An Overview of the WebSphere Application Server Class Loader

Thanh Giang(tgiang@us.ibm.com)
WebSphere Application Server Level 2 support
1 December 2011

Agenda

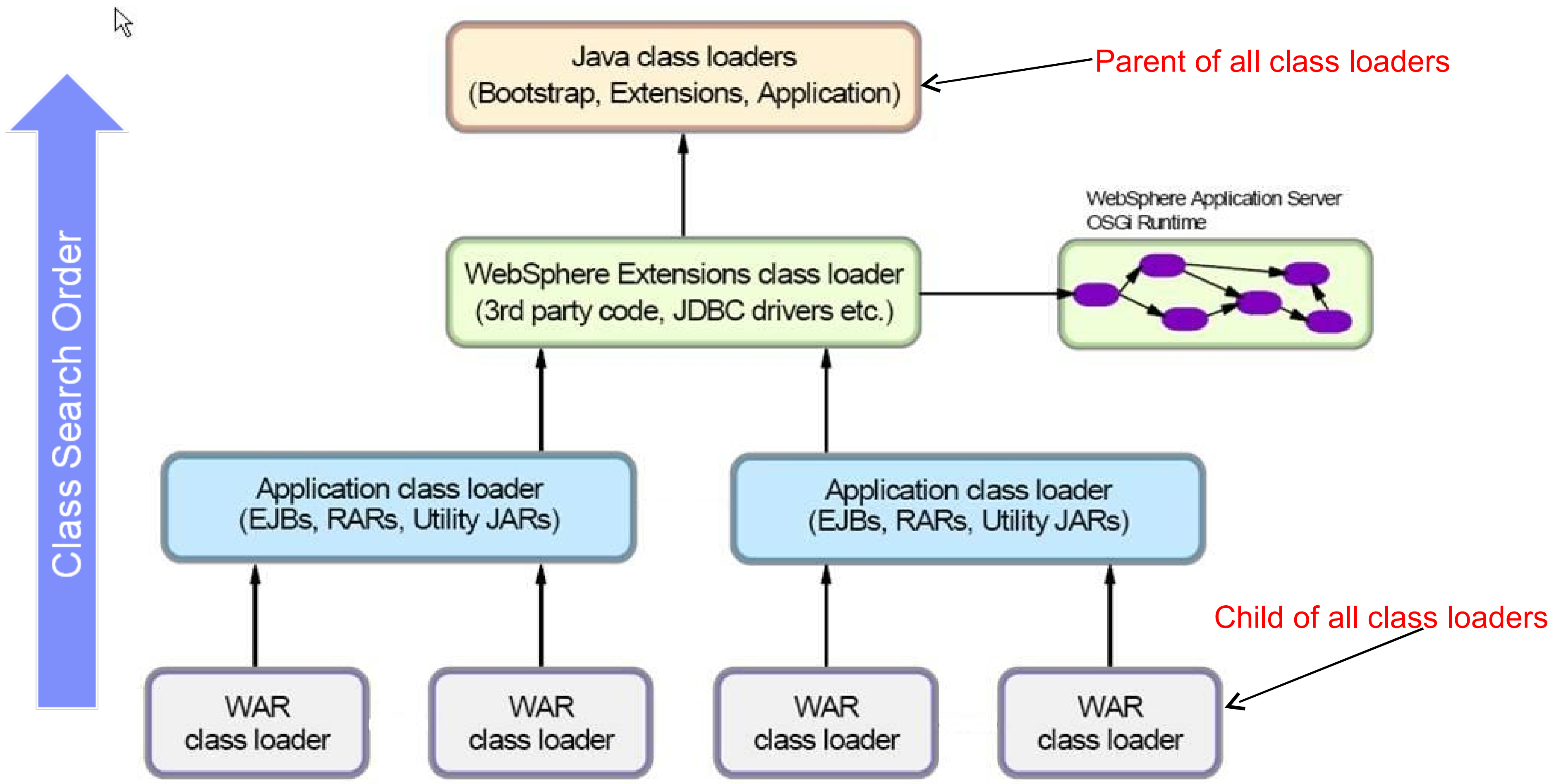
- WebSphere Class loader Hierarchy
  - Java™ Class loader
  - WebSphere Application class loader
  - OSGi class loader
- Configuring WebSphere Class loader
  - Class loader policy and class loader mode
  - Examples
- Shared library
  - Shared library configuration
  - Isolated shared library – new in WebSphere 7.0
- Common issues
- References

WebSphere Class loader Hierarchy

What is Class loader?

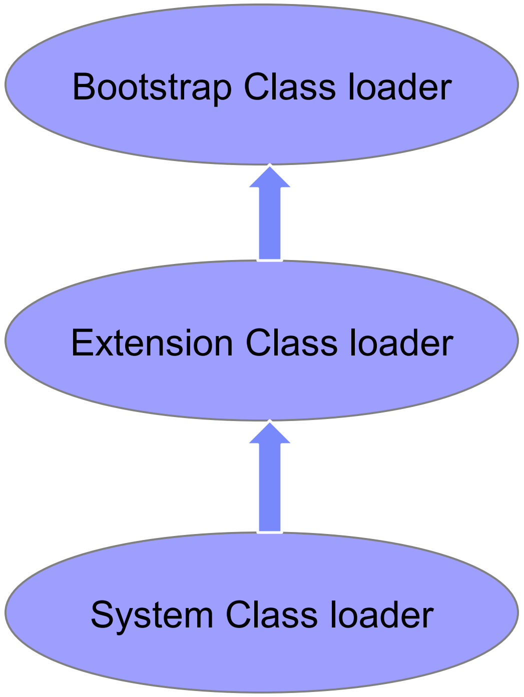
- A Class loader is an object that uses a delegation model to find and load classes and resources.
- Each instance of ClassLoader has an associated parent class loader.
- Requests for finding and loading the class can only go to a parent class loader; they cannot go to a child class loader.

WebSphere Class loader Hierarchy


Class Search Order
Java class loaders (Bootstrap, Extensions, Application)
Parent of all class loaders
WebSphere Application Server OSGi Runtime
WebSphere Extensions class loader (3rd party code, JDBC drivers etc.)
Application class loader (EJBs, RARs, Utility JARs)
Application class loader (EJBs, RARs, Utility JARs)
Child of all class loaders
WAR class loader
WAR class loader
WAR class loader
WAR class loader

Java Class loader

Java Class loader uses for loading classes during Java virtual machine (JVM) startup. It performs three major tasks: load Java class files, load resource files and locate native code shared library (.dll, .so).

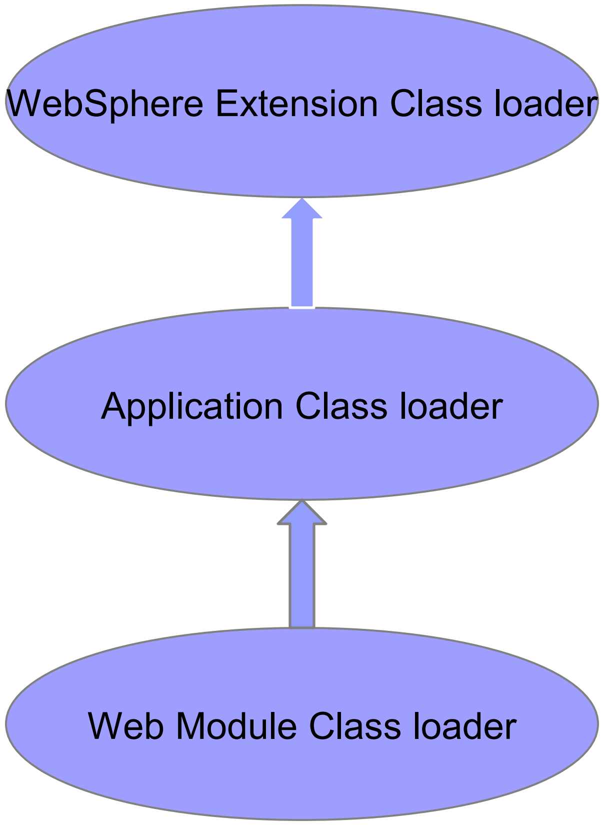


- Implemented by **NATIVE code**
- Load the the jar files in <JAVA_HOME>/jre/lib folder

- Implemented by **sun.misc.Launcher$ExtClassLoader**
- Load <JAVA_HOME>/jre/lib/ext and directory specified by the system property java.ext.dirs

- Implemented by **sun.misc.Launcher$AppClassLoader**
- Load directories and jar files specified by the CLASSPATH (java.class.path)

WebSphere Class loader

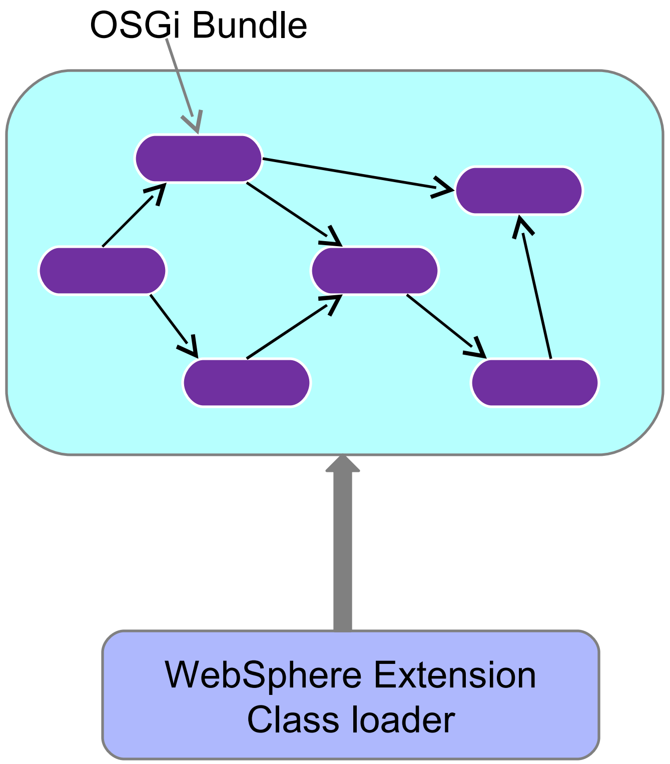


- Implemented by **com.ibm.ws.bootstrap.ExtClassLoader**
- <WAS_HOME>/classes
- <WAS_HOME>/lib
- <WAS_HOME>/lib/ext

- Implemented by **com.ibm.ws.classloader.CompoundClassLoader**
- EJB Modules, web modules, application client modules, resource adapters (RAR files) and dependency or utility JARs at the root of <WAS_PROFILE>/installedApps/EAR/ folder
- Application-associated shared libraries

- Implemented by **com.ibm.ws.classloader.CompoundClassLoader**
- WEB-INF/classes
- WEB-INF/lib
- shared libraries associated with the web module

OSGi Runtime class loader



- New in version 6.1
- WebSphere Application server (WAS) runtime classes are packaged as a set of OSGi bundles and are loaded by a network of several OSGi class loaders.
- Each OSGi bundle is loaded separately by its own class loader.
- The OSGi class loader is connected to the extensions class loader and the rest of the hierarchy through an OSGi gateway class loader.
- Implemented by **org.eclipse.osgi.internal.baseadaptor.DefaultClassLoader**
- OSGi bundle JARs are located in <WAS_HOME>/plugins folder.

Configuring WebSphere Class loader

Class loader Delegation Mode

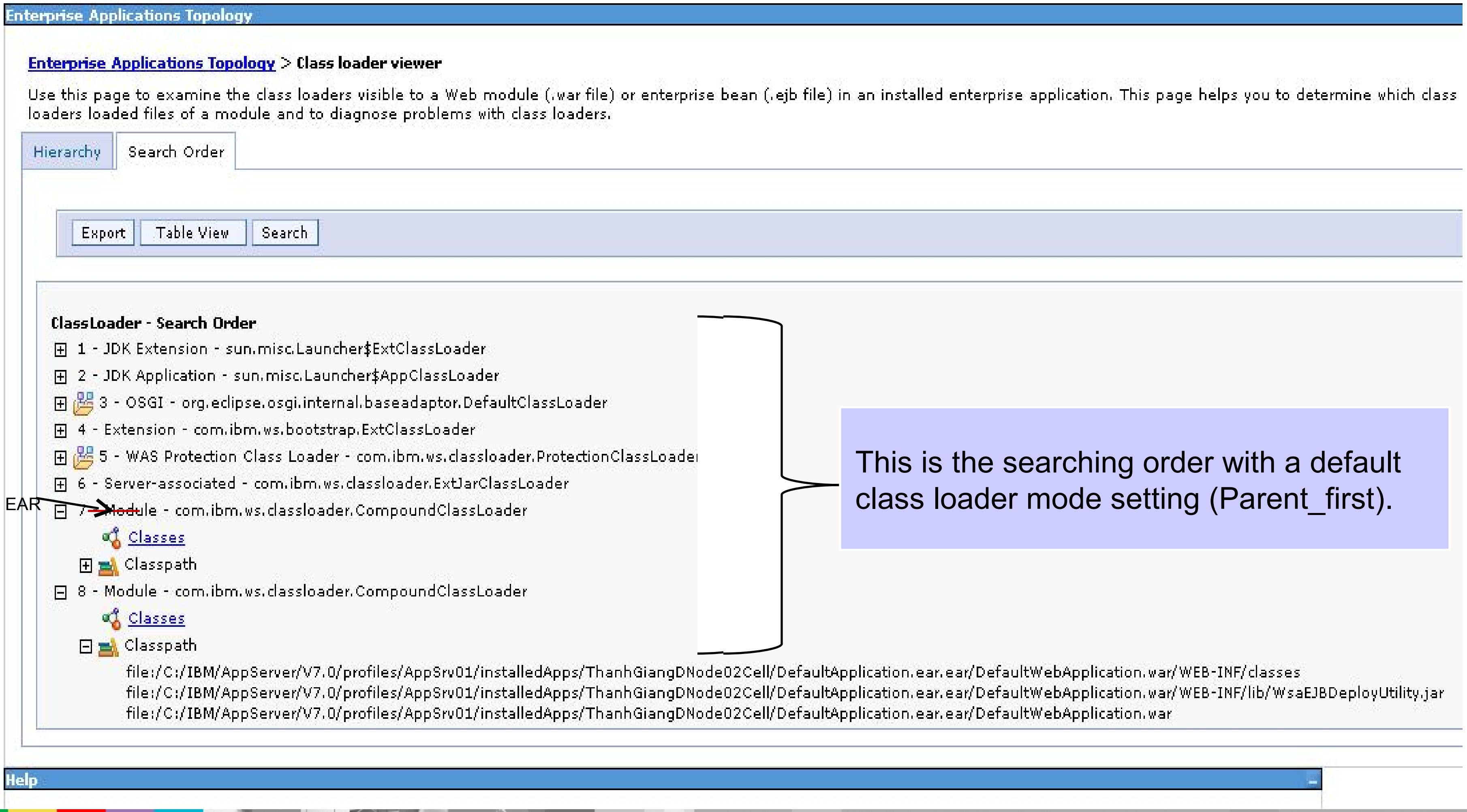
Every class loader has a delegation mode:

PARENT_FIRST (default)
The Parent first class-loader mode causes the class loader to delegate the loading of classes to its parent class loader before attempting to load the class from its local class path. This value is the default for the class-loader policy and for standard JVM class loaders.

PARENT_LAST
The Parent last class-loader mode causes the class loader to attempt to load classes from its local class path before delegating the class loading to its parent. Using this class loader mode, an application class loader can override and provide its own version of a class that exists in the parent class loader.

<u>Note</u>: A class loader can only delegate requests to its parent class loader, never to its child class loaders (it can go up the hierarchy but never down)

Class loader Viewer: Parent_first



Enterprise Applications Topology

Enterprise Applications Topology > Class loader viewer

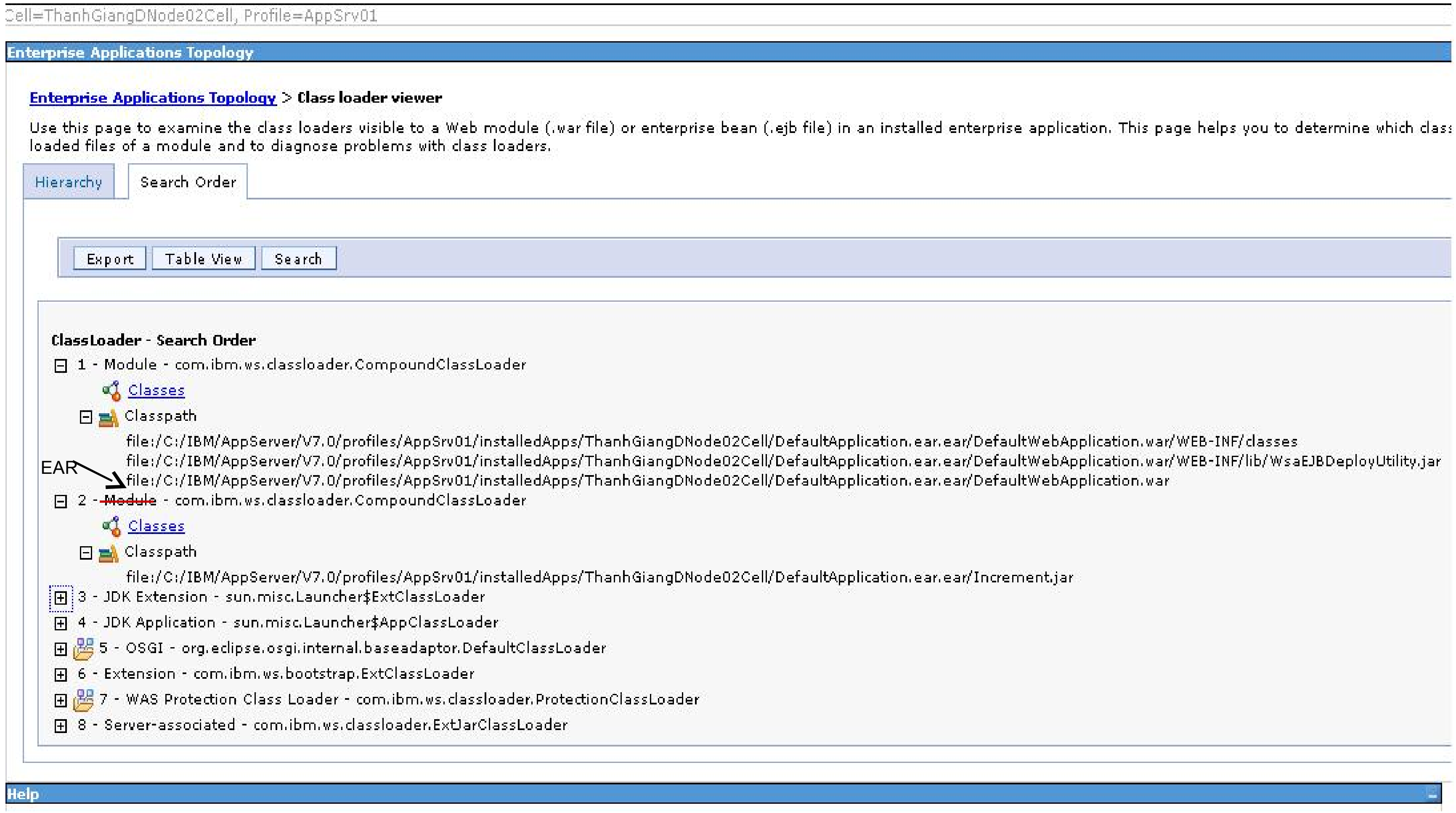
Use this page to examine the class loaders visible to a Web module (.war file) or enterprise bean (.ejb file) in an installed enterprise application. This page helps you to determine which class loaders loaded files of a module and to diagnose problems with class loaders.

ClassLoader - Search Order

```
⊞ 1 - JDK Extension - sun.misc.Launcher$ExtClassLoader
⊞ 2 - JDK Application - sun.misc.Launcher$AppClassLoader
⊞ 3 - OSGI - org.eclipse.osgi.internal.baseadaptor.DefaultClassLoader
⊞ 4 - Extension - com.ibm.ws.bootstrap.ExtClassLoader
⊞ 5 - WAS Protection Class Loader - com.ibm.ws.classloader.ProtectionClassLoader
⊞ 6 - Server-associated - com.ibm.ws.classloader.ExtJarClassLoader
⊟ 7 - Module - com.ibm.ws.classloader.CompoundClassLoader
      Classes
   ⊞ Classpath
⊟ 8 - Module - com.ibm.ws.classloader.CompoundClassLoader
      Classes
   ⊟ Classpath
      file:/C:/IBM/AppServer/V7.0/profiles/AppSrv01/installedApps/ThanhGiangDNode02Cell/DefaultApplication.ear.ear/DefaultWebApplication.war/WEB-INF/classes
      file:/C:/IBM/AppServer/V7.0/profiles/AppSrv01/installedApps/ThanhGiangDNode02Cell/DefaultApplication.ear.ear/DefaultWebApplication.war/WEB-INF/lib/WsaEJBDeployUtility.jar
      file:/C:/IBM/AppServer/V7.0/profiles/AppSrv01/installedApps/ThanhGiangDNode02Cell/DefaultApplication.ear.ear/DefaultWebApplication.war
```

This is the searching order with a default class loader mode setting (Parent_first).

Class loader viewer: Parent_last



Cell=ThanhGiangDNode02Cell, Profile=AppSrv01

Enterprise Applications Topology

Enterprise Applications Topology > Class loader viewer

Use this page to examine the class loaders visible to a Web module (.war file) or enterprise bean (.ejb file) in an installed enterprise application. This page helps you to determine which class loaded files of a module and to diagnose problems with class loaders.

ClassLoader - Search Order

```
⊟ 1 - Module - com.ibm.ws.classloader.CompoundClassLoader
      Classes
   ⊟ Classpath
      file:/C:/IBM/AppServer/V7.0/profiles/AppSrv01/installedApps/ThanhGiangDNode02Cell/DefaultApplication.ear.ear/DefaultWebApplication.war/WEB-INF/classes
      file:/C:/IBM/AppServer/V7.0/profiles/AppSrv01/installedApps/ThanhGiangDNode02Cell/DefaultApplication.ear.ear/DefaultWebApplication.war/WEB-INF/lib/WsaEJBDeployUtility.jar
      file:/C:/IBM/AppServer/V7.0/profiles/AppSrv01/installedApps/ThanhGiangDNode02Cell/DefaultApplication.ear.ear/DefaultWebApplication.war
```

EAR

⊟ 2 - ~~Module~~ - com.ibm.ws.classloader.CompoundClassLoader

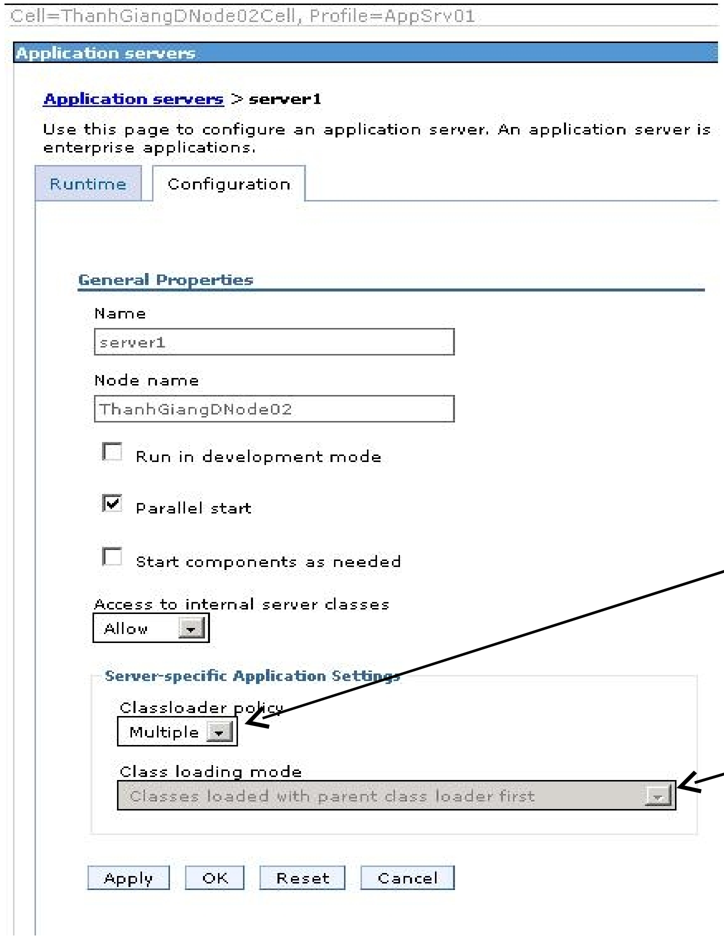
```
      Classes
   ⊟ Classpath
      file:/C:/IBM/AppServer/V7.0/profiles/AppSrv01/installedApps/ThanhGiangDNode02Cell/DefaultApplication.ear.ear/Increment.jar
⊞ 3 - JDK Extension - sun.misc.Launcher$ExtClassLoader
⊞ 4 - JDK Application - sun.misc.Launcher$AppClassLoader
⊞ 5 - OSGI - org.eclipse.osgi.internal.baseadaptor.DefaultClassLoader
⊞ 6 - Extension - com.ibm.ws.bootstrap.ExtClassLoader
⊞ 7 - WAS Protection Class Loader - com.ibm.ws.classloader.ProtectionClassLoader
⊞ 8 - Server-associated - com.ibm.ws.classloader.ExtJarClassLoader
```

Help

Class loader Policy

- Application class loader policy controls the isolation of applications
  - **Multiple** (default): Each application is loaded by its own class loader
  - **Single**: All applications are loaded by the same class loader
- Web module class loader policy controls the isolation of Web Module
  - **Module** (Default): Each web module in the application is loaded by a separate class loader
  - **Application:** All the web modules in the application EAR are loaded by the single application class loader

Server Class loader Setting



Cell=ThanhGiangDNode02Cell, Profile=AppSrv01

Application servers

Application servers > server1

Use this page to configure an application server. An application server is enterprise applications.

Runtime | Configuration

General Properties

Name: server1

Node name: ThanhGiangDNode02

☐ Run in development mode

☑ Parallel start

☐ Start components as needed

Access to internal server classes: Allow

Server-specific Application Settings

Classloader policy: Multiple

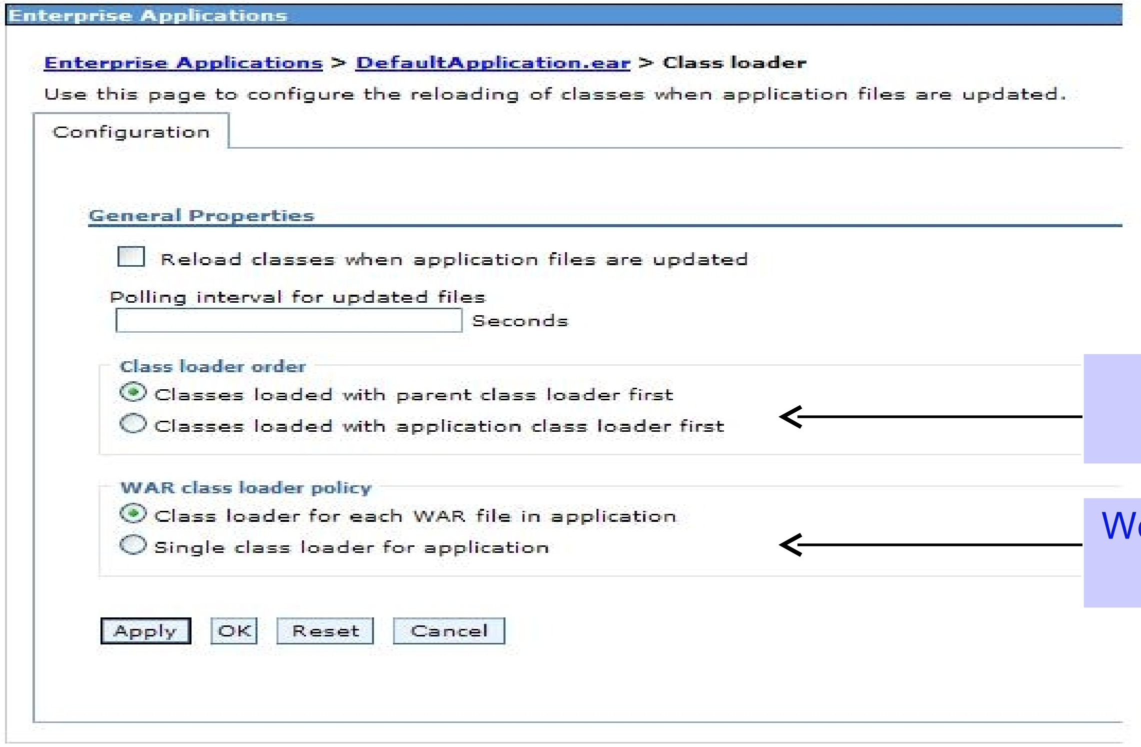
Class loading mode: Classes loaded with parent class loader first

Apply | OK | Reset | Cancel

EAR class loader policy: Multiple / Single

Server class loader mode: Parent_first / Parent_last

Application Class loader Settings



Enterprise Applications

Enterprise Applications > DefaultApplication.ear > Class loader

Use this page to configure the reloading of classes when application files are updated.

Configuration

General Properties

☐ Reload classes when application files are updated

Polling interval for updated files
[ ] Seconds

Class loader order
- [✓ Classes loaded with parent class loader first]
- Classes loaded with application class loader first

WAR class loader policy
- [✓ Class loader for each WAR file in application]
- Single class loader for application

EAR class loader mode

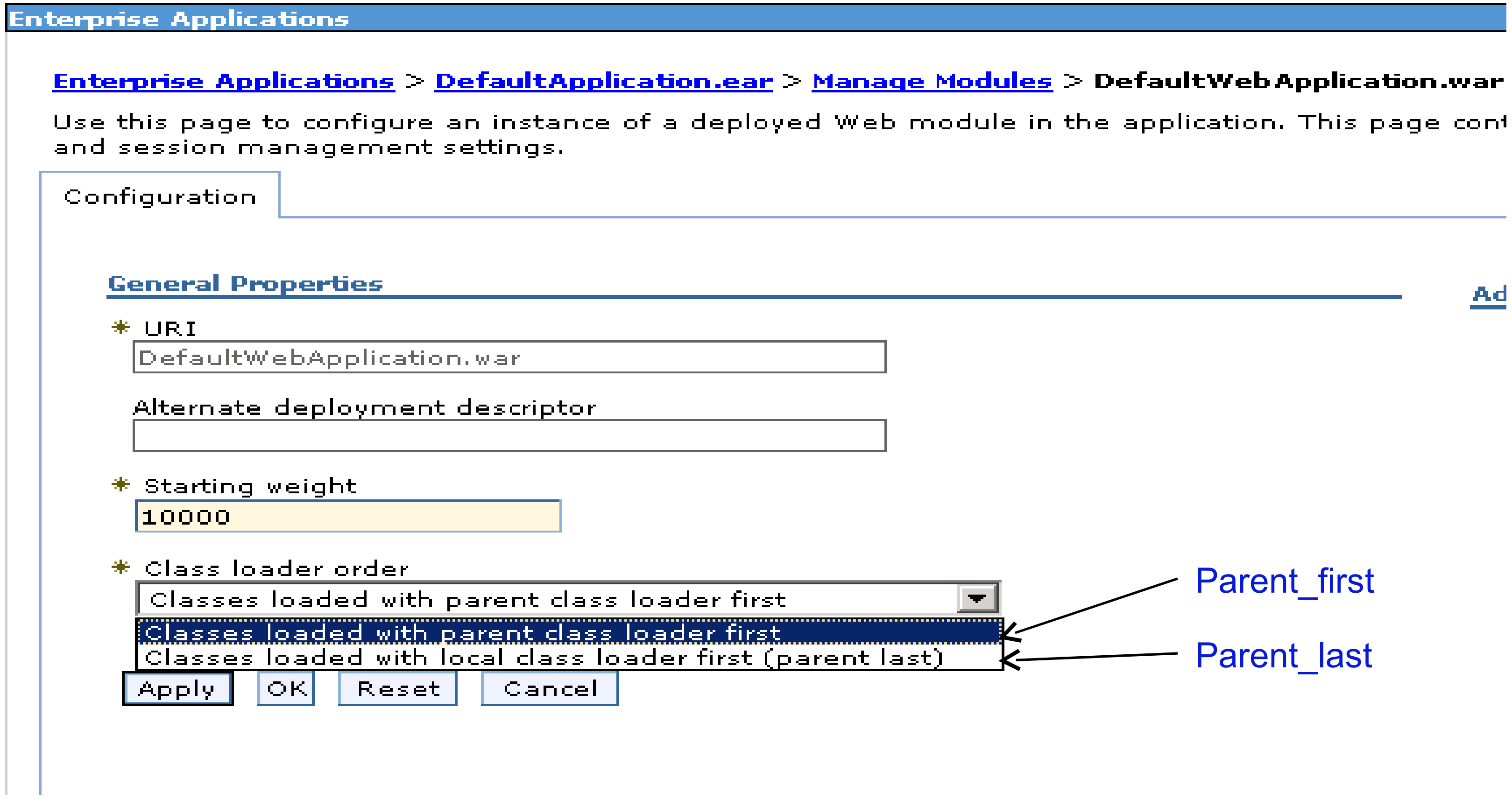
(parent_first / parent_last)

Web module class loader policy

(Application / Module)

Enterprise Applications > *application_name* > Class loading and update detection

Web Module Class loader Setting



Enterprise Applications

Enterprise Applications > DefaultApplication.ear > Manage Modules > DefaultWebApplication.war

Use this page to configure an instance of a deployed Web module in the application. This page conf
and session management settings.

Configuration

General Properties

* URI
DefaultWebApplication.war

Alternate deployment descriptor

* Starting weight
10000

* Class loader order
Classes loaded with parent class loader first

- Classes loaded with parent class loader first
- Classes loaded with local class loader first (parent last)

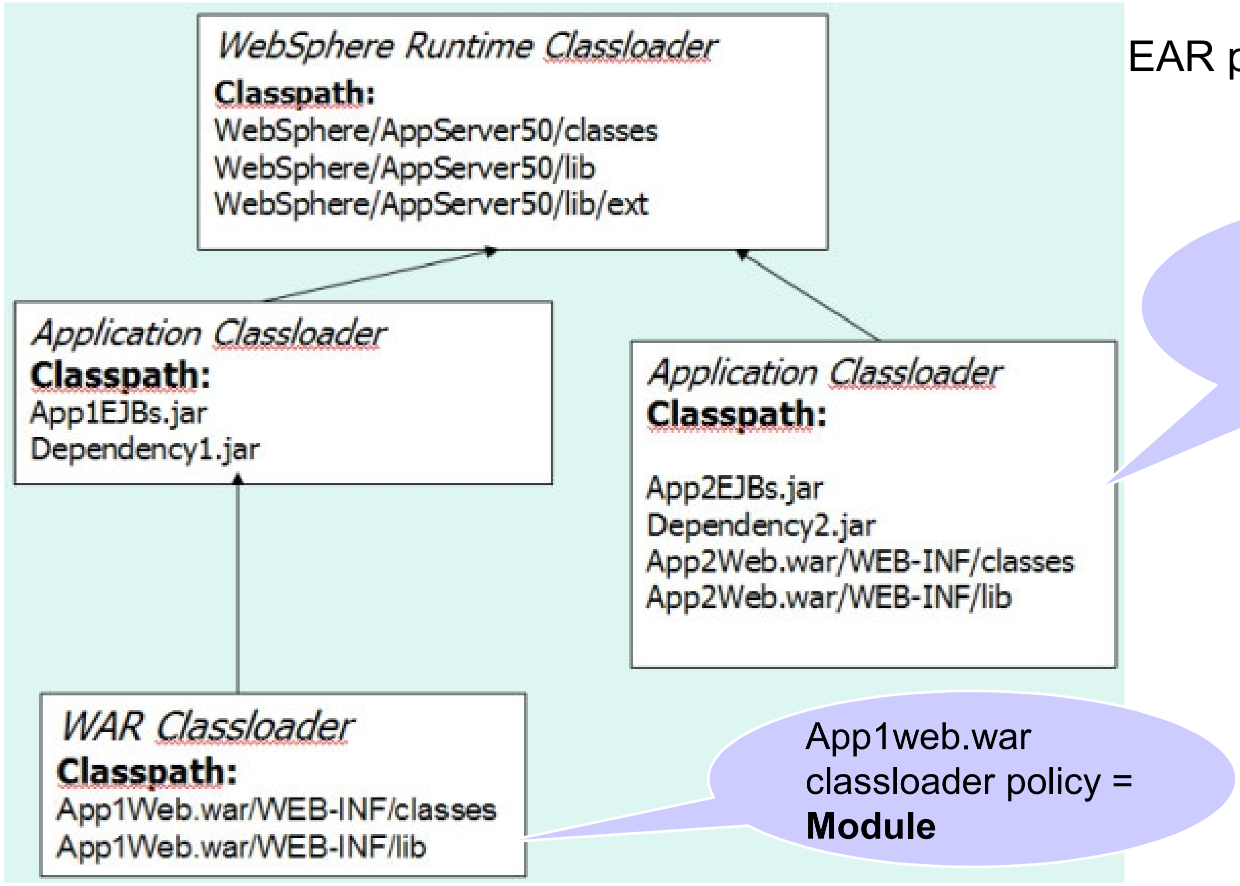
Parent_first

Parent_last

Enterprise Applications > *application_name* > Manage Modules > *web_module*

Application Class loader Policy: Multiple

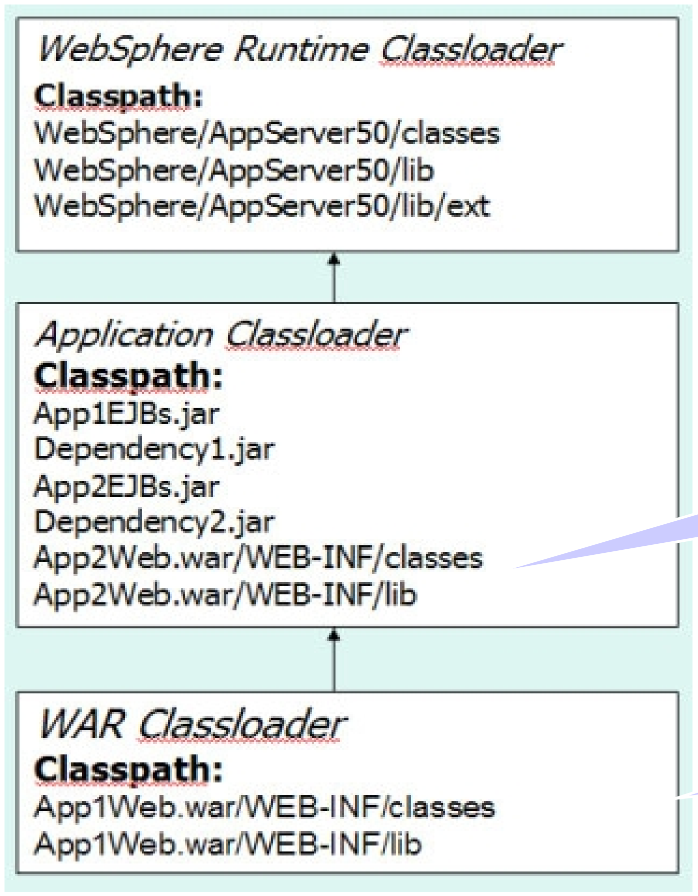
- Each application is loaded by its own class loader
- Classes within an EAR cannot reference classes in another EAR


WebSphere Runtime Classloader
Classpath:
WebSphere/AppServer50/classes
WebSphere/AppServer50/lib
WebSphere/AppServer50/lib/ext
Application Classloader
Classpath:
App1EJBs.jar
Dependency1.jar
Application Classloader
Classpath:
App2EJBs.jar
Dependency2.jar
App2Web.war/WEB-INF/classes
App2Web.war/WEB-INF/lib
WAR Classloader
Classpath:
App1Web.war/WEB-INF/classes
App1Web.war/WEB-INF/lib
App1web.war classloader policy = **Module**

EAR policy = **Multiple**

App2Web.war classloader policy = **Application**

Application Class loader Policy - Single


WebSphere Runtime Classloader
Classpath:
WebSphere/AppServer50/classes
WebSphere/AppServer50/lib
WebSphere/AppServer50/lib/ext
Application Classloader
Classpath:
App1EJBs.jar
Dependency1.jar
App2EJBs.jar
Dependency2.jar
App2Web.war/WEB-INF/classes
App2Web.war/WEB-INF/lib
WAR Classloader
Classpath:
App1Web.war/WEB-INF/classes
App1Web.war/WEB-INF/lib
App2Web.war classloader policy = **Application**
App1Web.war classloader policy = **Module**

- All applications are loaded by single class loader.
- Each application EJBs, Dependent JARs can reference other classes in other applications.

Shared Library

- Shared libraries enable you to package application code outside the scope of an EAR and have the code visible to either all applications on a server or to specific applications on a server

- Shared libraries are defined in the administrative console under:

  Environment -> Shared library

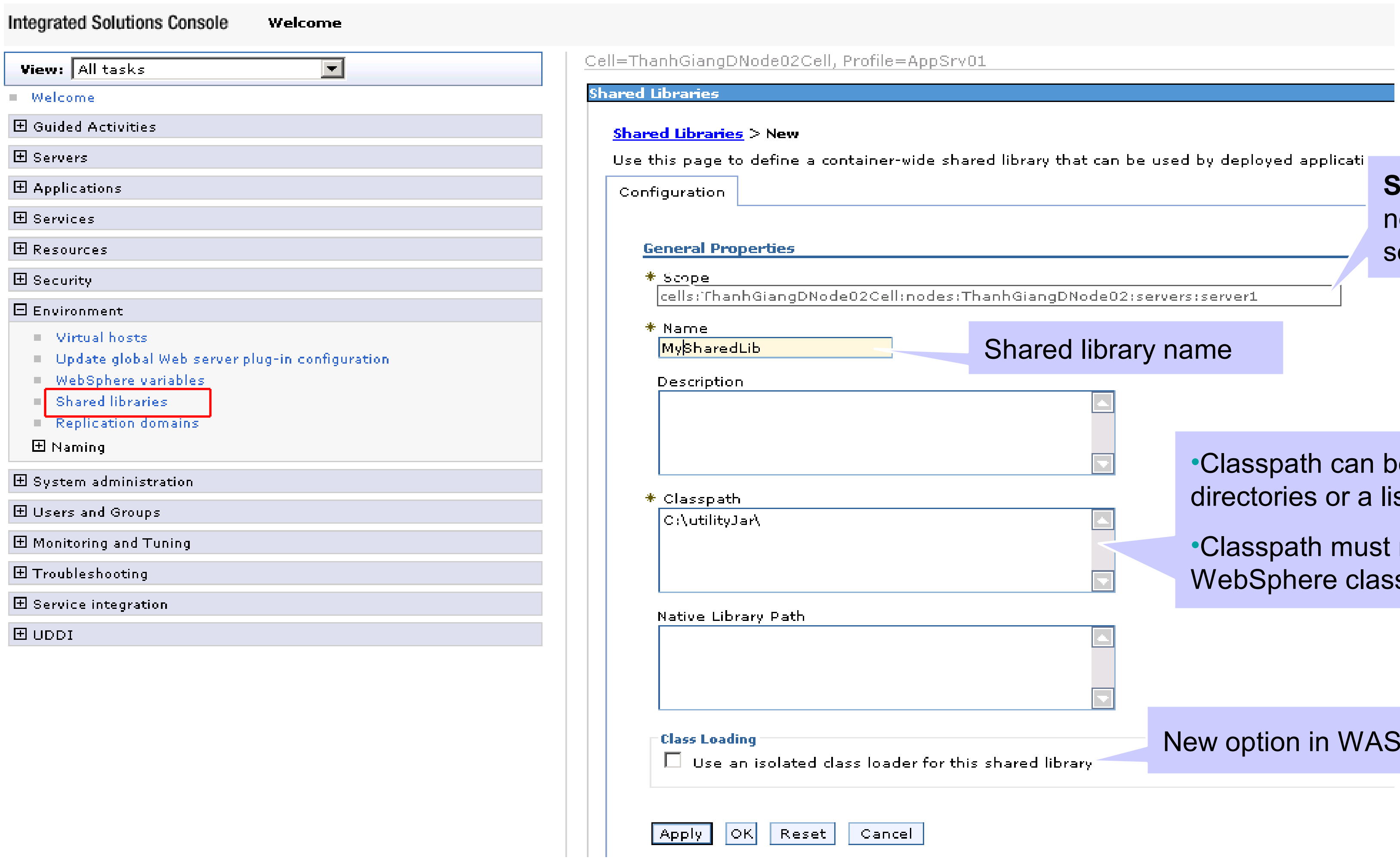
- Shared libraries can be defined at the Cell, Node, Cluster or server scope.

- You **MUST** associate the shared library with a server or application in order to use it.

- When associating the shared library with application server the **ExtJarClassLoader** loads classes represented by the shared library and makes those classes available to all applications deployed on the server.

- When associating the shared library with application the **CompoundClassLoader** loads classes represented by the shared library and makes those classes available to the application.

Shared library Configuration



Integrated Solutions Console   Welcome

View: All tasks

- Welcome
- Guided Activities
- Servers
- Applications
- Services
- Resources
- Security
- Environment
  - Virtual hosts
  - Update global Web server plug-in configuration
  - WebSphere variables
  - Shared libraries
  - Replication domains
  - Naming
- System administration
- Users and Groups
- Monitoring and Tuning
- Troubleshooting
- Service integration
- UDDI

Cell=ThanhGiangDNode02Cell, Profile=AppSrv01

Shared Libraries

Shared Libraries > New

Use this page to define a container-wide shared library that can be used by deployed applicati

Configuration

General Properties

* Scope
cells:ThanhGiangDNode02Cell:nodes:ThanhGiangDNode02:servers:server1

* Name
MySharedLib

Description

* Classpath
C:\utilityJar\

Native Library Path

Class Loading
☐ Use an isolated class loader for this shared library

Apply | OK | Reset | Cancel

Scope: cell, node, cluster, server

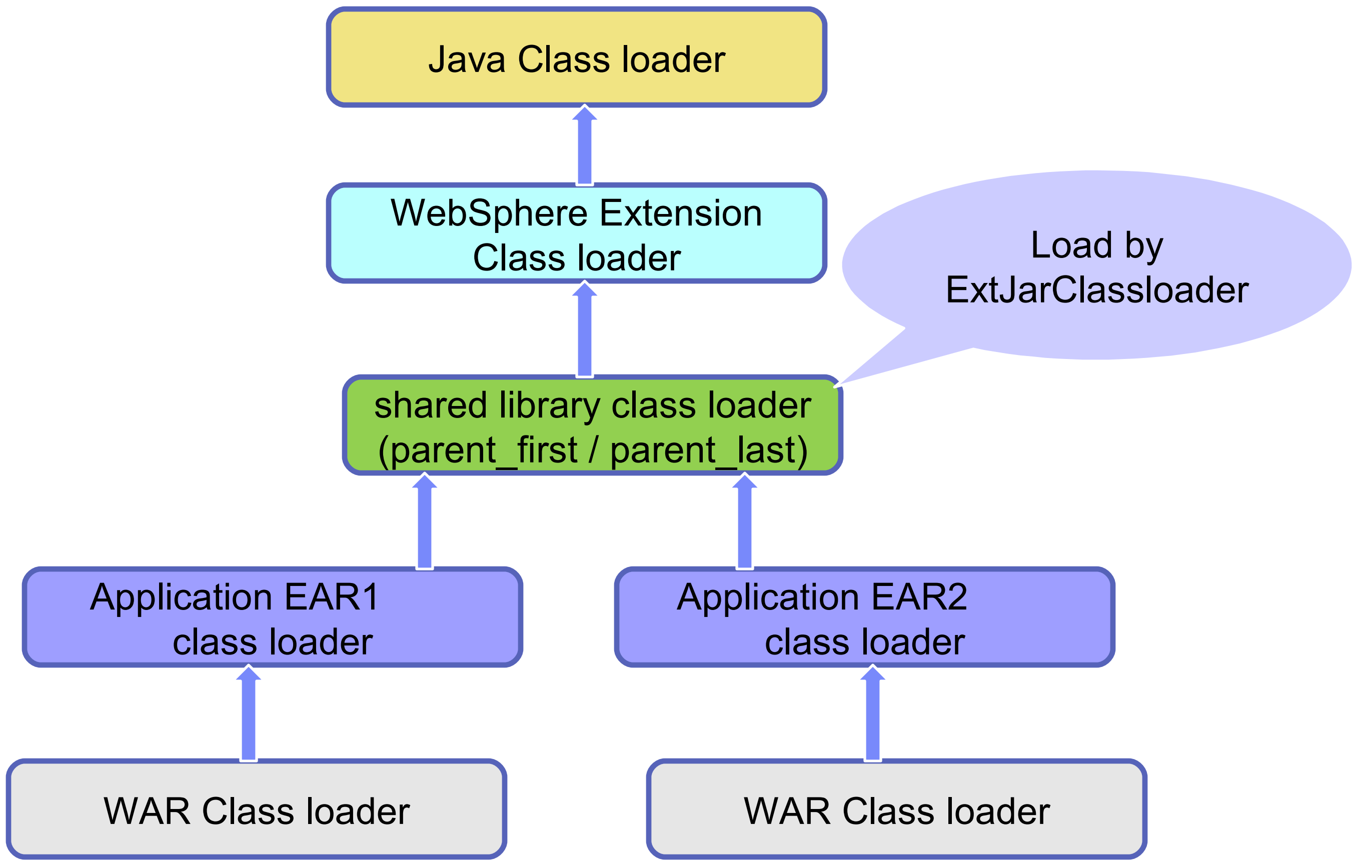
Shared library name

- Classpath can be a list of directories or a list of files.
- Classpath must not be WebSphere classpath.

New option in WAS v7.0

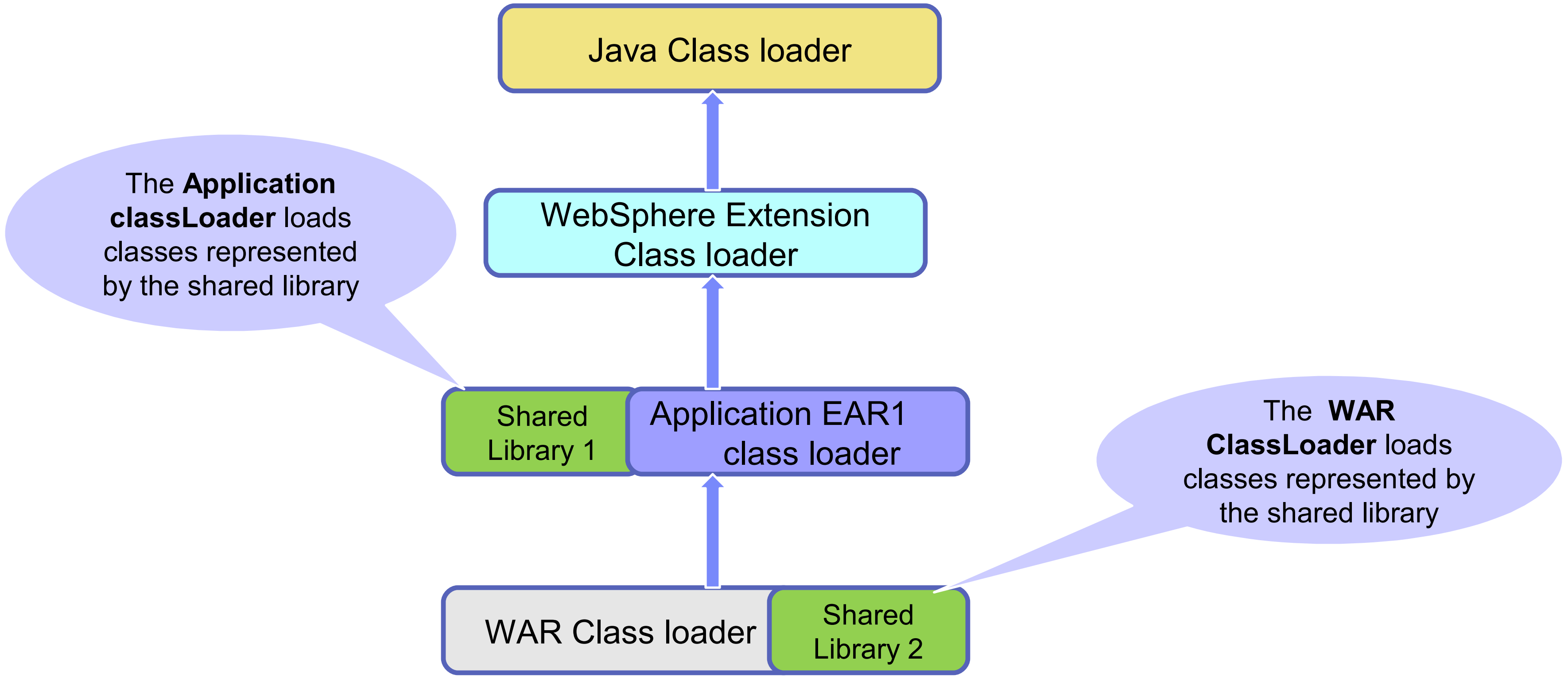
Associating shared library with server

Classes represented by the shared library are loaded by an extension class loader, making the classes visible to all applications deployed on the server.


Java Class loader
WebSphere Extension Class loader
Load by ExtJarClassloader
shared library class loader (parent_first / parent_last)
Application EAR1 class loader
Application EAR2 class loader
WAR Class loader
WAR Class loader

Associating shared library with application or module

The Classes represented by the shared library are loaded by the application's class loader, making the classes visible to the application.

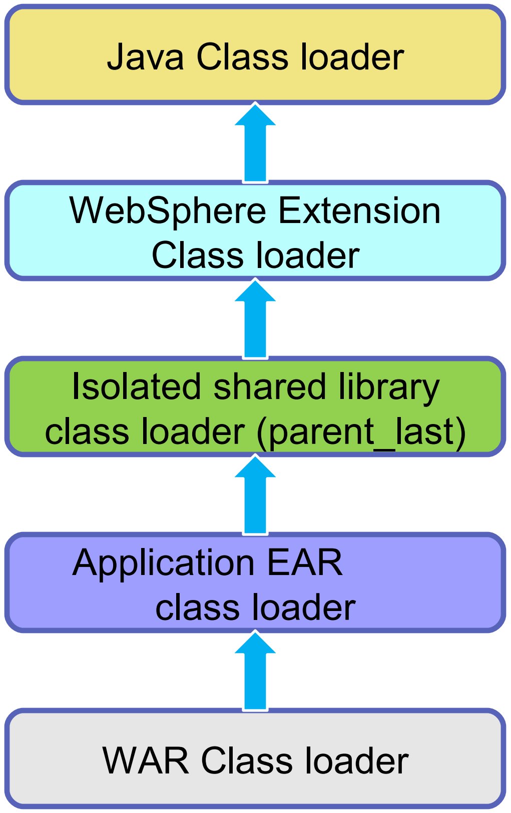

Java Class loader
WebSphere Extension Class loader
The **Application classLoader** loads classes represented by the shared library
Shared Library 1
Application EAR1 class loader
The **WAR ClassLoader** loads classes represented by the shared library
WAR Class loader
Shared Library 2

Isolated shared library – New in WAS 7.0

- Have their own class loader, enabling a single instance of the classes to be shared across the applications
- Must associate the isolated shared library with applications or Web modules.
- The option will be ignored if associating the shared library with a server.
- The class loader mode is PARENT_LAST, and it cannot be changed.
- Classes loaded with the parent class loader first (Parent first):
  - Checks whether the associated library class loaders can load the class.
  - Checks whether its parent class loader can load the class.
  - Checks whether the application or WAR module class loader can load the class.

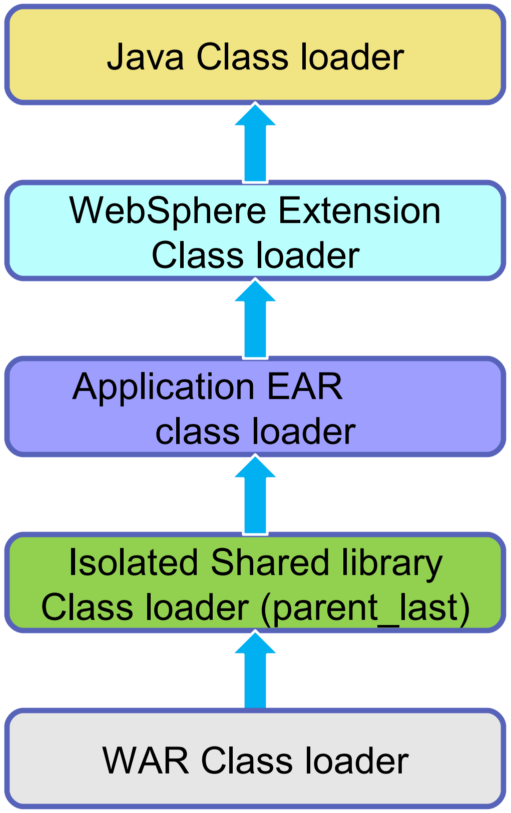
- Classes loaded with the local class loader first (Parent last):
  - Checks whether the application or WAR module class loader can load the class.
  - Checks whether the associated library class loaders can load the class.
  - Checks whether its parent class loader can load the class.

Associating isolated shared library with application EAR


Java Class loader
WebSphere Extension Class loader
Isolated shared library class loader (parent_last)
Application EAR class loader
WAR Class loader

Note: The class loader for loading shared library is the parent class loader of the EAR class loader .

Associating isolated shared library with Web Module


Java Class loader
WebSphere Extension Class loader
Application EAR class loader
Isolated Shared library Class loader (parent_last)
WAR Class loader

Note: The class loader for loading shared library is the parent class loader of the WAR class loader .

Common Issues

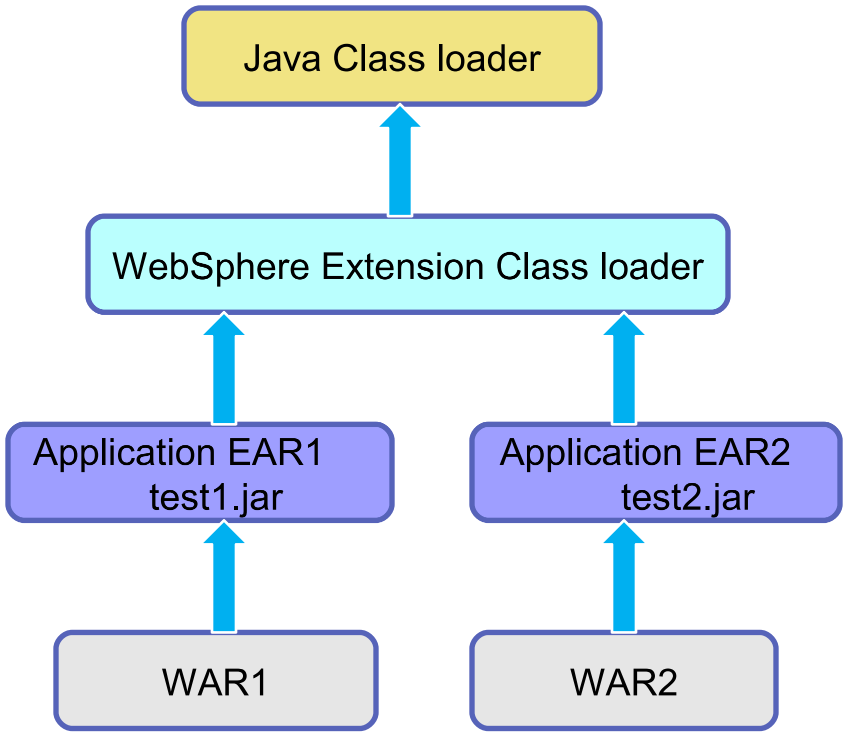
ClassNotFoundException

Scenario:

- The test1.jar is located within the root of an EAR1 file.
- The test2.jar is located within the root of an EAR2 file.
- Test2.jar depends on test1.jar.
- EAR policy: Multiple

Cause:

- The ClassNotfoundException is thrown because the test1.jar file is only visible to EAR1 but not EAR2.


Java Class loader
WebSphere Extension Class loader
Application EAR1 test1.jar
Application EAR2 test2.jar
WAR1
WAR2

ClassNotFoundException

Solution:

Create a shared library to hold the test1.jar file and associate the shared library with both EAR1 and EAR2.

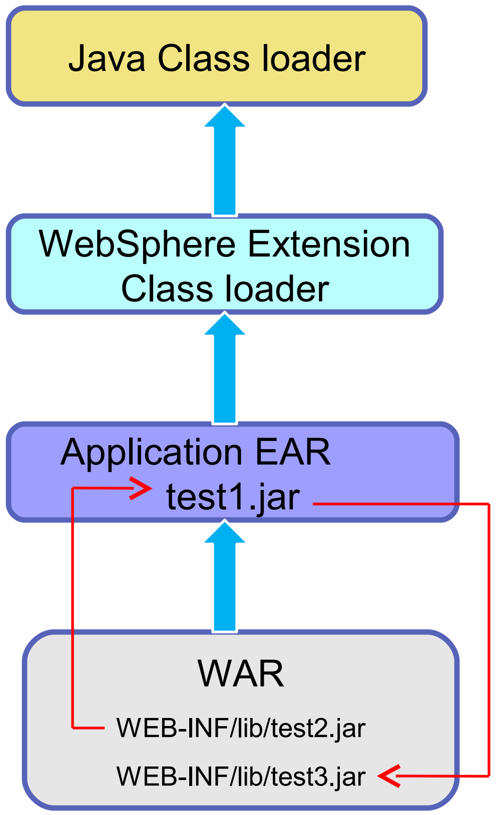
NoClassDefFoundError/ClassNotFoundException

Scenario:

- The test1.jar is located at the root of the EAR file.
- The test2.jar and test3.jar are located within a WAR file.
- Test2.jar depends on test1.jar and test1.jar depends on test3.jar
- EAR policy: Multiple
- EAR mode: parent_first

Cause:

- ClassNotFoundException or NoClassDefFoundError is thrown for class packaged in test3.jar.
- The class loader from which test1.jar was loaded is not able to find test3.jar since it cannot delegate to the child ClassLoader to find and load the class.



NoClassDefFoundError/ClassNotFoundException

Solution:

Place the dependent jars, test3.jar, in the same classloader as test1.jar so that both jar files are loaded by the same class loader and visible to each other.

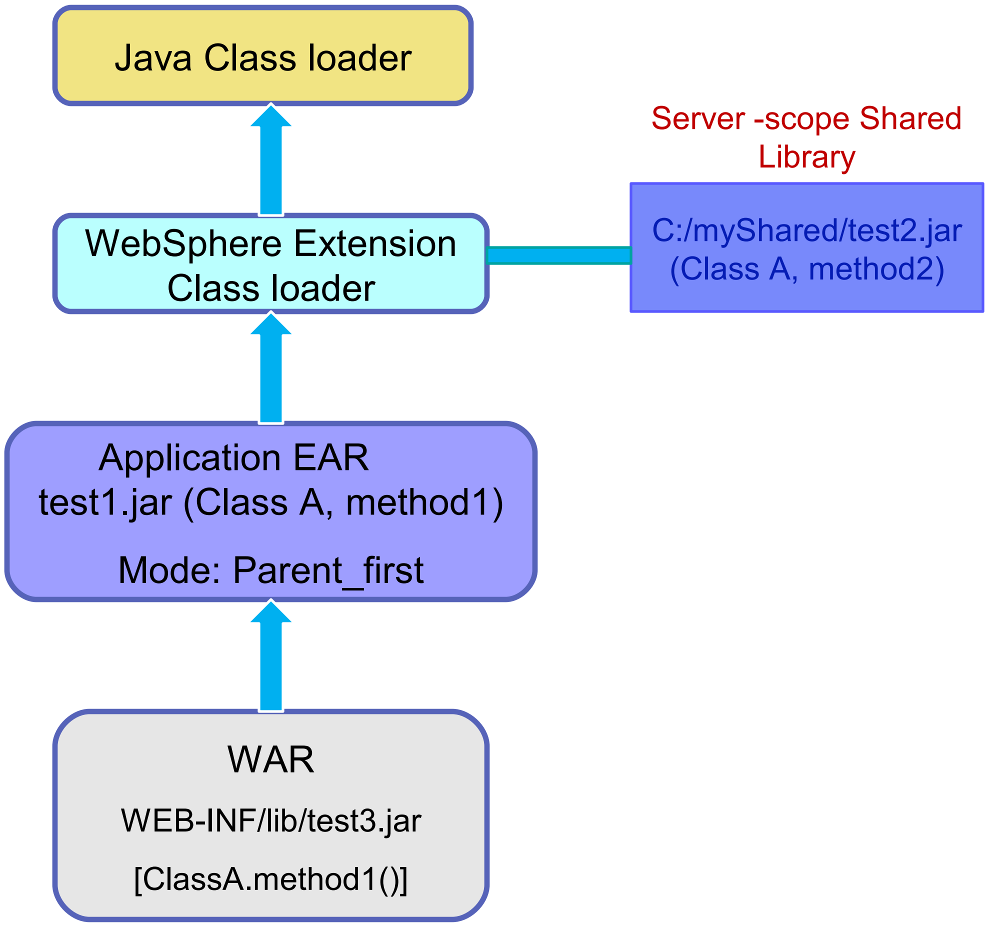
NoSuchMethodError

Scenario:

- Test1.jar is within an EAR file.
- Test2.jar is within a shared library associated with application server.
- Test1.jar and test2.jar contain different versions of the same class files.
- Test3.jar depends on test1.jar. It made method call ClassA.method1().
- Application delegation mode is PARENT_FIRST.

Cause:

NoSuchMethodError is thrown because a wrong version from Test2.jar is being picked up first and it does not contains method1.


Java Class loader
Server -scope Shared Library
WebSphere Extension Class loader
C:/myShared/test2.jar (Class A, method2)
Application EAR test1.jar (Class A, method1)
Mode: Parent_first
WAR
WEB-INF/lib/test3.jar
[ClassA.method1()]

NoSuchMethodError

Solution:

- Set the application class loader mode to parent_last so the correct version from the EAR file to be picked up first.
- Or remove the duplicate class (wrong version) from the shared library.

ClassCastException:

(Source object is not an instance of the target class)

Scenario:

The code below generates a ClassCastException:

```
Object o = new Integer(0);
String s = (String) o;     // generates ClassCastException when casting the object o to
                           // the String
```

Cause:

A ClassCastException is thrown by Java when code attempts to cast an object to a subclass of which it is not an instance.

Solution:

Use the class loader viewer or class loader trace to find out where the two classes are loaded from and by what class loader. The developer uses this information from the trace to examine the source code and determine if the source object is an instance of the target class or why objects of these classes are being cast to each other.

Summary

- Class loader hierarchy and how class loader works
- Class loader settings: Class loader mode and policy and how to configure them
- Shared library configurations
- Isolated shared library
- How to resolve common class loader issues

Appendix

ClassCastException:
(Class A incompatible with Class B)

Scenario:
- Test1.jar is within both EAR1 and EAR2.

Problem:
- ClassCastException: Class A incompatible with Class B is thrown during EJB call.

Explanation:
- The Class A from test1.jar is visible on the classpaths of more than one class loader.

- Modify the application code to output the class loader of cast target class, the class loader of the cast source object class, and the context class loader immediately before the point of failure. Example code is below:

- // For a cast such as "(TargetClass)sourceObject", add the
  // following lines before the cast statement:
  ```
  ClassLoader ctxCL = Thread.currentThread().getContextClassLoader();
  ClassLoader tcCL = TargetClass.class.getClassLoader();
  ClassLoader soCL = sourceObject.getClass().getClassLoader();
  System.out.println("ctxCl=" + ctxCL);
  System.out.println("tcCl=" + tcCL);
  System.out.println("soCl=" + soCL);
  ```

- The output is written to systemOut.log and trace.log as follow:
  ```
  ctxCl= compoundClassloader@xxxxx for loading EAR1
  tcCl = compoundClassloader@xxxxx for loading EAR1
  soCl= compoundClassloader@yyyyy for loading EAR2
  ```

- The source and target classes are loaded by two different class loaders so if one class from one class loader is trying to cast to an interface loaded by another class loader the ClassCastException is thrown.

ClassCastException

Solution:

Deploy exactly one copy of the classes used by both applications where the classes are loaded by the same class loader by moving the Test1.jar file out both EARs to an isolated shared library, and then associate the shared library to both applications.

References

- Class loaders from the information Center:
http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-zos&topic=crun_classload

- Redbook: WebSphere Application Server V6.1: Class loader problem determination
http://www.redbooks.ibm.com/redpapers/pdfs/redp4307.pdf

- Best Practices for Using Common Application Files
http://www-1.ibm.com/support/docview.wss?uid=swg27006159

- J2EE Class Loading Demystified
http://www.ibm.com/developerworks/websphere/library/techarticles/0112_deboer/deboer.html

Additional WebSphere Product Resources

- Learn about upcoming WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at: http://www.ibm.com/software/websphere/support/supp_tech.html
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at: http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere Community: http://www.websphereusergroup.org
- Access key product show-me demos and tutorials by visiting IBM® Education Assistant: http://www.ibm.com/software/info/education/assistant
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically: http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My Notifications emails: http://www.ibm.com/software/support/einfo.html

Connect with us!

1. **Get notified on upcoming webcasts**
   Send an e-mail to wsehelp@us.ibm.com with subject line “wste subscribe” to get a list of mailing lists and to subscribe
2. **Tell us what you want to learn**
   Send us suggestions for future topics or improvements about our webcasts to wsehelp@us.ibm.com
3. **Be connected!**
   Connect with us on Facebook
   Connect with us on Twitter

Questions and Answers