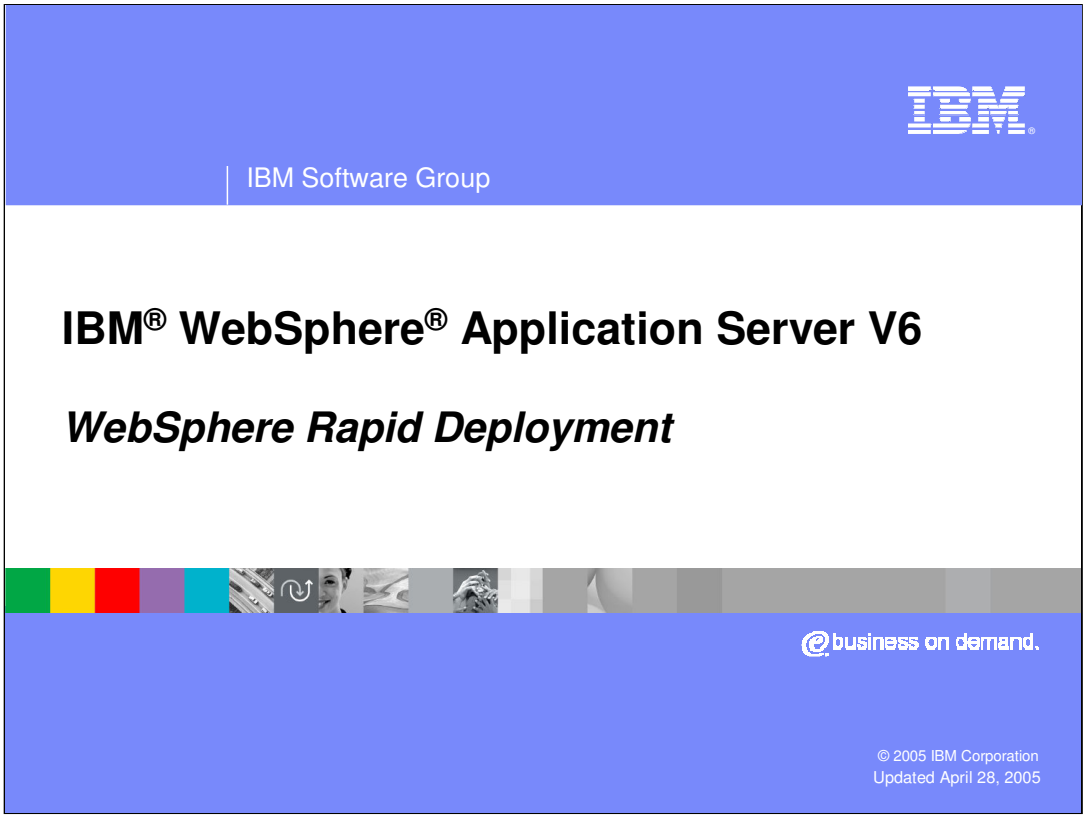


This presentation will focus on an overview of WebSphere Rapid Deployment, a new feature of IBM WebSphere Application Server V6 that allows developers an easier and faster way to deploy and develop Java™ 2 Enterprise Edition (J2EE) applications.

Goals

- Understand WebSphere Rapid Deployment
- Understand How to Apply the Benefits of WebSphere Rapid Deployment

The goals of this presentation are to understand the WebSphere Rapid Deployment feature and to understand how best to take advantage of the benefits offered by this new feature.

Agenda

- WebSphere Rapid Deployment Overview
  - WebSphere Rapid Deployment in general
  - Annotation-based Programming
  - Deployment Automation
- Common Usage Scenarios
- Summary

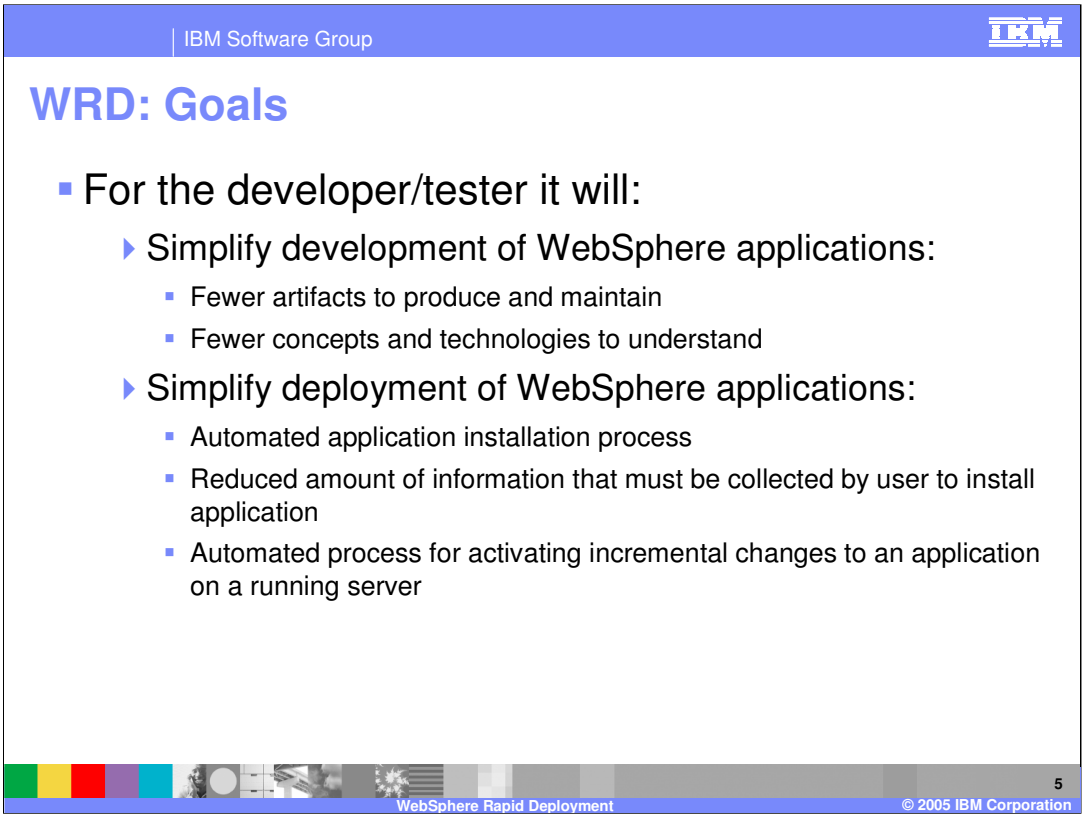
This presentation will cover an overview of WebSphere Rapid Deployment (WRD) in general terms and will also introduce the two concepts that make up this feature; deployment automation and annotation-based programming. You will also see a few usage scenarios of the technology to reinforce your understanding of the benefits that WRD provides.

Section

WebSphere Rapid Deployment (WRD)

Overview

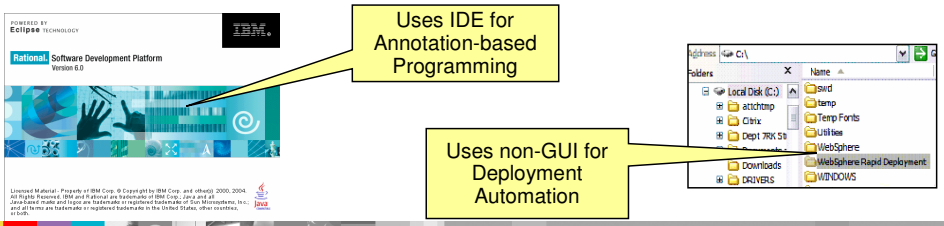
This section provides an overview of WebSphere Rapid Deployment.



What are the goals that WRD achieves? There are really two main areas upon which WRD is trying to simplify and improve. The first area is the development of WebSphere applications. By maintaining fewer artifacts, a developer can concentrate more on the business logic. The other area is fewer concepts and technologies to learn and understand. As an example, WRD has a style called free-form (or by-part application) that can construct a Java™ 2 Enterprise Edition (J2EE) application from just simple artifacts like servlets. There is no need to understand the project structure of a J2EE application.

WRD: What is it and Where does it run?

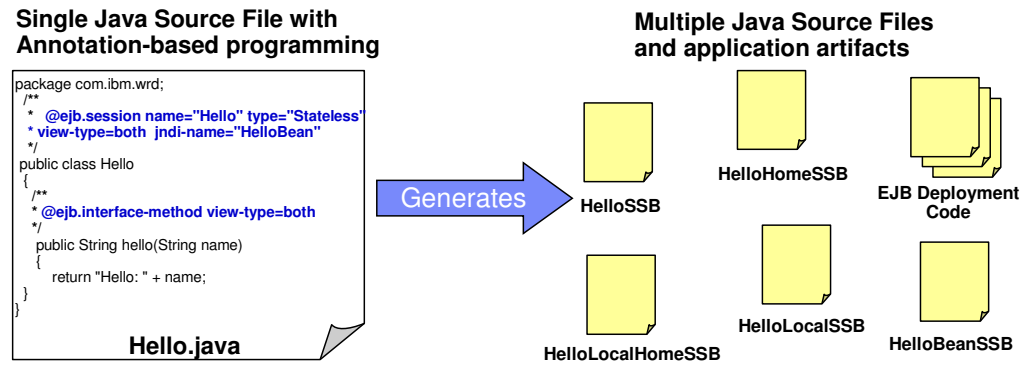
- Comprised of following key concepts:
  - Annotation-based programming
  - Deployment automation
- WRD is a collection of Eclipse plug-ins:
  - **Annotation-based programming:**
    - Used within IBM® Rational® Application Developer and Application Server Toolkit (AST) applications
  - **Deployment Automation:**
    - Uses a non-graphical user interface (GUI) mode on a user-defined file system directory, defined as a WRD workspace
    - Can be run from IBM Rational Application Developer runtime or WebSphere Application Server running on distributed platform (not supported running on z/OS)



As stated earlier, there are two key concepts that make up WRD; annotation-based programming and deployment automation. Before diving into the detail of what each is, you should first understand where WRD runs. First off, WRD requires no changes on the Application Server – it uses existing application server administration function to deploy and control applications. WRD is comprised of a collection of Eclipse plug-ins. When running WRD you are running an instance of Eclipse, however there is no graphical user-interface.

WRD: Annotation-based Programming

- Developer adds metadata tags into application source code
  - Uses XDoclet tag syntax, where defined
- WRD uses the metadata to generate additional artifacts needed to run the application on the Application Server
- Minimizes number of artifacts a developer needs to create and understand – user maintains the single artifact



Annotation-based programming (ABP) is the notion of allowing the developer to add additional metadata into the source code of their application and using that additional metadata to derive the additional artifacts necessary to execute the application in a J2EE environment. The goal of ABP is to minimize the number of artifacts that the developer has to create and understand, thereby simplifying their development experience. As an example, consider a stateless session EJB. With ABP the developer would simply create a single Java source file containing the bean implementation logic and a few tags indicating that they desire to deploy this class as an EJB and indications as to which methods should be made public on the interface of the EJB. Using this single artifact, WRD can create the home and remote interface classes, a stateless session implementation wrapper class, the ejb-jar.xml deployment descriptor, the WebSphere-specific binding data, and all of the remaining artifacts necessary to produce a compliant J2EE application. All the developer has to deal with is the single Java artifact.

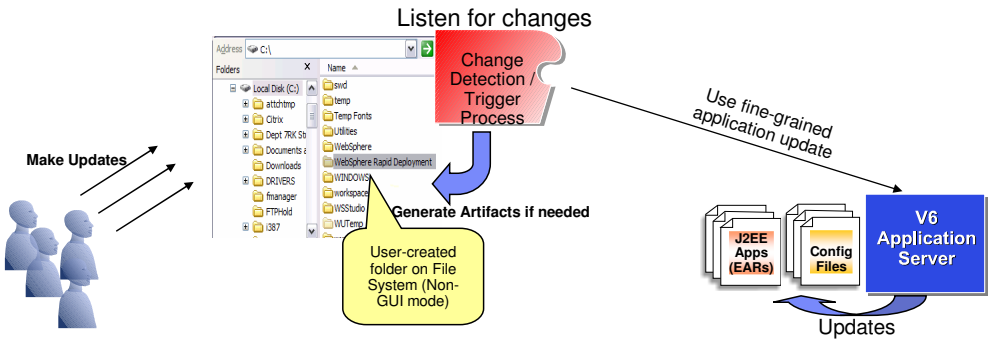
WRD: Deployment Automation

- Enable automatic installation of applications and modules onto a local or remote WebSphere Application Server running on any platform (including z/OS)
- Free form application development (initially only available in non-GUI mode)
  - Enables a “Hot Directory” concept for “file copy” and “Notepad” development and deployment
  - Constructs a well-formed EAR file from individual artifacts
  - Makes key decisions about default settings
- Support deployment of fine-grained application changes
- Goal of minimal application impact

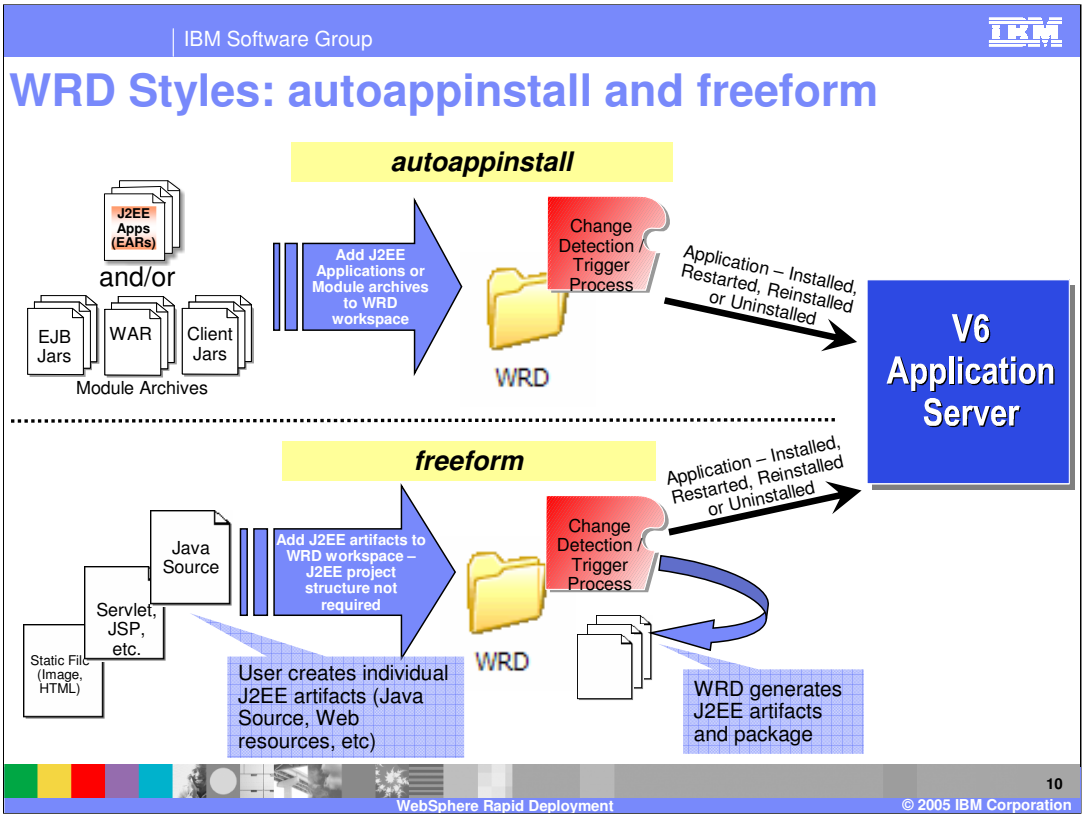
Deployment automation is the notion that the system monitors changes being made by a user and automatically ensures that those changes are reflected in a running copy of the application. In order to do this, the system will make decisions about default settings necessary to minimize the interaction required of the user. This monitoring may takes place in the form of an actively monitored folder on the file system. As an example, the user of the system can place fully composed applications (EARs), application modules (WARs, EJB Jars), or application artifacts (java source files, java class files, images, XML, HTML, etc) into a configurable location on their file system, and WRD will automatically detect the addition or modification of those parts and perform any necessary steps to produce a running application on WebSphere Application Server. WRD does this in a manner that is as efficient as possible, only performing the minimum number of steps required for the detected change.

WRD: Change Detection/Triggering Process

- Monitors the file system for changes in the WRD user workspace
- Drives processing operations based on the detection of change in artifacts of the application
  - Generates new application artifacts from existing artifacts
  - Drive deployment to the targeted WebSphere Application Server



In WRD, the deployment automation capability is provided by a collection of Eclipse incremental builders applied to the set of projects in an Eclipse workspace. The collection of builders can be applied to a project(s) in a way that combines to provide a particular behavior. Not all builders are applicable to all projects. The required set of builders and the appropriate order is determined by the set of artifacts that are expected in the source project and the required outcome. In WRD, this notion is called a Rapid Deployment Style. Styles are the mechanism exposed to a user to allow them to configure which WRD behaviors should be applied to their project. The notions of Styles is described below. Essentially deployment automation is achieved by using a style to configure an appropriate set of builders into an eclipse project. The set of builders then collaborate to automate the construction and activation of an application onto a WebSphere Application Server.



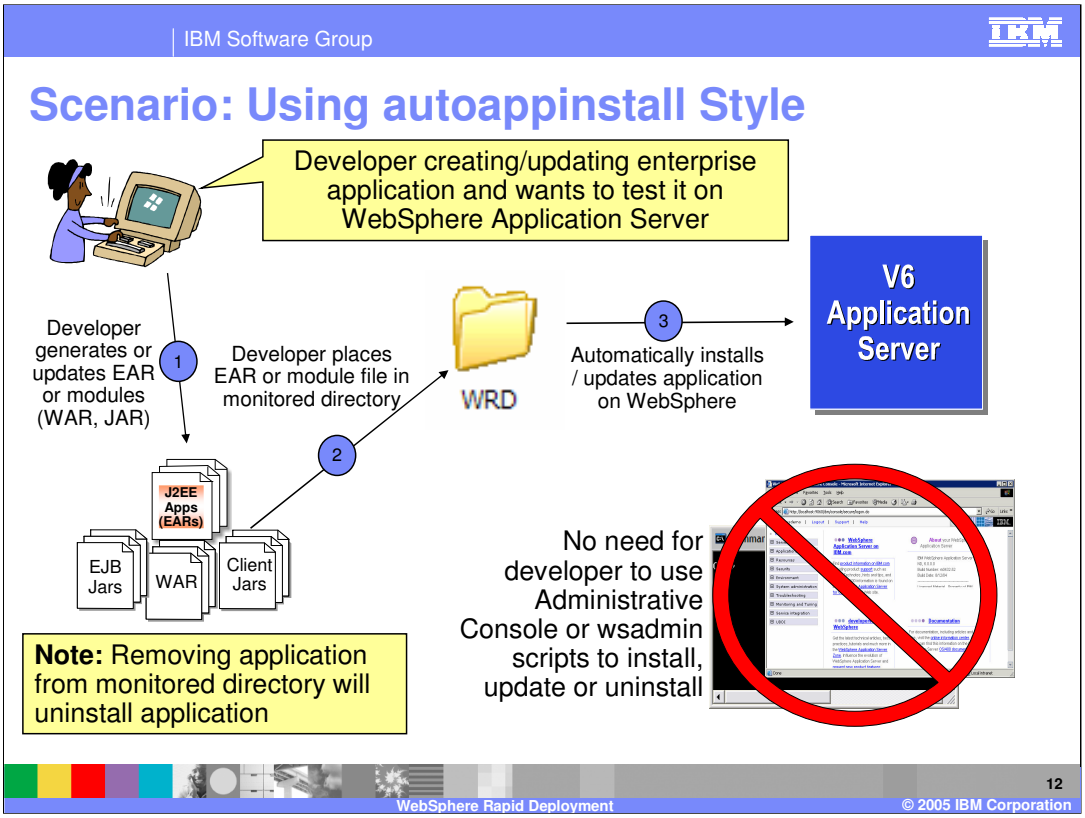
WRD could be used to create a style whose purpose is to automate the installation, modification, and uninstallation of J2EE compliant applications and modules. Instead of the customer taking an EAR file and using wsadmin or the Administration console to install that application through a multi-step wizard, WRD could be used to create a monitored directory in which the customer simply places the EAR. Upon placing the EAR in the directory, WRD would detect the addition of the new file. The builders configured by the style, in this case autoappinstall, would detect that the new part was an EAR, expand it into the installedApps folder of the local WebSphere installation, and install that application into the server using the standard WebSphere Application Management APIs (WAMA). If the customer placed on new copy of the same EAR in the directory, WRD would detect the modification, re-expand the EAR, and either call the reinstall APIs provided by WAMA, or simply restart the application using the standard WebSphere JMX APIs. Which path is taken can be determined by an activation builder based on the change that was detected. If the customer then deleted the EAR file from the directory, WRD would detect the deletion, stop the application, uninstall it using the WAMA, and then remove the expanded files. WRD would also, in addition to handling EAR files, consume individual J2EE modules (WARs, EJB Jars, RARs) and construct EARs as part of its execution.

WRD could also be used to reduce the complexity of application construction for the customer. In this scenario, instead of the customer providing a fully-constructed J2EE application, they would place in a directory the individual parts of the application, such as Java Source files that represent Servlets or EJBs, static resources, XML files, etc. WRD could then be configured to construct a J2EE compliant application and deploy that application on a target server. This style is called free form.

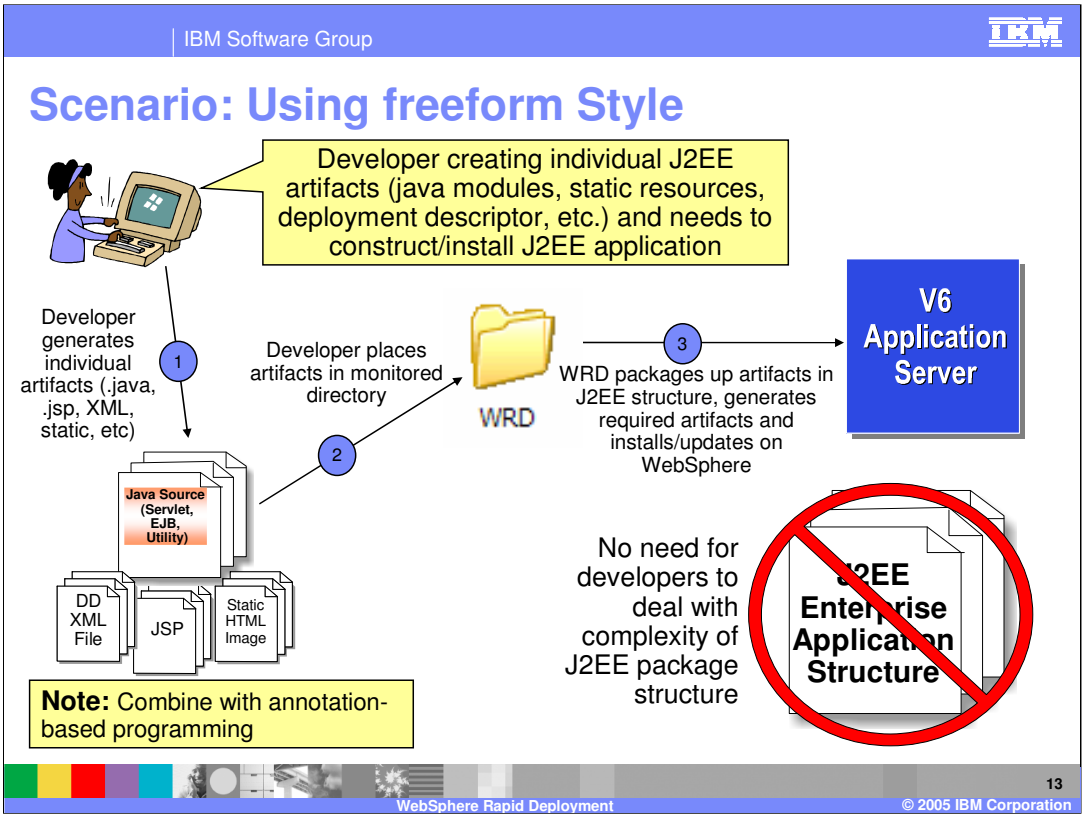
Section

Usage Scenarios

This section reviews several usage scenarios to reinforce the concept of WRD and the capabilities it provides.



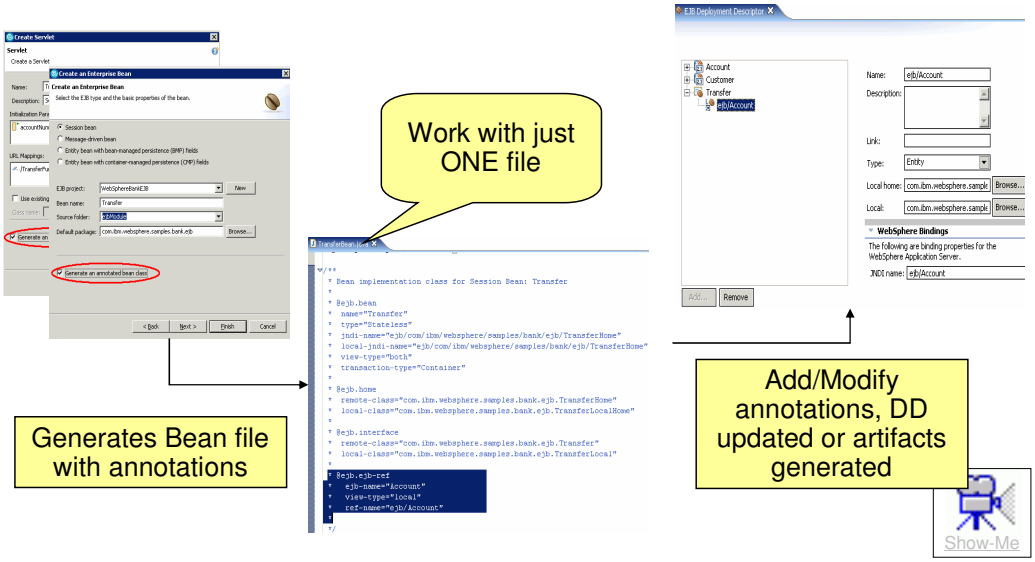
In this first scenario the developer needs an easy and fast way to deploy their application to a WebSphere Application Server for testing purposes. First the user will export their EAR file or modules for that EAR (WAR, EJB JAR, etc). The developer would then place those files in a configured WRD directory that uses the autoappinstall style. By placing those fully-formed J2EE application files in the monitored directory, the addition of the files would be detected, and, if necessary, WRD will construct a J2EE application that can be deployed and installed on WebSphere Application Server. If the developer decides to change or update the files, the change will be detected and WRD will send an update request to the WebSphere Application Server. If the file is removed, an uninstall application operation will be sent to the WebSphere Application Server.



Free from style allows the developer the freedom of not having to understand the structure of a J2EE application. In this scenario the developer places a single Java artifact, like a servlet, into the monitored directory that is configured to use the free form style of WRD deployment automation. The addition of the file will be detected and a fully-formed J2EE application will be constructed. After the construction of the application is complete, the application will then be deployed and installed on the target WebSphere Application Server. Just like the autoappinstall style, any changes or deletions of files in the monitored directory will cause WRD to send the appropriate command to WebSphere Application Server to perform a specified function. For example, removing a single servlet will cause the WAR module of the J2EE application to be updated. In WebSphere V6 it will use a new function called fine-grained application update to perform this, meaning it will only update the WAR file and not require the application to be restarted.

Scenario: Annotations with Tool products

- When creating EJBs/Servlets can generate annotations



Annotation-based programming is the other half of functionality that WRD provides. Here, developers can use tags to created their artifacts for their J2EE application. Integrated with the Application Server Toolkit (AST) and IBM Rational Application Developer is the support for annotation tags. When you generate either a servlet or EJB you now have the option to generate them with annotation tags. This will allow you to work with one source file for a specific artifact. Looking at this scenario as an example, using Rational Application Developer EJB creation wizard, you create a Session EJB and choose to generate annotations for the EJB. The generated artifacts, like the interface files, will be stored in a new directory called gen/src (generated source). The bean file, which would supposedly contain all of your business logic, would be under the Java Source directory just like previous releases of WebSphere's application development products. By generating annotations for the specific bean, you now only have to work with a single file. Making updates to the annotations of that file will generate the necessary artifacts and update the deployment descriptor as needed.

Section

Summary

Summary

- Described WebSphere Rapid Deployment
- Provided Usage Scenarios
- Real power of WRD comes when both annotation-based programming and deployment automation are brought together and utilized

In this presentation you learned about WebSphere Rapid Deployment and the functions it includes; deployment automation and annotation-based programming. You were also provided usage scenarios using each of the functions to reinforce the concepts. The real power of WRD comes from the combination of the two key concepts. WRD enables a model where a developer can be actively editing a small number of simple artifacts with annotations and, as they save changes to those artifacts, a running, specification-compliant copy of the application can be constantly updated and available in the background, thus reducing the developer's time of the edit-compile-debug cycle.

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.