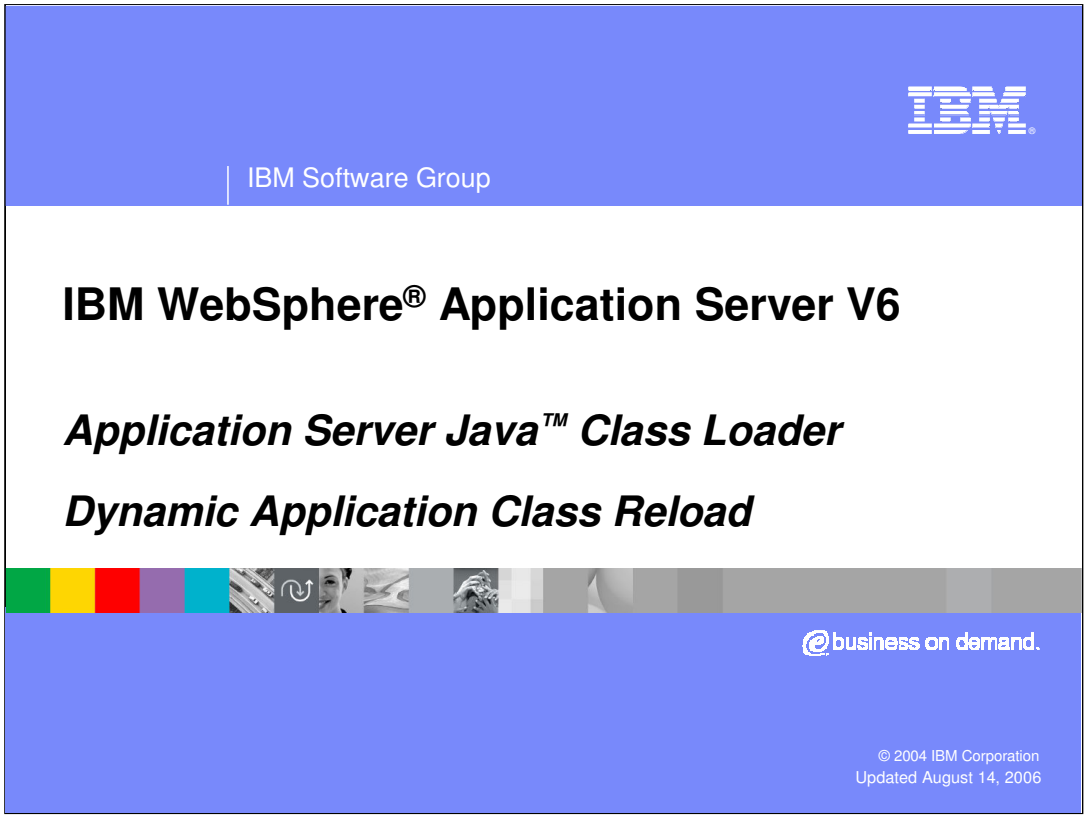
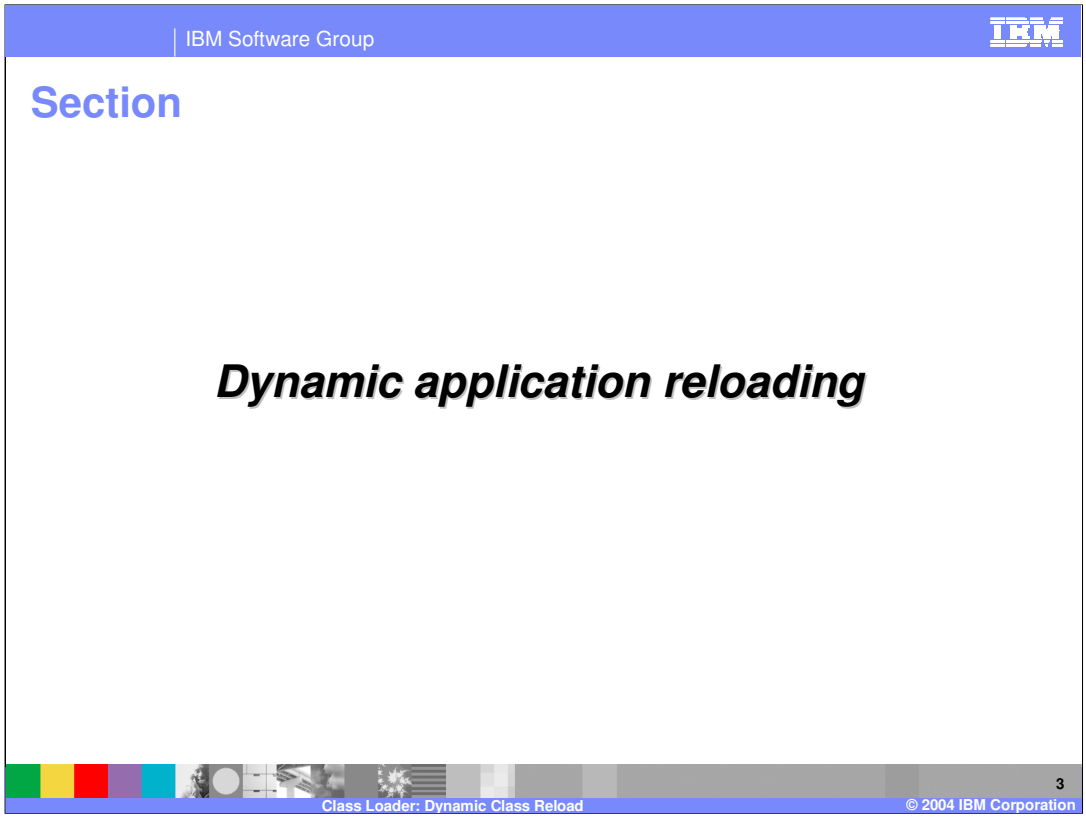


This presentation will focus on Dynamic class reload options with the WebSphere Application Server V6 Java Class Loader.

Goals

- Understand Dynamic Application Reload options and its effect
- Prerequisite:
  - Understanding of WebSphere V6 Class Loader provided in the WebSphere Application Server V6 Class Loader Overview presentation
  - Optional: Class loader Examples

The goal for this presentation is to understand dynamic application reload options and its effect. It is suggested that you first review the WebSphere Application Server V6 Class Loader Overview presentation and, optionally, the WebSphere Application Server V6 Class Loader Examples presentation.



The next section will discuss dynamic application reloading.

Dynamic application reloading - Overview

- Ability to change an existing application artifact without the need to restart the server, or in some cases, the entire application
- Available only when Application Class loading policy is MULTIPLE
- Changes to specific application artifacts are detected for dynamic reloading
  - These changes are listed on an upcoming page

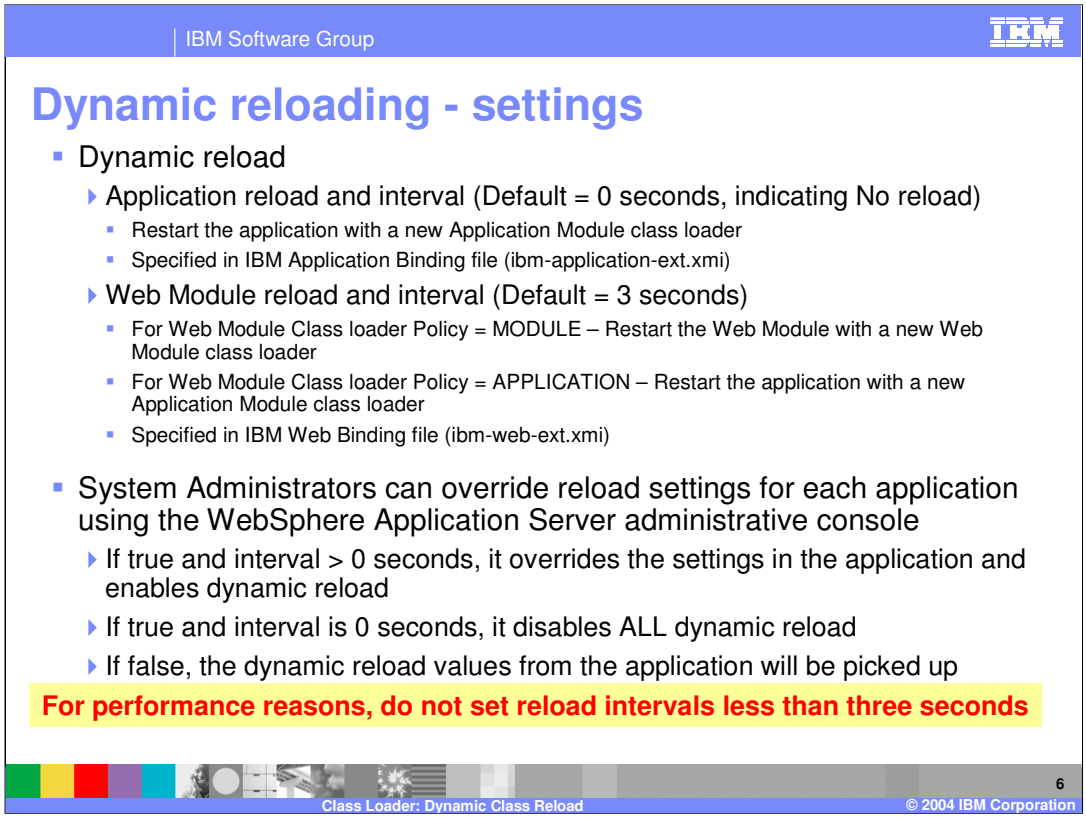
Dynamic application reload is the ability for the Application Server to reload certain changes in applications, without the need to restart the server. In some cases, only the affected module needs to be restarted. The ability to dynamically reload is based on the kind of changes in the application. The specific changes are listed on an upcoming page.

When the Server-level application class loader policy is SINGLE, the server cannot recreate the class loader without affecting other applications in the server. Therefore, dynamic reload is not available if the class loader policy is SINGLE. Dynamic reload is available in situations where each application has its own class loader, as is the case with Server-level application class loader policy of MULTIPLE.

Dynamic reloading – How does it work?

- It creates an alarm that awakens over a specified interval for each EAR and Web module of an application
- It then associates each alarm to the Application module or War module class loader
- When the interval expires, the alarm notifies the associated class loader to check if specific items on its classpath have changed
- If yes, it directs the runtime to stop the application and then restart it
  - The stopping of the application will discard the Application class loader and all the classes loaded by the application
  - The restart of the application will construct a new Application class loader and load the revised classes and resources

The process of dynamic reload is described here. An alarm is created for each Application and Web module class loader. At regular intervals, the alarm notifies its class loader to check for changes. If changes are detected, the application or the module is stopped and restarted. The process of stopping and restarting will discard the previous class loader, and a new class loader will be created which will load the new, affected classes.



Dynamic reload is specified using the Rational® Web or Application Developer, as well as the Application Server Toolkit (AST). The settings are specified as WebSphere extensions. The reload intervals are applied at the Application level and the Web module levels. Application reload is for application class changes, whereas the Web module reload is for Web class changes.

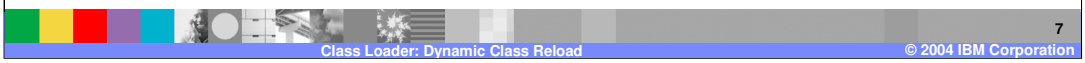
If the Web module class loader policy is Module, indicating that the web module will have its own class loader, then dynamic reload will need to restart the web module only and not the entire application. For Application class changes or Web module changes when the Web class loader policy is Application, the entire application will be restarted.

The reload settings specified in the application can be overridden by the System Administrator for each application separately. The override is limited to disabling the reload or enabling the reload with a specific interval.

The smaller the reload interval, the greater will be the frequency of the alarm requesting the class loader check for changes. For performance reasons, it is suggested to not set the reload intervals to less than three seconds.

Section

Application artifacts used for dynamic reload detection



The next section will discuss application artifacts used for dynamic reload detection.

Dynamic reload detection on application files

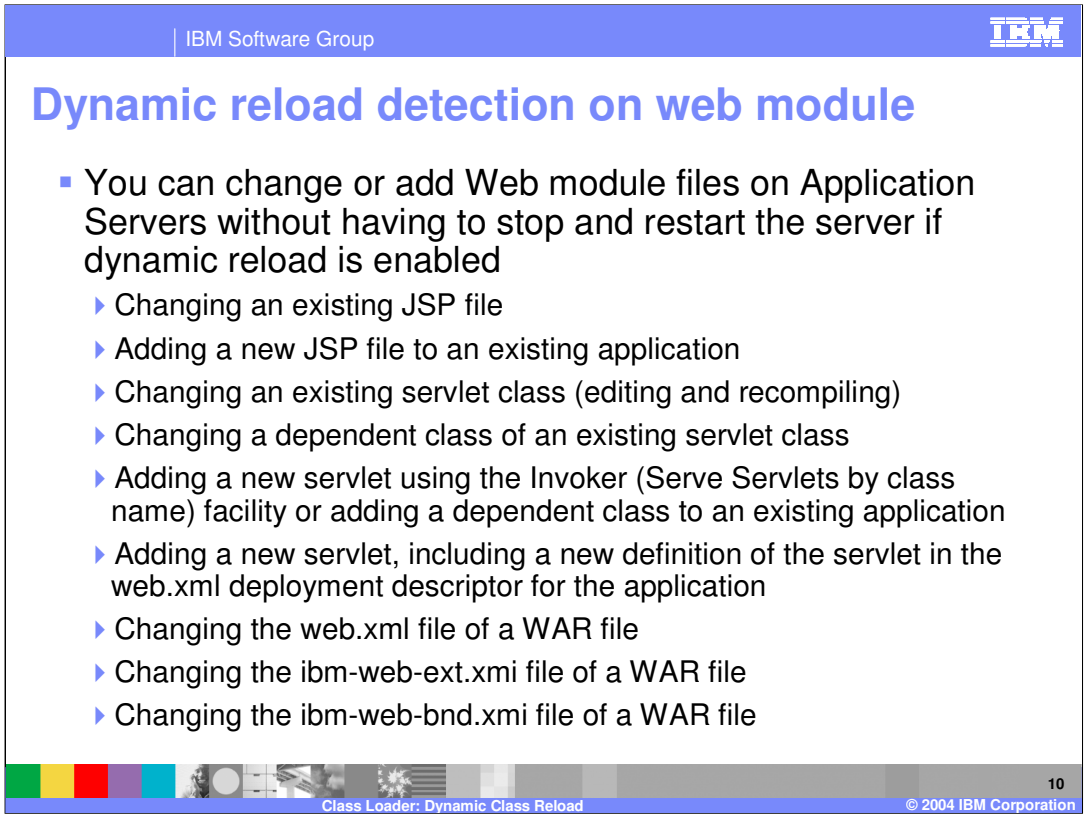
- You can change or add application files on Application Servers without having to stop and restart the server if dynamic reload is enabled
  - Updating an existing application on an active server (providing a new EAR file)
  - Adding a new application to an active server
  - Removing an existing application from an active server
  - Changing or adding files to existing Enterprise JavaBean or Web modules
  - Changing the application.xml file for an application
  - Changing the ibm-app-ext.xmi file for an application
  - Changing the ibm-app-bnd.xmi file for an application
  - Changing a non-module JAR file contained in the EAR file

This list shows the application file changes that dynamic reload will detect and perform restart of the application, if dynamic reload is enabled. This includes: Updating an existing application on an active server (providing a new EAR file), adding a new application to an active server, removing an existing application from an active server, changing or adding files to an existing Enterprise JavaBean or Web modules, changing the application.xml file for an application, changing the ibm-app-ext.xmi file for an application, changing the ibm-app-bnd.xmi file for an application, and changing a non-module JAR file contained in the EAR file.

Dynamic reload detection on EJB module

- You can change or add EJB module files on Application Servers without having to stop and restart the server if dynamic reload is enabled
  - Changing the ejb-jar.xml file of an EJB Jar file
  - Changing the ibm-ejb-jar-ext.xmi or ibm-ejb-jar-bnd.xmi file of an EJB Jar file
  - Changing the Table.ddl file for an EJB Jar file
  - Changing the Map.mapxmi or Schema.dbxmi file for an EJB Jar file
  - Updating the implementation class for an EJB file or a dependent class of the implementation class for an EJB file
  - Updating the Home or Remote interface class for an EJB file
  - Adding a new EJB file to an existing EJB Jar file

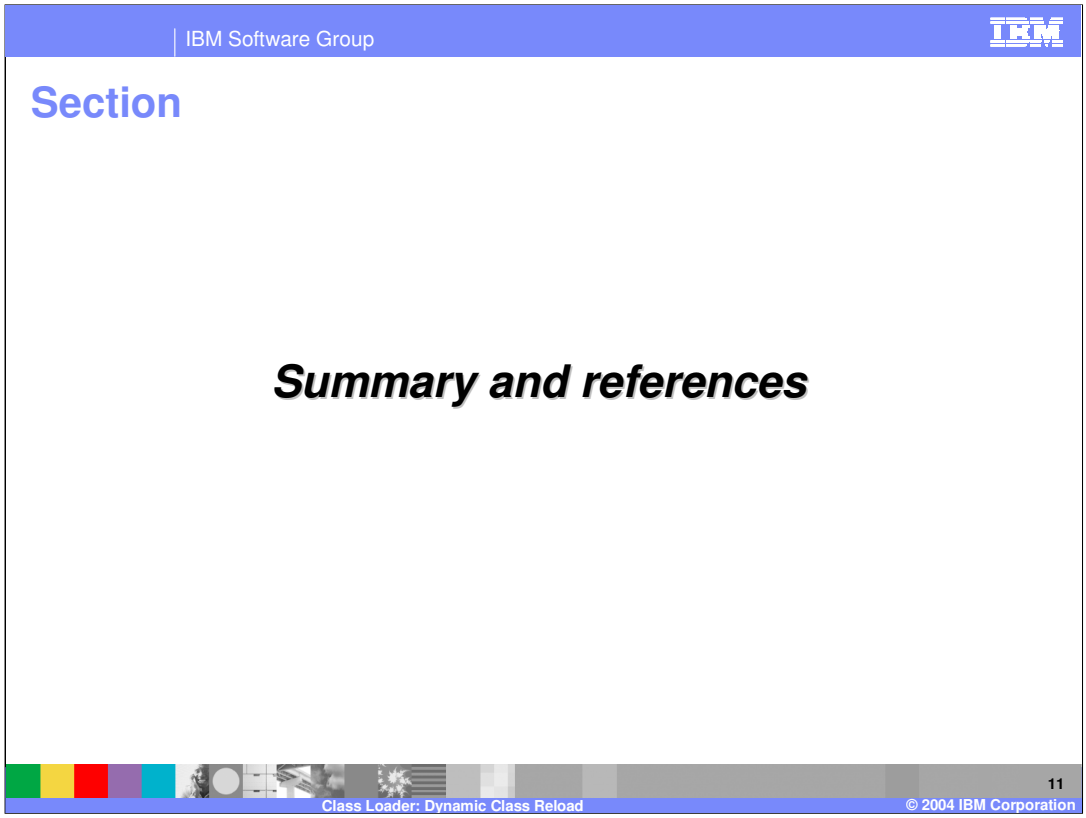
This list shows the EJB module file changes that dynamic reload will detect and perform a restart of the application, and in turn the EJB modules, if dynamic reload is enabled. This includes: changing the ejb-jar.xml file of an EJB Jar file, changing the ibm-ejb-jar-ext.xmi or ibm-ejb-jar-bnd.xmi file of an EJB Jar file, changing the Table.ddl file for an EJB Jar file, changing the Map.mapxmi or Schema.dbxmi file for an EJB Jar file, updating the implementation class for an EJB file or a dependent class of the implementation class for an EJB file, updating the Home or Remote interface class for an EJB file, and adding a new EJB file to an existing EJB Jar file.



The list shows the Web module file changes that dynamic reload will detect and perform reload of either the Web module or the entire application, if dynamic reload is enabled. This list includes: changing an existing JSP file, adding a new JSP file to an existing application, changing an existing servlet class (editing and recompiling), changing a dependent class of an existing servlet class, adding a new servlet using the Invoker (Serve Servlets by class name) facility or adding a dependent class to an existing application, adding a new servlet, including a new definition of the servlet in the web.xml deployment descriptor for the application, changing the web.xml file of a WAR file, changing the ibm-web-ext.xmi file of a WAR file, and changing the ibm-web-bnd.xmi file of a WAR file.

If the Web module class loader policy is Module, only the Web module will be restarted due to reload.

If the Web module class loader policy is Application, the application will be restarted due to reload.



The next section will discuss a summary of the concepts you've just reviewed.

Summary

- Dynamic Reload option provides the ability for the Application Server to automatically reload certain application changes without the need to restart the server

In summary, this presentation has focused on how the dynamic reload option provides the ability for the Application Server to reload certain changes in applications without the need to restart the server.

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.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.