.

The wizards create two main types of portlets: those that comply with the IBM Portlet application programming interface (API) and those that comply with the JSR 168 architecture. JSR 168 is a Java specification from the Java Community Process that addresses the requirements of aggregation, personalization, presentation and security for portlets running in a portal environment. The IBM Portlet API extends the basic features of JSR 168.

Create portlets visually

Developers can lay out the interface for portlets using Page Designer and create portlets using the Struts framework and the Web Diagram Editor to visualize the structure. In addition, they can use JavaServer Faces UI components to visually develop highly interactive portlets.

One of the biggest time savers in the IBM Portlet API is its brokered approach to inter-portlet communication. The Click-to-Action approach to messaging speeds up development of portal applications, because portlets do not need to be aware of each other to send information back and forth. Rational Developer provides a drag and drop component for implementing Click-to-Action without programming and it uses an icon to visually indicate portlets with Click-to-Action enabled.

Create portals visually

The new Portal Designer tool lets developers lay out a Portal interface visually without writing any code. They simply select a visual design for the Portal (controlled by themes and skins), import the portlets to be used on the page and then drag and drop them on the portal page. A special palette of portal controls and a Thumbnails view of themes and skins make it easy to control the page design. Rational Developer ships with several predefined themes and skins that allow developers to quickly create a new portal. Figure 5 shows the Portal Designer:

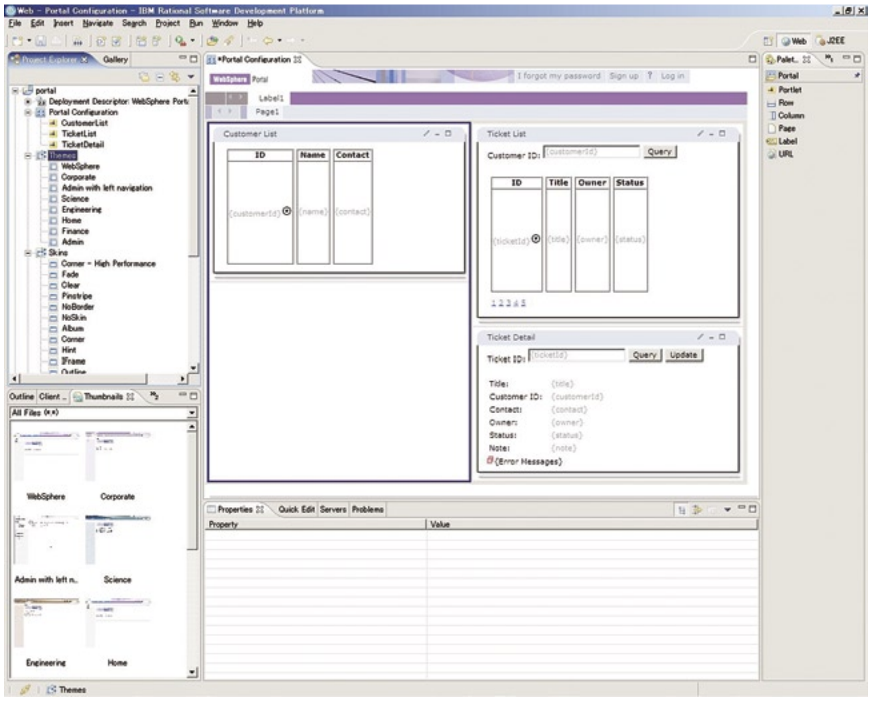


Figure 5. The Portal Designer helps developers lay out a Portal interface without writing code

Rational Developer also saves time with the new capability to visually edit themes and skins with the familiar Page Designer tool, helping developers quickly tailor the design of their portal and the portlets. The Style Editor provides fast access to properties that control the visual page elements,

Finally, the Portal Test Environment lets developers quickly deploy their entire Portal project and test the functionality of portals and portlets on an instance of WebSphere Portal Server running within Rational Developer.

Rapidly develop Web services

Web services are modular, standards-based applications that can be dynamically mixed and matched to perform complex transactions with minimal programming. Web services let buyers and sellers discover each other, connect via the Web and execute transactions in real time with minimal human interaction. Web services are currently the most common example of a service-oriented architecture that is gaining popularity across application development. With J2EE 1.4, Web services are now considered core J2EE components.

Rational Developer makes it easy to discover and use existing Web services or create new ones without writing any code. Developers can also create Web services from existing artifacts, such as JavaBeans, stored procedures, EJB components or SQL queries. The Web services tools are based upon the open, cross-platform standards of the Universal Description, Discovery and Integration (UDDI), Simple Object Access Protocol (SOAP) and Web Services Description Language protocols. These protocols ensure the interoperability of Web services with other programs, while Rational Developer handles all of the programming details. Developers do not have to learn any new languages or protocols to create Web services with Rational Developer—they simply code their objects as usual and let Rational Developer package them as Web services.

As with data access, Rational Developer provides both a top-down and bottom-up approach to developing Web services. The bottom-up approach is the easiest way to create Web services, because it uses existing programs and turns them into Web services. However, the top-down approach is recommended when developing entirely new services because programs that are designed from the outset as Web services offer a higher degree of interoperability. Rational Developer provides tools to automate each method.

Discovering and using Web services

Rational Developer makes it very easy to find existing Web services and use them in Web applications without additional programming. UDDI registries are online listings of Web services that provide technical specifications and company descriptions. The UDDI Explorer tool makes it easy to browse these registries, find Web services and import them into a project. Web designers can use Page Designer to drag and drop Web services from the Page Data view. Any Rational Developer artifact (for example, Faces applications, Struts applications, portlets, servlets and EJB components) can call a Web service as if it were a local entity. Rational Developer even renders the controls on the JSP pages that are needed to invoke the Web service operations and display the information returned by the Web service. It also supports authentication protocols to let developers sign on to protected remote servers to access Web services.

Without Web services support, developers would have to manually discover the Web service, add a UDDI client to the application, manually code the SOAP proxy and hand-code the JSP that uses the Web service. Rational Developer automates all of the tasks needed to integrate a Web service, saving an incredible amount of developer time and effort.

When testing the application, Rational Developer automatically deploys the link to the remote server, configures a Web service proxy and compiles the Web service with the application. Tests can be automatically generated based on the Web service's description and used for regression testing. When deploying the application to WebSphere Application Server, the application package contains everything needed to run the Web service.

Top-down Web services development
The top-down approach consists of first developing a WSDL file that contains an XML schema to describe the Web service and then generating a skeleton JavaBean or EJB files. Rational Developer automates the creation of WSDL files by providing a skeleton WSDL that developers can modify for their Web services. A graphical editor that lays out the elements of the WSDL file and a hierarchy makes it easy to edit the WSDL file. Many features of this editor accelerate WSDL file editing—for example, the editor automatically renames all associated sub-objects when an object is renamed. It also contains an XML schema definition (XSD) editor for specifying the format of messages. Figure 6 shows the WSDL editor.

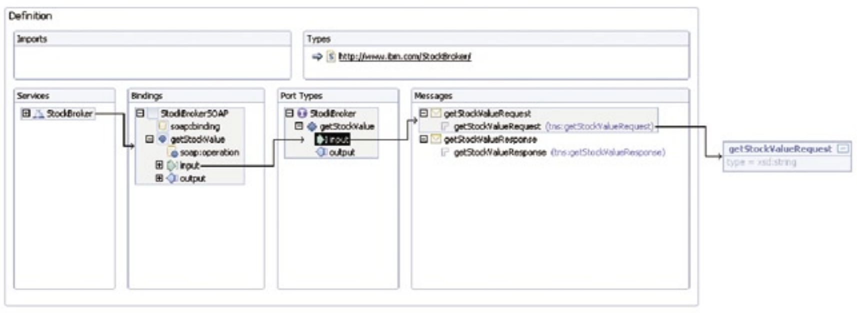


Figure 6. The WSDL Editor facilitates Web services development

When the WSDL file is complete, Rational Developer generates a skeleton for either a JavaBean or EJB component. With this "plumbing" code generated, the developer then writes the business logic code the same as for any other JavaBean or EJB component and debugs it in the Web Services Explorer. Finally, Rational Developer provides a Web services client wizard that generates a complete JSP-based sample with proxy code to test the Web service.

Rational Developer ships with a UDDI Version 2–compliant registry for Web services. This Unit Test UDDI helps developers test UDDI-related aspects of the development cycle, such as publishing their Web service to a registry. They can even test for Web Services Interoperability Organization (WS-I) conformance (which checks for interoperability with applications running on other WS-I–compliant platforms) and automatically configure a TCP/IP monitor session to monitor the execution of the Web services over the network.

To create a top-down Web service manually is an incredible amount of work. The WSDL file alone would require coding hundreds of lines of XML in addition to the wrapper JavaBean or EJB components and creating a test client and proxy. Again, Rational Developer lets the developer concentrate on business logic rather than the details of administration.

Bottom-up Web services development
When turning an existing Java class or EJB component into a Web service, developers can use a simple wizard to specify information about the component. Rational Developer generates the WSDL files describing the Web service, generates an SOAP deployment descriptor and the proxy that a client can use to access the Web service. Rational Developer also provides wizards that automate the creation and deployment of handlers, which are programs that sit between the Web service and the client machine that can transform data into special formats.

Web services are deployed to the WebSphere Test Environment, where testing is performed using the same interface as for any other program. It even creates a sample test client that can be used to test the functions of the Web service. The WebSphere Rapid Deploy feature shortens the test cycle, eliminating the need to restart WebSphere Application Server when deploying changes to the Web service.

When it is time to publish the Web service, Rational Developer makes it easy to add the Web service to a public UDDI registry for other users to find and use or automatically generate a private registry to publish the Web service to a select group of users. Java and J2EE developers do not have to do any additional coding to offer their application as a Web service, so there is no learning curve for Web service development and no resulting decrease in productivity.

Additional productivity tools

Rational Developer provides numerous additional tools to enhance developer productivity. This section covers just a portion of the noteworthy features that can accelerate the development process.

Performance analysis

One of the biggest headaches of application development is finalizing a major project, testing its performance as part of system test and then having to re-architect the solution to address performance issues. Rational Developer provides several tools that let developers test an application's performance early in the development cycle. This early detection of performance issues can save crucial development time, because it allows time for re-architecting the design before proceeding too far down the implementation path.

The Profiling view provides tools that collect data about a program's runtime behavior and presents the information in graphical and statistical views that allow for easy analysis. These views expose memory leaks and help illustrate patterns in an application's design. Developers can also launch remote applications and concurrently monitor multiple live processes that might reside on several other machines. The Profiling view can gather information on a stand-alone Java application or an application running within a J2EE-based application server such as WebSphere Application Server. In both cases, profiling can be performed either locally or remotely relative to the application's target host. Furthermore, profiling can occur across multiple Java Virtual Machines.

For Web projects, developers can also analyze performance using the TCP/IP Monitoring Server, which displays requests and responses as they occur. This tool is especially useful for monitoring the performance of Web services, because it lets you view the interactions with the service running on a remote machine.

Team programming

Rational Developer now provides a fully integrated version of Rational ClearCase LT, which is a reliable, entry-level version control tool designed for small project workgroups. Rational ClearCase LT is part of the IBM Rational ClearCase product family that can scale from small project workgroups to the distributed, global enterprise.

Rational ClearCase LT provides support for parallel development. With automatic branching and snapshot views, it helps multiple developers efficiently design, code, test and enhance software from a common code base. Snapshot views support a disconnected use model for working away from the office and automatically update all changes made since the last snapshot. Developers can disconnect from the client without restarting Rational Developer and can configure workspace check-in/check-out on reconnect. Developers can create their own local views to configure ClearCase to meet their individual requirements.

The diff/merge technology in Rational ClearCase makes it practical to merge source code, HTML and XML. It automatically accepts uncontested changes and highlights conflicting ones for fast resolution without manual intervention. Rational Developer also provides the Eclipse diff/merge patterns, which enable structural comparison of files stored in ClearClase.

ClearCase LT can greatly increase the productivity of teams, letting them easily share code and manage large projects. If your team begins to require enterprise-level functions (such as distributed servers, database replication or advanced build management), you can seamlessly upgrade to Rational ClearCase without retraining your team—or changing your process, data or tools.

Externally available toolkits
To help enable integration with server environments from IBM and other vendors, IBM provides several freely available toolkits to download and install for use with Rational Developer. Some popular toolkits include:

- ***Everyplace Toolkit for WebSphere Studio****—extends the development environment to help enable developers to design, develop and implement portlet applications on mobile devices. Templates help them quickly and easily create mobile portlets and applications. This toolkit provides additional remote test facilities to help test applications before deployment. In addition, it contains a preview of the Multi-Device Authoring Technology tool set, which helps enable the development of Web and portlet applications targeted to multiple devices with different characteristics. It is available on the Web at: ibm.com/software/pervasive/everyplace_toolkit/*

- ***Lotus Domino Toolkit for WebSphere Studio****—easily program applications that connect to Domino® applications and work with Domino data. This toolkit provides the Domino Version 6 custom JSPs. From within Rational Developer, you can view a Domino application residing on your server and work with forms, views and data. You can use those elements in a new JavaServer Page or in existing transactional applications. It is available on the Web at: lotus.com/products/product4.nsf/wdocs/toolkitwsstudio*

- ***IBM Rational Deployment Toolkit for WebLogic Server****–provides tools for working with BEA WebLogic Servers and files in Rational Developer. It supports WebLogic Server version 6.1, 7.0 and 8.1. The toolkit lets developers develop, test and deploy J2EE applications using the BEA WebLogic Server with Rational Developer. The WebLogic Server can be installed on the same machine as Rational Developer or on a remote machine, and developers can launch the WebLogic Server console from within Rational Developer as an embedded Web application. It is available on the Web at: ibm.com/software/info1/websphere/index.jsp?tab=products/logictoolkits*

- ***Eclipse plug-ins****–hundreds of additional plug-ins for Eclipse-based IDEs are available through the Eclipse open source community. Open source and commercial tools are available through several online registries. Check out Eclipse Plug-in Central (www.eclipseplugincentral.com) for a comprehensive list and the eclipse.org community page for other online registries. The IBM developerWorks® site also provides a listing of IBM-validated plug-ins at WebSphere Studio Plug-in Central: ibm.com/developerworks/websphere/downloads/plugin/index.html*

Building Rational Developer skills

IBM offers a variety of methods to help your developers quickly get up to speed on Rational Developer, ranging from online self-help and education to on-site consulting services and education.

Integrated learning tools

Rational Developer comes with a variety of skill-building tools that are integrated with the development environment. The Tutorial and Sample galleries provide detailed training on using product features and let you customize complete programs for your individual needs. The product tours and mini-videos introduce the interface to new users and the context-sensitive help, show-me guides and cheat sheets help developers navigate the user interface.

The IBM Rational Unified Process® (RUP®) platform is a configurable software development process platform that delivers proven best practices and a configurable architecture. It gathers knowledge gleaned from thousands of projects worldwide and encourages iterative development, which helps teams address risk early and continuously throughout the development process. Rational Developer provides context-sensitive access to the RUP, which gives practical process guidance at every stage of development.

Community support

For online community support, visit developerWorks Rational Developer Zone, which lets you access all online technical resources from one place. Here you can find migration help, technical articles and tutorials on all the great Rational Developer features and scenarios on how to use Rational Developer with other tools. You can also access online forums to trade tips and techniques with other Rational users and find a local Rational User Group in your area.

Education and training

IBM provides a wealth of course materials on using Rational Developer. At the developerWorks Rational Developer Zone, you can find tutorials, distance learning classes with voiceover and animation, and information on classroom courses in your area. For training on the J2EE programming model and other new technologies, visit the developerWorks technology areas, such as the Java and Web Services Zones. IBM Learning Services provides a wide range of educational packages for your company.

Consulting services

If you have a critical application that needs to get deployed very quickly, you might consider calling in the big guns—the IBM Software Services for WebSphere team. They offer a wide range of consulting packages, from on-site education packages to intense mentoring programs. For more information, visit the developerWorks WebSphere Services page or contact your sales representative.

Try it out

Rational Developer can greatly decrease the time-to-market for your Web, portal and J2EE applications. The easy-to-use tools, combined with the customized, task-oriented perspectives, provide a development experience that makes it easy for novice developers, procedural developers and expert J2EE developers alike to quickly create and deploy e-business applications.

But don't take our word for it. Visit the Rational Developer Trial Program page and download trials of Rational Web Developer or Rational Application Developer to start exploring this rich development environment today by visiting:

ibm.com/developerworks/websphere/downloads/WSsupport.html

Resources

developerWorks subscription: A starter level subscription gives you low-cost access to Rational Web Developer and even more IBM software: ibm.com/developerworks/subscription/

Rational Developer trial program: ibm.com/developerworks/websphere/downloads/WSsupport.html

Eclipse open source site: www.eclipse.org

developerWorks WebSphere: ibm.com/developerworks/websphere/

WebSphere Studio Plug-in Central: ibm.com/developerworks/websphere/downloads/plugin/index.html

J2EE application profiling in WebSphere Studio: ibm.com/developerworks/websphere/library/techarticles/0311_manji/manji1.html

Using Ant with WebSphere Studio Application Developer: ibm.com/developerworks/websphere/library/techarticles/0203_searle/searle1.html

Design service-oriented architecture frameworks with J2EE technology: ibm.com/developerworks/webservices/library/ws-designsoa/

Introduction to Lotus Domino Toolkit for WebSphere Studio: ibm.com/developerworks/websphere/library/techarticles/0304_schumacher/schumacher.html

IBM WebSphere Developer Technical Journal: Developing JSF Applications using WebSphere Studio V5.1.1: ibm.com/developerworks/websphere/techjournal/0401_barcia/barcia.html

Web Services development and deployment with WebSphere V5 Tools and Technologies: ibm.com/developerworks/websphere/library/techarticles/0302_flurry/flurry1.html

Developing and debugging portlets using the IBM Portal Toolkit Plug-in for WebSphere Studio Application Developer: ibm.com/developerworks/websphere/library/techarticles/0210_phillips/phillips.html

IBM WebSphere Developer Technical Journal: Writing a Simple Struts Application using WebSphere Studio V5. ibm.com/developerworks/websphere/techjournal/0302_fung/fung.html

IBM WebSphere Developer Technical Journal: Using EGL and Struts with WebSphere Studio Enterprise Developer Version 5. ibm.com/developerworks/websphere/techjournal/0304_barosa/barosa.html

About the Author

Stephanie Parkin currently works as an information architect on the IBM developerWorks WebSphere Web site. She has co-authored two books on visual programming: *Building Applications with WebSphere Studio* and *JavaBeans and VisualAge for Java for Non-Programmers*. She is currently writing a book on developing Web services with WebSphere Studio.

G325-1122-01