WebSphere® software

IBM

Rapid Java and J2EE Development with IBM WebSphere Studio

By Stephanie Parkin
IBM Software Group

e® business

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 developing application development solutions to 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 J2EE solutions. IDEs that contain rapid application development tools automate many time-consuming development tasks. Organizations look for IDEs that are easy to learn and maximize the productivity of developers with diverse backgrounds.

This paper describes how the IBM WebSphere® Studio products speed up both the adoption of Java and J2EE technologies for IT shops new to the technology, and the development process for shops already well versed in Java technologies.

The importance of Java and J2EE technologies

Odds are, if you've picked up this paper you're already convinced of the importance of the Java platform. For those who may still be uncertain, let's review its basic benefits. If you're interested in more details on the history of the platform, see the developerWorks® Java zone, listed in the Resources section at the end of this paper.

Multiplatform:

The real benefits of Java technology lie in its deployment architecture. Since the Java platform relies on the unique concept of a single Java virtual machine (JVM), all Java programs can run on any system that has a version of the JVM. This JVM provides unparalleled portability across platforms and operating systems. Furthermore, existing Java applications can easily be adapted for devices with limited-memory resources. In essence, the Java platform extends a user's computing power from the desktop to the resources of the Web. Java technology can help you bridge your different computing environments, and to 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. Using a language based on open standard ensures you won't get locked into a proprietary environment that might suddenly go in a direction that doesn't suit your company's needs.

Object-oriented:

From a programmingent point of view, the Java language is object-oriented (OO), which means that it models real-world objects, and describes their properties and their interactions or 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.

Java 2 Enterprise Edition (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 provides 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 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. In this section we'll discuss each of these types of users in detail, and describe what challenges each of these user sets face when developing Java applications.

Challenges for novice Java developers

Many companies are experiencing a shortage of skilled Java and J2EE developers, and they don't come cheap. Most companies need their existing employees to quickly learn Java skills, and a large number of developers trained in client/server programming technologies such as VisualBasic are scrambling to get up to speed on the Java language, and especially J2EE technology.

VisualBasic was widely and quickly adopted by masses of developers largely thanks to the simplicity and ease of use offered by 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 VisualBasic developers.

WebSphere Studio provides the ideal environment for novice Java developers, especially visual developers. As we'll explore in this paper, WebSphere Studio's new JavaServer Faces technology, coupled with its visual editor and diagram editors, and a data access framework based on Service Data Obejcts, make it easy to develop Web applications without writing code.

WebSphere Studio uses perspectives to tailor the user interface to the task of the developer, so while it contains a wealth of features for expert developers, it effectively hides the complexity from less experienced users. The wizards, code assist features and interactive guides (called cheat sheets) all guide new users through the application development process. WebSphere Studio is the ideal tool to get your staff up to speed on Java technology.

Challenges for legacy procedural developers

Many companies procedural programmers who possess a wealth of knowledge about the company's business, and supporting legacy systems and databases, but who lack the OO skills required to develop new e-business solutions. These developers are proficient in COBOL, RPG, C, or other procedural languages, and they might also be mainframe developers well versed in subsystems like CICS®, or IMS. With their in-depth knowledge of existing systems and the company's business processes, their expertise is critical for the modernization of legacy systems. However these programmers often are resistant to, or have trouble adapting to OO programming techniques.

Fourth generation (4GL) programming languages bridge the gap between procedural programming and object-oriented programming. They let developers use a programming paradigm that's comfortable for them, 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 WebSphere. It's a simple procedural language that any programmer proficient in business-oriented languages would find easy to pick up. 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.

For companies that need all the benefits of Java technology, but have a programming team trained on procedural languages, WebSphere Studio's EGL language can unlock the power of these developers in delivering innovative e-business systems without having 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 the 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 this set of developers.

For expert J2EE developers, WebSphere Studio brings together tools for rapidly developing and deploying all the various components that comprise J2EE applications: JavaServer Pages (JSPs), servlets, Enterprise JavaBean (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, to dynamic application performance analysis.

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

IBM WebSphere Studio 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 WebSphere Studio unique in the world of Java IDEs.

Part of 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 lifecycle. WebSphere Studio fits into a tools framework that supports structured application development, including modeling, proven design practices and patterns, and an iterative development process that ensures 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 freedom of choice in a multi-language, multi-platform, multi-vendor supported environment. Eclipse delivers a plug-in based framework that makes it easy to create, integrate and use software tools together.

WebSphere Studio is IBM's premier application development offering based on Eclipse. It provides a comprehensive and extremely productive application development environment for creating and maintaining J2EE-compliant enterprise application systems and includes many features not available in Eclipse. And because WebSphere Studio 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 to maximize developer productivity.

While the Eclipse platform does provide an open-source IDE that can be used to code applications, it's really a bare-bones platform upon which you build application development tools. Table 1 gives a quick overview of a few of the features provided with WebSphere Studio that you don't get with the Eclipse IDE:

Feature	Usage
Page Designer	Visually laying out and designing dynamic HTML and JSP pages
Web Site Designer	Managing the structure and navigation of entire Web sites
JavaServer Faces support	Quickly developing rich Web user interfaces
Service Data Objects support	Quickly developing data-driven Web application
Web Diagram Editor	Visually mapping and constructing Struts-based Web applications
Enterprise Generation Language	Generating Web applications without coding in Java
Portal toolkit	Visually developing portlets using JSF and Struts
Integrated Web services tools and wizards	Discovering and using existing Web services and building, testing and publishing new Web services
Visual Editor for Java	Building Java application user interfaces using AWT and Swing components
J2EE and EJB wizards and editors	Developing, testing and deploying J2EE apps and EJB components
Integrated WebSphere Application Server and WebSphere Portal unit test environments	Unit testing J2EE and portlet applications
Performance profiling tools	Identifying and fixing performance problems
Component Test tools	Building and managing test cases for unit and system testing
XML tools	Creating, editing and transforming XML documents
Relational Schema Center	Managing and accessing databases
Integrated UML Visual Editor for Java and EJB	Visualizing and managing complex code
Rational ClearCase LT	Adding version control and team support

Table 1. What Does WebSphere Studio add to Eclipse?

Provides a complete family of tools

Various configurations of WebSphere Studio ensure that the tool 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 main WebSphere Studio configurations provide more functionality as you move up the ladder; for example WebSphere Studio Application Developer contains all of the functionality of Site Developer.

- *WebSphere Studio Site Developer*
  - is an entry level IDE for Web developers and Java developers, 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.

- *WebSphere Studio Application Developer*
  - allows for more advanced J2EE development, including EJB components. It also supports portlet and UML-based development, and contains Rational® ClearCase® LT for version control. Another flavor of Application Developer is the WebSphere Studio Application Developer Integration Edition, which enables accelerated development and integration of composite business applications that deploy to the IBM WebSphere Business Integration Server Foundation. In addition to all of the features of WebSphere Studio Application Developer, 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 developing applications that target legacy back-end systems such as CICS and zSeries. It also provides EGL code generation that outputs COBOL source code.

WebSphere Studio Site Developer is also provided in WebSphere Application Server – Express. WebSphere Application Server - Express provides an entry-level application server for lightweight Web applications.

This paper focuses on WebSphere Studio's rapid application development features, so it will be primarily describing WebSphere Studio Site Developer and WebSphere Studio Application Developer V5.1.2.

IBM WebSphere Studio accelerates Java and J2EE development

Now let's dig into the details of how WebSphere Studio accelerates Java and J2EE development. Two basic WebSphere Studio design principles decrease the Java learning curve and increase programmer productivity:

- *Hides complexity from novice users*
  WebSphere Studio, using a series of perspectives, organizes the user interaction with the various tools based on their role in the development team or project, or the task at hand. All the perspectives share a common look and feel which reduces learning curves and helps developers organize complex projects and focus on the task at hand. However, they are particularly beneficial to novice Java developers, who might only need to access a subset of the wealth of capabilities available in the rich WebSphere Studio environment. Advanced users can also customize their perspectives to show the tools they use most often, or to include other plug-in tools not included with the base product.

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

Rapidly develop Web applications

WebSphere Studio provides many features to speed up the development of interactive, forms-driven Web applications. Companies that are creating lightweight, dynamic Web applications need a tool that, for example, lets them provide marketing information online, and some basic forms for gathering registrations and perhaps providing 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 covering all these roles).

WebSphere Studio supports all of these roles in the Web perspective. The Web master can 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 layout dynamic Web pages as JavaServer Pages (JSPs), and connect to existing databases, again without writing code. If the application will run on a portal server, Web developers can also create portlets to improve usability and to allow end users to customize their interface. 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 their applications.

Build dynamic Web pages without coding

Page Designer lets Web designers quickly layout a Web page, with an interface similar to other WYSIWYG Web editors. It provides tabs for viewing the page in Design mode (WYSIWYG composition), Source mode (code-level composition), and Preview (view that page as it will appear in the browser at runtime)

Designers can quickly add images, links, and forms to the page simply by 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.

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 don't need to look up the syntax.) WebSphere Studio also provides context-sensitive help for the current tag being edited.

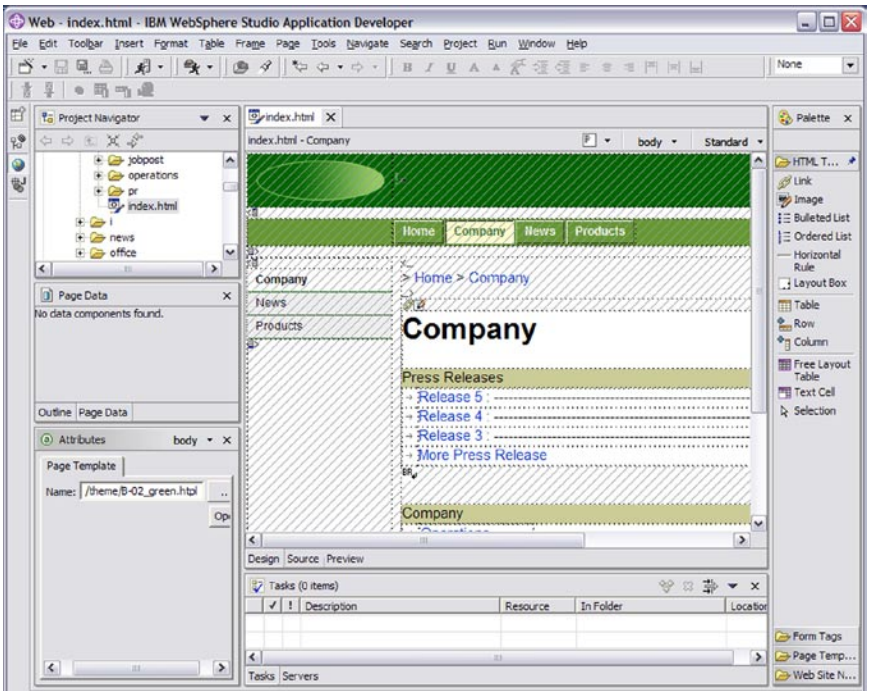


Figure 1. Page Designer

To quickly design a new Web page, designers can start with one of the many example Web sites shipped with WebSphere Studio, and modify as needed. They can also use predefined Page Templates. Figure 1 shows the Page Designer tool with a Company Web page template open.

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

"The new JavaServer Faces tools in WebSphere Studio provide an easy way to rapidly develop user interfaces for J2EE applications and to integrate relational databases,"

..."WebSphere Studio eliminates a lot of the hand-coding we used to do, and now makes our Lotus® workplace development faster and more efficient."

– Dr. Rolf Kremer,
Chief Development Officer,
Pavone AG.
Germany

Rapidly develop interactive Web pages with JavaServer Faces

Java Server Faces (JSF) 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, validating and converting strings. The JSF framework consists of UI components, and a run time that renders these components to the client and manages the page lifecycle (errors, validators, navigation, etc.). Using the visual JSF tools in WebSphere Studio, you can create very ich, dynamic Web sites without writing a single line of code.

WebSphere Studio provides intuitive wizards and visual UI components for creating JSF files. The wizards do all the set-up work needed to use JSF in an application. WebSphere Studio 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.

Obviously 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. WebSphere Studio creates code-behind files that contain the page source code. The source is dynamically generated as you edit the JSF design surface – there's no additional generate step, so the code and UI design are always in lock step. Another time-saving component is the Service Data Object, which deserves its own section.

Quickly add dynamic data to your Web pages

WebSphere Studio 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 Java bean, 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 or Web service. Developers do not have to write the JDBC-access code; the Service Data object does this for them.

WebSphere Studio makes it easy to build data-driven Web applications. Using JavaServer Faces components, and Service Data Objects, a simple CRUD (create, rename, 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 WebSphere Studio generate the UI). They can work with a tabular view of several rows 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, and even handle paging through data (for example showing 5 rows at a time). The data column can contain images, hyperlinks, etc. The generated code ensures 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 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

|< < Page 1 of 7 > >|

Figure 2. Service Data Object - Data Grid

Easily create, import and manage Web sites

The Web Site Designer tool helps developers create and manage the organization and layout of the Web 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 navigational relationships. From this view developers add, delete, and move pages in the Web site.

The navigation information is used to automatically generate navigation menus, links, and site maps. Navigation links are automatically regenerated to reflect changes in the site's structure. The resulting menus, links and maps are standard HTML and so can be deployed to any HTTP server. WebSphere Studio also lets developers import existing Web sites, and redesign them with Web Site Designer.

Without this capability, developers would have to create each of the navigation elements by hand, using a Web page editing tool. They would have to manually create page templates, and couldn't easily enforce their use. When the site structure changes, they would have to either manually update the navigation elements on each page, or manually code the XML file that manages the structure.

The Web Site Designer and Navigation view are enormous time savers for managing any size Web site. They automate the tasks of ensuring consistent page layout and navigation across the site. Figure 3 shows the Navigation view in Web Site Designer.

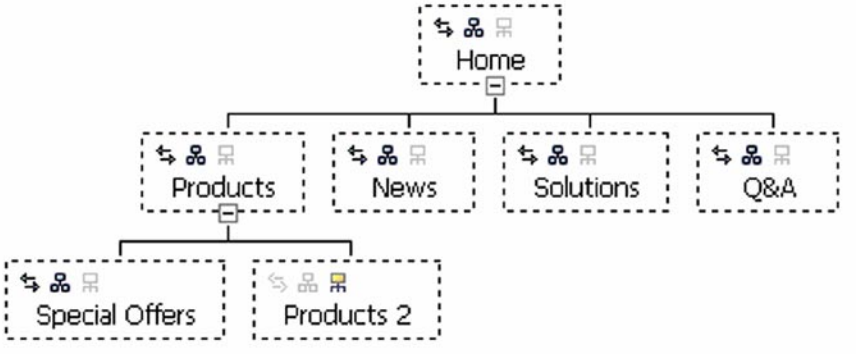


Figure 3. Web Site Navigation view

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. WebSphere Studio's Struts tools let novice developers create well-architected Web applications, without coding the layout and connecting code.

WebSphere Studio's Struts tools include a Struts Explorer, and a Web Diagram Editor. The Struts Explorer helps developers understand and navigate through the various pieces of a Struts application. When a new Struts project is created, WebSphere Studio builds a Web diagram that shows the structure of the application. The Web Diagram Editor provides an intuitive facility for laying out the Web pages (JSPs), business logic (Actions) and data that passes through the application (FormBeans). The developer then draws connections that indicate how the application flow proceeds. As the diagram is edited, WebSphere Studio updates the Struts XML configuration files and generates Java code.

For example, Figure 4 shows a Web Diagram for a simple Hello World application. Developers can click on any of these objects to add business logic.

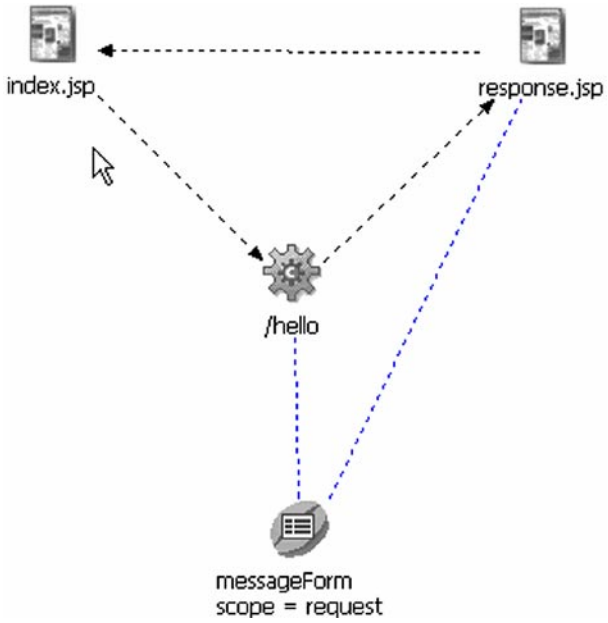


Figure 4. Web Diagram Editor

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 WebSphere Studio and encapsulated in reusable beans.

Without this feature, your 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've separated business logic from interface elements.

Generate Java and Web applications with EGL

The Enterprise Generation Language (EGL) provides a "fourth generation language", RAD environment for developing both Web and traditional character-based applications. EGL evolved from IBM VisualAge Generator, to enable moving applications to the Web. WebSphere Studio extends EGL by incorporating 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, RPG, and other 4GL programmers. Using the EGL language in WebSphere Studio, your procedural developers can rapidly develop Web and J2EE applications using familiar programming constructs.

EGL is a fully supported language in WebSphere Studio, enabling procedural developers to become immediately productive in J2EE projects. For example, EGL is integrated into the new JavaServer Faces (JSF) 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.

WebSphere Studio 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" type events), which enables more intuitive controller logic. This event helps procedural programmers map their traditional text-based applications to more GUI-driven designs.

"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 (EGL),"

– Patrick van Dorsselaer,
Senior 4GL Developer
Informa
Netherlands

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

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

WebSphere Studio Site Developer and Application Developer can generate Java code to deploy to Windows, AIX®, Linux, and iSeries. Enterprise Developer can also generate COBOL code to deploy to iSeries, and zSeries® (CICS and batch). EGL also provides connectivity to IBM Informix® Dynamic Server, to DB2, and to other relational database servers through JDBC.

Quickly debug and test Web applications

Because developers spend so much time debugging applications, WebSphere Studio 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 can either fix broken links or globally convert links to a new root when an application's directory structure changes. The Source code views also provide visual cues to identify possible problems.

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

WebSphere Studio 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 into a separate server, and view the application running exactly as it would in the target deployment environment.

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

WebSphere Studio 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 either be a physically separate machine, or another server instance on the same machine running outside of WebSphere Studio. 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 to WebSphere Application Server, Tomcat, or BEA WebLogic. New JRE definitions can also be added to WebSphere Studio, and can target different JDK 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, WebSphere Studio provides many rapid application development tools for programmers involved in creating more complex Java applications that require more Java programming knowledge, and where most of the logic runs 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 test them on servers. They need tools to help them 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

WebSphere Studio provides Java editors, including a visual editor, and a traditional (text-based) one. The WebSphere Studio Visual Editor for Java is based on the JavaBeans components model, which defines the interface to a Java bean through methods, properties, and events. This editor lets developers visually design the GUI components for an application, and then implement the necessary Java beans to add application functionality.

Unlike other visual editors, the Visual Editor for Java performs round-tripping of changes between the Java beans and the source code. Developers can modify the Java beans and see the modifications reflected in the source, and can also change the source code and see the changes applied to the Java beans. The Visual Editor for Java is a code-centric editor that not only supports round-tripping of changes, but also displays the Java beans and the source code side by side using a split pane. When a Java bean is selected, the source code shows the method that initializes it, and 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 Java beans from Swing or AWT component libraries, or imported from other libraries.

Speed up coding of Java and J2EE applications

The text-based WebSphere Studio Java editor provides several features that accelerate Java development. The editor gives visual aides such as syntax highlighting, and code assist for completing the current language element.

WebSphere Studio provides automatic incremental compilation, which means that the code editor can give immediate feedback about problems in code, helping eliminate many bugs before compiling the program.

Another big time saver is the ability to reorganize code without having to manually fix all the references to specific components. Refactoring code means changing the code while preserving the behavior. The refactoring tools enable

reorganization of program elements from any editor, and even renaming elements. WebSphere Studio will dynamically update all references to the code, and give a preview of the impact of the changes before making them permanent. This saves developers a huge amount of tedious work locating and changing references in code.

For EJB development, WebSphere Studio 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 generate wrappers for most code artifacts.

The Java editor recognizes EJB components, and offers several extensions such as a task list that provides immediate feedback on the correctness of the EJB components and other J2EE artifacts. The code editor verifies that different types of enterprise beans are constructed correctly and that they are consistent with other associated enterprise beans.

WebSphere Studio also provides special editors for viewing and modifying EJB deployment descriptors, and tools to create, edit and validate Enterprise Archive (EAR) files. It can generate cross-EAR EJB references, and generate EJB components that tie into transaction processing systems. Developers can quickly turn their EJB components into Web services (more on that later). All of these EJB features work together to provide a J2EE structure for developers, so that they can concentrate on coding the 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 (more on that later too).

Rapidly access data

One of the biggest challenges for developers has always been integrating their applications with back-end data systems. Usually developers need to access existing data that has a predefined structure that they need to bring into their programs. Occasionally developers that aren't familiar with data modeling must develop the data structures themselves. WebSphere Studio provides tools that automate much of the coding for both of these scenarios, by importing a data structure from an existing data source, or creating a new data model within WebSphere Studio, and exporting it to the actual data source. It also provides a Data perspective 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.

Automatically map data to EJB components

WebSphere Studio provides wizards that help quickly create certain kinds of EJB components. 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 WebSphere Studio import the target data structure, and handle all the mappings, which takes most of the coding out of this sort of EJB development.

WebSphere Studio also provides wizards for top-down mapping for EJB development, for both container-managed persistence (CMP) beans and bean-managed persistence (BMP) beans. After defining the structure of the data, developers can map it to the data source using a simple map browser that shows the Enterprise Beans and the database tables side by side. They can drag and drop entities from the Enterprise Bean table onto the database tables, and WebSphere Studio 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 5. shows the Map Browser

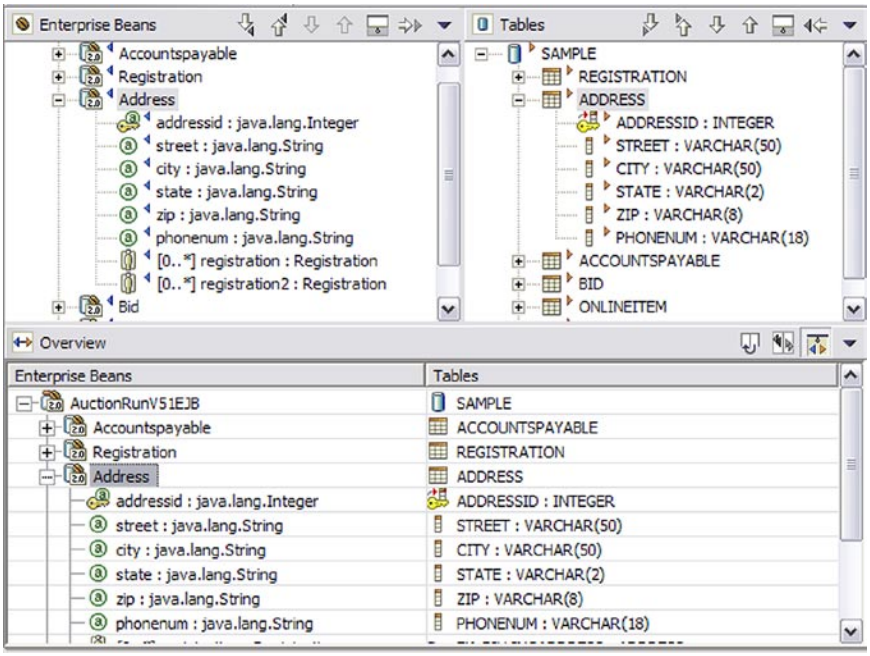


Figure 5. Map Browser

Visually build SQL queries

The SQL query builder provides a visual interface for creating and executing SQL statements, from very simple statements to complex expressions and groupings. It provides an SQL to XML wizard to generate XML artifacts for se 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 retrieve. They 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 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. WebSphere Studio also provides 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 the standard WebSphere Studio debugging tools, including running on a server without going through the cumbersome steps of packaging and deploying the stored procedures.

Visualize and edit code with UML

The UML Visual Editor for Java and EJB tool provided with WebSphere Studio Application Developer gives a graphical view of the code. It displays code artifacts as Unified Modeling Language (UML) class diagrams, which shows the structure and relationships of Java classes, interfaces, and EJB components. UML notation is an industry standard for object modeling, ratified by the Object Management Group, Inc. 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. Figure 6 shows a simple UML class diagram, and the relationships between the classes. However the UML Visual Editor is not just a static view of the code. This interface can be used to actually implement the objects in a program. The editor is tightly integrated with the entire WebSphere Studio development environment, so that developers can implement classes and EJB components directly from this editor, and see the relationships between classes illustrated as they code them.

For example, the UML Visual Editor provides powerful graphical editing capabilities for EJB components. The object model is saved along with the rest of the J2EE project. 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 also an easy way to refactor code; as objects are dragged and dropped in the model, the references are automatically updated in the underlying code.

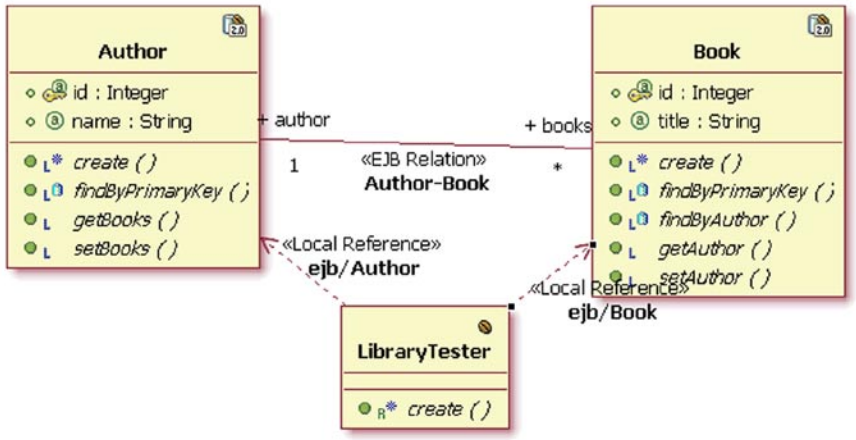


Figure 6. UML Visual Editor

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. Often this task has to be performed frequently, sometimes weekly or even daily. WebSphere Studio allows for 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 WebSphere Studio. 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 with debugging of Ant scripts.

Quickly test J2EE applications

In addition to the unit test environment described above, WebSphere Studio provides testing features that accelerate the test cycle for J2EE applications. When unit testing parts of a J2EE application, such as EJB components, WebSphere Studio can automatically configure the server to access data sources and generate EJB deployment code. Developers can even modify the application on the fly, and 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 deploying changes.

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 WebSphere Studio or an external one. This client lets developers set breakpoints, define parameters to pass to beans, 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, your developers would have to manually create a test client that covers all test cases for an application, and deploy your test client to the target server.

WebSphere Studio also provides component test tools for building and managing test cases for unit and system testing, The Component Test perspective lets testers manage manual test cases and automate testing of Web sites and Java applications, running locally or remotely. Testers also can track test results, generate reports, and create custom report generators. Without this feature, the test team would need to manage test cases manually, and manually code programs to call the components that require testing.

Rapidly develop portlets

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. WebSphere Studio Application Developer includes the IBM Portal Toolkit, which enables developers to create, edit and test portlets just as they would any other type of programming artifact. For example, they can manage complex portlet projects that render portlet JSPs, locate and use Web services, and manage data. WebSphere Studio Application Developer also includes a Portal Test environment for unit testing a portlet's behavior, and system testing how it interacts with other programming artifacts.

Generate portlets with wizards

As with other types of applications, developers create a new portlet using WebSphere Studio wizards, which generate a portlet project structure that conforms to J2EE standards, and creates a complete portlet. WebSphere Studio 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 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 layout 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 within portlets 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 don't need to "know" about each other to send information back and forth. WebSphere Studio provides a drag and drop component for implementing Click-to-Action without programming, and it shows an icon to visually identify portlets with Click-to-Action enabled.

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 (SOA) that's gaining popularity across application development.

WebSphere Studio 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 Java beans, stored procedures, EJB components or SQL queries. The Web services tools are based upon the open, cross-platform standards of Universal Description, Discovery and Integration (UDDI), Simple Object Access Protocol (SOAP), and Web Services Description Language (WSDL). These protocols ensure the interoperability of Web services with other programs, but again WebSphere Studio handles all the programming details. Developers don't have to learn any new languages or protocols to create Web services, with WebSphere Studio they simply code their objects as usual and let WebSphere Studio package them as Web services.

As with data access, WebSphere Studio provides both a top-down and a 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. WebSphere Studio provides tools to automate each method.

Discovering and using Web services

WebSphere Studio makes it very easy to find existing Web services and use them in Web applications, without programming. UDDI registries are online listings of Web services that provide technical specifications and company descriptions. The WebSphere 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 WebSphere Studio artifact can call a Web service as if it were a local entity (for example Faces applications, Struts applications, portlets, servlets, and EJB components). WebSphere Studio even renders the controls on the JSP pages that are needed to invoke the Web service operations and to display the information returned by the Web service.

Without this RAD 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. WebSphere Studio automates all the tasks needed to consume a Web service, saving an incredible amount of developer time and effort.

When testing the application, WebSphere Studio automatically deploys the link to the remote server, configures a Web service proxy, and compiles the Web service with the application. 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 Java bean or EJB files. WebSphere Studio provides a graphical editor for creating the WSDL file, which lays out the elements of the WSDL file as a hierarchy. It also contains an XSD editor for specifying the format of messages. Figure 7 shows the WSDL editor.

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

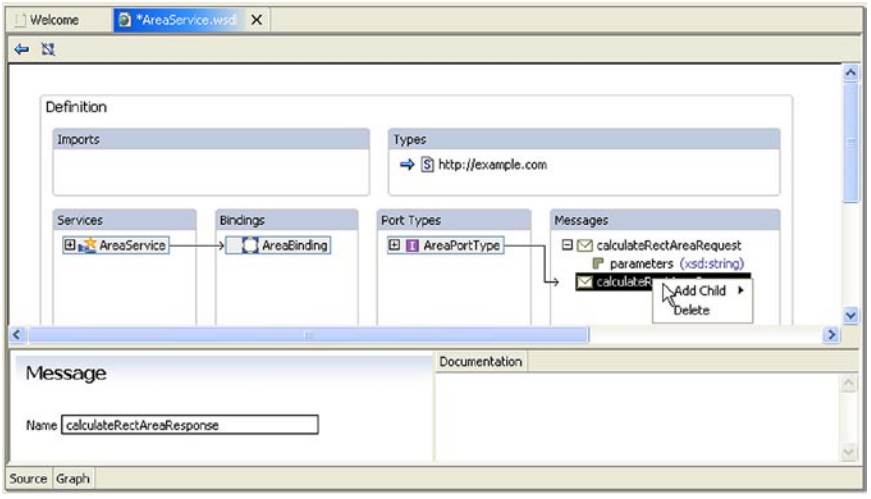


Figure 7. WSDL Editor

WebSphere Studio ships 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 WS-I conformance (which checks for interoperability with applications running on other WS-I compliant platforms), and monitor the execution of the Web services over the network using the TCP/IP Monitor.

To create a top-down Web service manually is an incredible amount of work. The WSDL file alone would be hundreds of lines of XML, plus writing the wrapper bean or EJB components, and creating a test client and proxy. Again, WebSphere Studio lets the developer concentrate on business logic rather than the administrivia.

Bottom-up Web services development

When turning an existing Java class or EJB component to into a Web service, developers can use a simple wizard to specify information about the component. WebSphere Studio generates the WSDL files describing the Web service, generates a SOAP deployment descriptor and the proxy that a client can use to access the Web service.

It deploys the Web service in 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. When it's time to publish the Web service, WebSphere Studio makes it easy to add the Web service to a UDDI registry for other users to find and use. Java and J2EE developers don't have to do any additional coding to offer their application as a Web service, so there's no learning curve for Web service development, and no productivity hit.

Additional productivity tools

WebSphere Studio provide a wealth of additional tools to enhance developer productivity, too numerous to mention here. This section covers just a few noteworthy features that can speed up 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 fix performance issues. WebSphere Studio provides several tools that let developers test an application's performance early in the development cycle. This early detection of performance issues can be a huge time saver, because it allows time for re-architecting the design before going too far down the implementation path.

The Profiling perspective provides tools that collect data about a program's run-time behavior, and display this 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 Profiler can gather information on a standalone Java application or an application running within a J2EE-based application server such as WebSphere Application Server. In both cases, profiling can be done 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.

Externally available toolkits

To enable integration with IBM and other vendors' server environments, IBM provides several freely available toolkits to download and install in WebSphere Studio. This is just a sampling of the most popular toolkits:

- ***Everyplace Toolkit for WebSphere Studio –*** extends the development environment to enable developers to design, develop, and implement portlet applications on mobile devices. Templates help them quickly and easily create own mobile portlets and applications. It provides additional

remote test facilities to help test applications before deploying. In addition, it contains a preview of the Multi-Device Authoring Technology tool set, which enables 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 WebSphere Studio 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 (JSP) or in existing transactional applications. It is available on the Web at: **lotus.com**/products/product4.nsf/wdocs/toolkitwsstudio

- ***IBM Deployment Toolkit for WebSphere Studio, WebLogic Edition*** – provides tools for working with BEA WebLogic Servers and files in WebSphere Studio. It supports WebLogic Server 6.1 and WebLogic Server 7.0. WebSphere Studio v 5.1.1 introduced support for WebLogic Server 8.1 as well. The toolkit lets developers develop, test, and deploy J2EE applications using the BEA WebLogic Server with WebSphere Studio. The WebLogic Server can be installed on the same machine as WebSphere Studio, or on a remote machine, and developers can launch the WebLogic Server console from within WebSphere Studio 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 listing, 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 WebSphere Studio Skills

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

Community Support

For online community support, visit developerWorks WebSphere Studio 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 WebSphere Studio features, and scenarios on how to use WebSphere Studio with other tools. You can also access online forums to trade tips and techniques with other WebSphere users, and find a local WebSphere User Group in your area.

Education and Training

IBM provides a wealth of course materials on using WebSphere Studio. At the developerWorks WebSphere Studio zone, you can find tutorials, distance learning classes with voice over 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

This paper has shown that WebSphere Studio 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 to quickly create and deploy e-business applications.

But don't take our word for it, try it out yourself! Visit the WebSphere Studio Trial Program page and download trials of WebSphere Studio Site Developer or Application Developer to start exploring this rich development environment today. **ibm.com**/developerworks/websphere/downloads/WSsupport.html

Resources

- ***developerWorks Subscription:*** A starter level subscription gives you low-cost access to WebSphere Studio Site Developer and even more IBM software: **ibm.com**/developerworks/subscription/

- ***WebSphere Studio 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

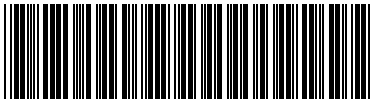
- ***IBM WebSphere Developer Technical Journal: Using EGL and Struts with WebSphere Studio Enterprise Developer Version 5:*** **ibm.com**/developerworks/websphere/techjournal/0304_barosa/barosa.html

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's currently writing a book on developing Web services with WebSphere Studio.

IBM Corporation
3039 Cornwallis Road
Research Triangle Park, NC 27709
U.S.A.

Produced in the United States of America
05-04 All Rights Reserved



G325-1122-00