Deploying applications

Overview of Deployment

- Two views of deployment:
  - Developers
    - Development environment
    - Single stand-alone machine
    - Deploy over and over again at will during the testing phase
  - Administrators
    - Production environment
    - Multiple WebLogic Server instances or clusters
    - Deploy infrequently during maintenance schedules

Deployment Methods

- WLS supports following deployment methods:
  - Weblogic Console deployment
  - Command-line deployment (WLST, *weblogic.Deployer* class, *wldeploy* Ant task)
  - Auto deployment folder

- Applications and EJBs can be deployed in an:
  - Archived file `(.ear`, `.war`, `.jar`)
  - Exploded (open) directory format

Weblogic Console Deployment

- Deploying with the console allows full administrator control:
  - Installation of an application from a location of your choice
  - Manual configuration of the application name
  - Targeting the application to individual servers or clusters, or both
  - Configuring the application without targeting it
  - Activating deployment when desired

Deployment with `weblogic.Deployer`

- Prepare and deploy a new application:

```
java weblogic.Deployer -adminurl t3://adminserver:7001
  -username myuser -password welcome1 -name HRServices
  -source /usr/HRServices.ear -targets serverA -deploy
```

- Redeploy an application:

```
java weblogic.Deployer -adminurl t3://adminserver:7001
  -username myuser -password welcome1 -name HRServices
  -redeploy
```

- Undeploy an application:

```
java weblogic.Deployer -adminurl t3://adminserver:7001
  -username myuser -password welcome1 -name HRServices
  -undeploy
```

- To list all deployed applications:

```
java weblogic.Deployer -adminurl t3://localhost:7001
  -username myuser -password welcome1 -listapps
```

Deploying an Application with WLST

- Deploy an application (`deployapp.py`):

```
##
# WLST script for Deploying Java EE Application #
##

# Connect to the server
print 'Connecting to server  .... '
connect('weblogic','welcome1','t3://localhost:7001')

appname = "mbeanlister"
applocation = "c:/domains/MedRecDomain/apps/mbeanlister"

# Start deploy
print 'Deploying application ' + appname
deploy(appname, applocation, targets='myserver',
       planPath='c:/myapps/plan/plan.xml')
print 'Done Deploying the application '+ appname
exit()
```

Autodeployment

- By default, the autodeployment feature is enabled only if the domain is *not* running in production mode.
- When enabled:
  - The administration server monitors its “autodeploy” folder for new, updated, or removed applications
  - Applications are targeted only to the administration server
  - Developers can quickly test or experiment with an application
- **`<WL_HOME>/user_projects/domains/`*`domain`*`/autodeploy`**

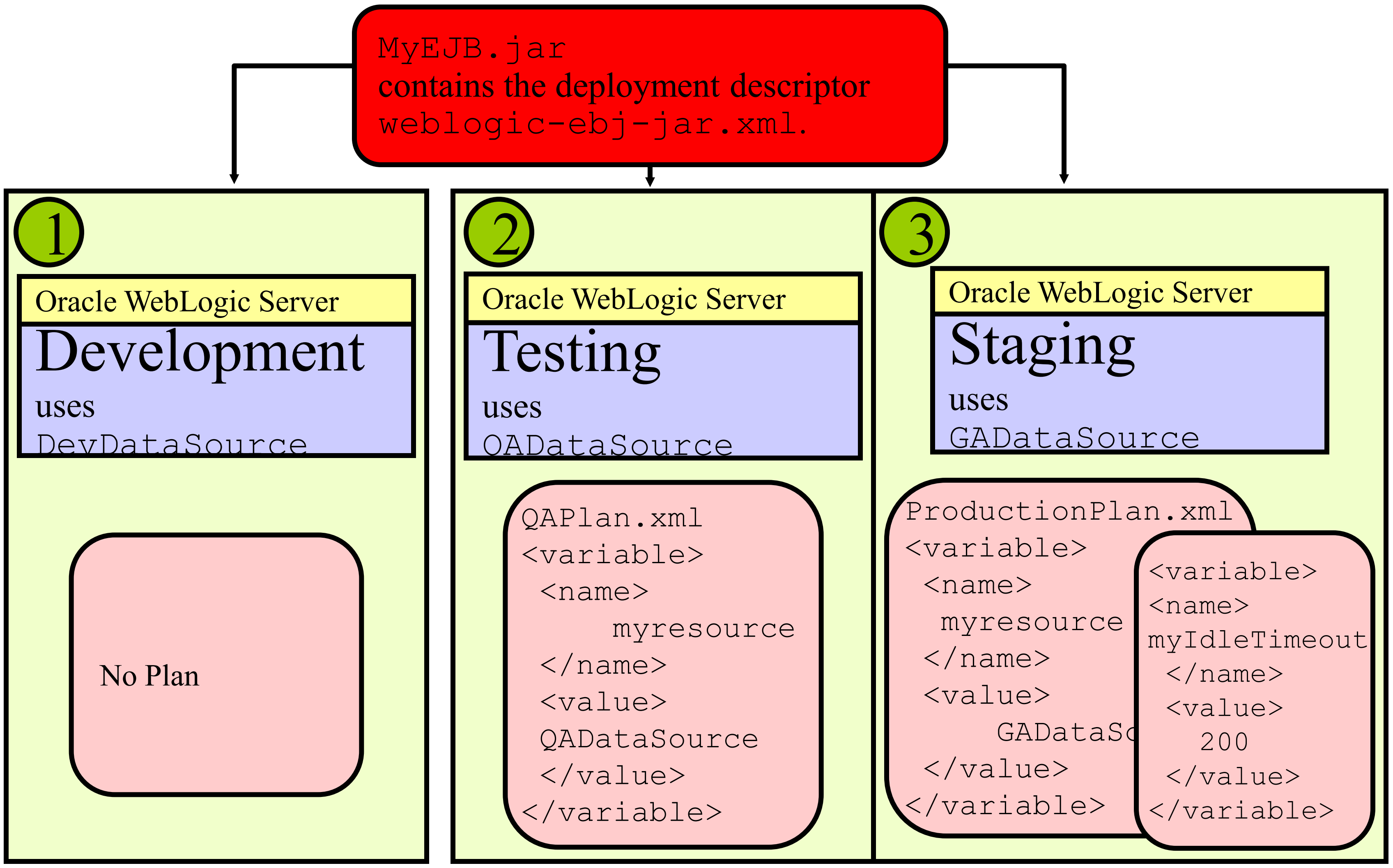
FastSwap and On-Demand Deployment

- WebLogic’s FastSwap feature is:
  - Enabled using the WebLogic deployment descriptors
  - Available only if the domain is *not* running in production mode
  - Applicable only to Web applications that are *not* archived
- When enabled:
  - WebLogic automatically reloads the modified Java class files within applications
  - Developers can perform iterative development without an explicit redeployment
- On-demand deployment:
  - **`weblogic.xml:`**
    `<fast-swap>true</fast-swap>`

Deployment Plan

- Java EE deployment plan:
  - Is an optional XML file associated with an application
  - Resides outside an application archive
  - Sets or overrides the values in the Java EE deployment descriptors
  - Allows a single application to be easily customized to multiple deployment environments

Deployment Plan

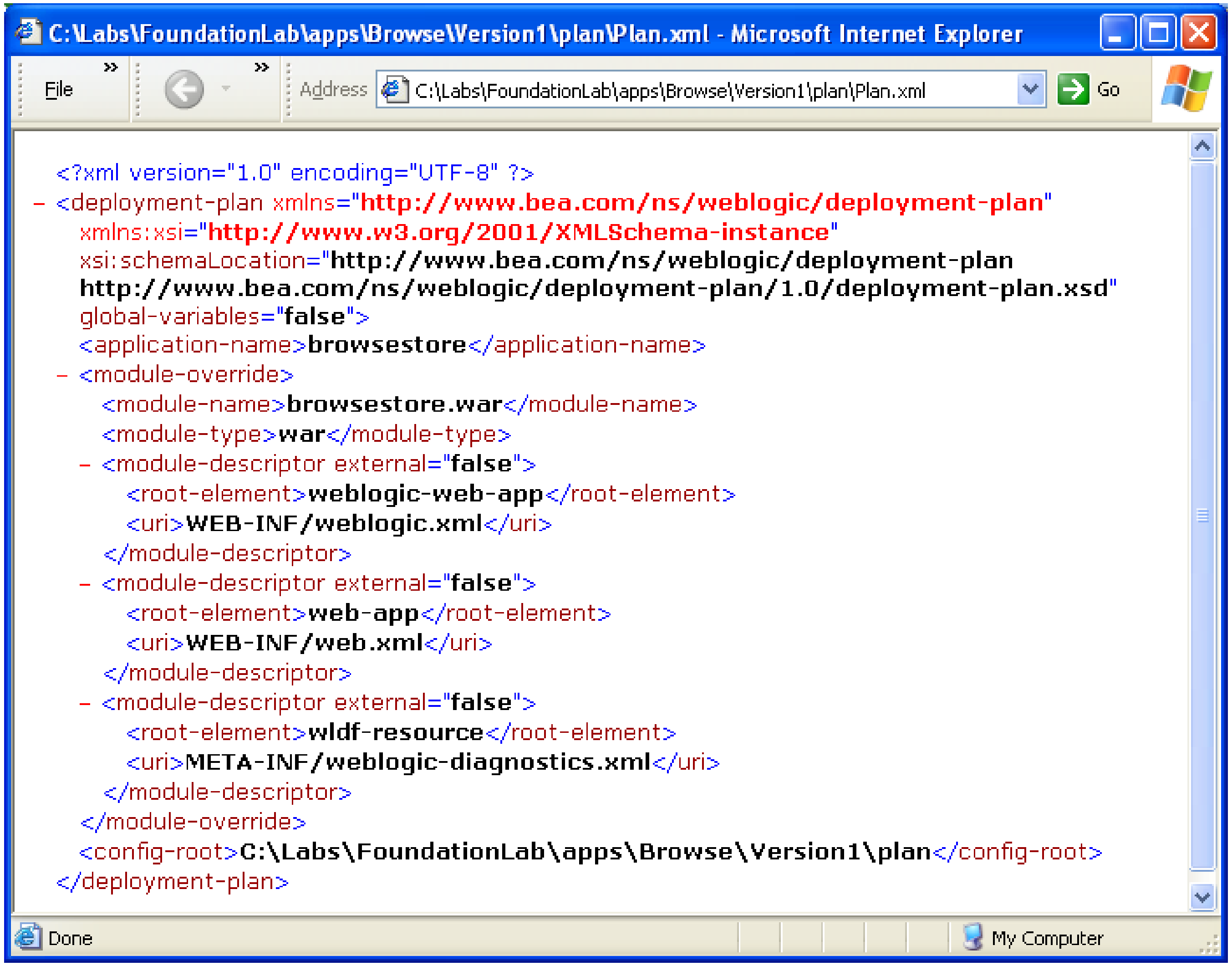


```
QAPlan.xml
<variable>
 <name>
      myresource
 </name>
 <value>
 QADataSource
 </value>
</variable>
```

```
ProductionPlan.xml
<variable>
 <name>
  myresource
 </name>
 <value>
      GADataSo
 </value>
</variable>
```

```
<variable>
<name>
myIdleTimeout
 </name>
 <value>
   200
 </value>
</variable>
```

Sample Deployment Plan


C:\Labs\FoundationLab\apps\Browse\Version1\plan\Plan.xml - Microsoft Internet Explorer
File
Address
C:\Labs\FoundationLab\apps\Browse\Version1\plan\Plan.xml
Go
Done
My Computer

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <deployment-plan xmlns="http://www.bea.com/ns/weblogic/deployment-plan"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.bea.com/ns/weblogic/deployment-plan
    http://www.bea.com/ns/weblogic/deployment-plan/1.0/deployment-plan.xsd"
    global-variables="false">
    <application-name>browsestore</application-name>
  - <module-override>
      <module-name>browsestore.war</module-name>
      <module-type>war</module-type>
    - <module-descriptor external="false">
        <root-element>weblogic-web-app</root-element>
        <uri>WEB-INF/weblogic.xml</uri>
      </module-descriptor>
    - <module-descriptor external="false">
        <root-element>web-app</root-element>
        <uri>WEB-INF/web.xml</uri>
      </module-descriptor>
    - <module-descriptor external="false">
        <root-element>wldf-resource</root-element>
        <uri>META-INF/weblogic-diagnostics.xml</uri>
      </module-descriptor>
    </module-override>
    <config-root>C:\Labs\FoundationLab\apps\Browse\Version1\plan</config-root>
  </deployment-plan>
```

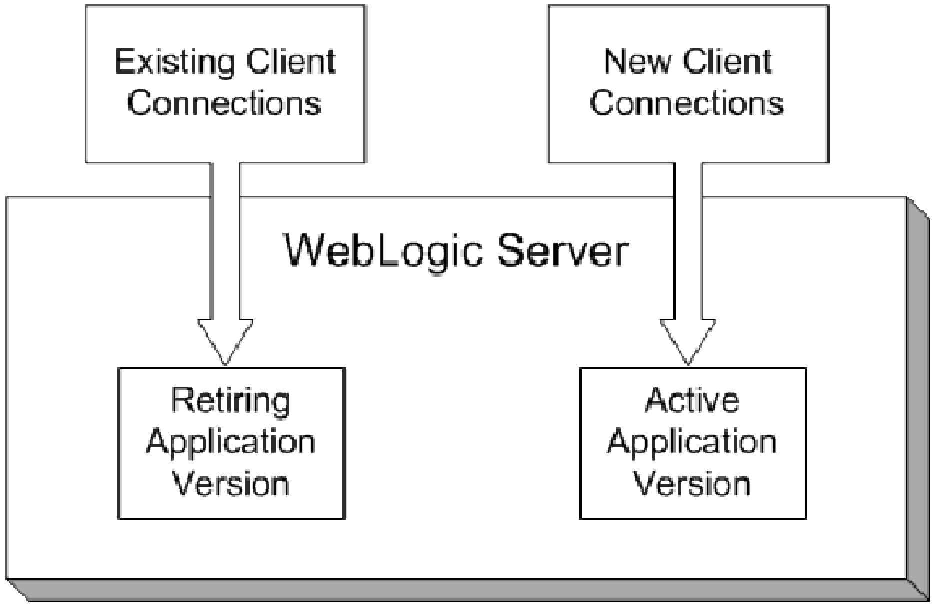
Creating Deployment Plan

- Tools for creating a deployment plan:
  - Development tool—for example, JDeveloper or Eclipse
  - `weblogic.PlanGenerator`
  - Administration Console
- Goals for creating a deployment plan:
  - To expose the external resource requirements of the application as variables in the deployment plan
  - To expose additional configurable properties, such as tuning parameters as variables in the deployment plan

Production Redeployment

Side by Side Deployment

- Multiple application versions can co-exist
  - New client requests are routed to active version;
    Existing client requests can finish up with existing version
- Automatic Retirement Policy: Graceful, Timeout
- Test application version before opening up for business
- Rollback to previous application version
- Two versions of the application can be active at any given point of time


Existing Client Connections
New Client Connections
WebLogic Server
Retiring Application Version
Active Application Version

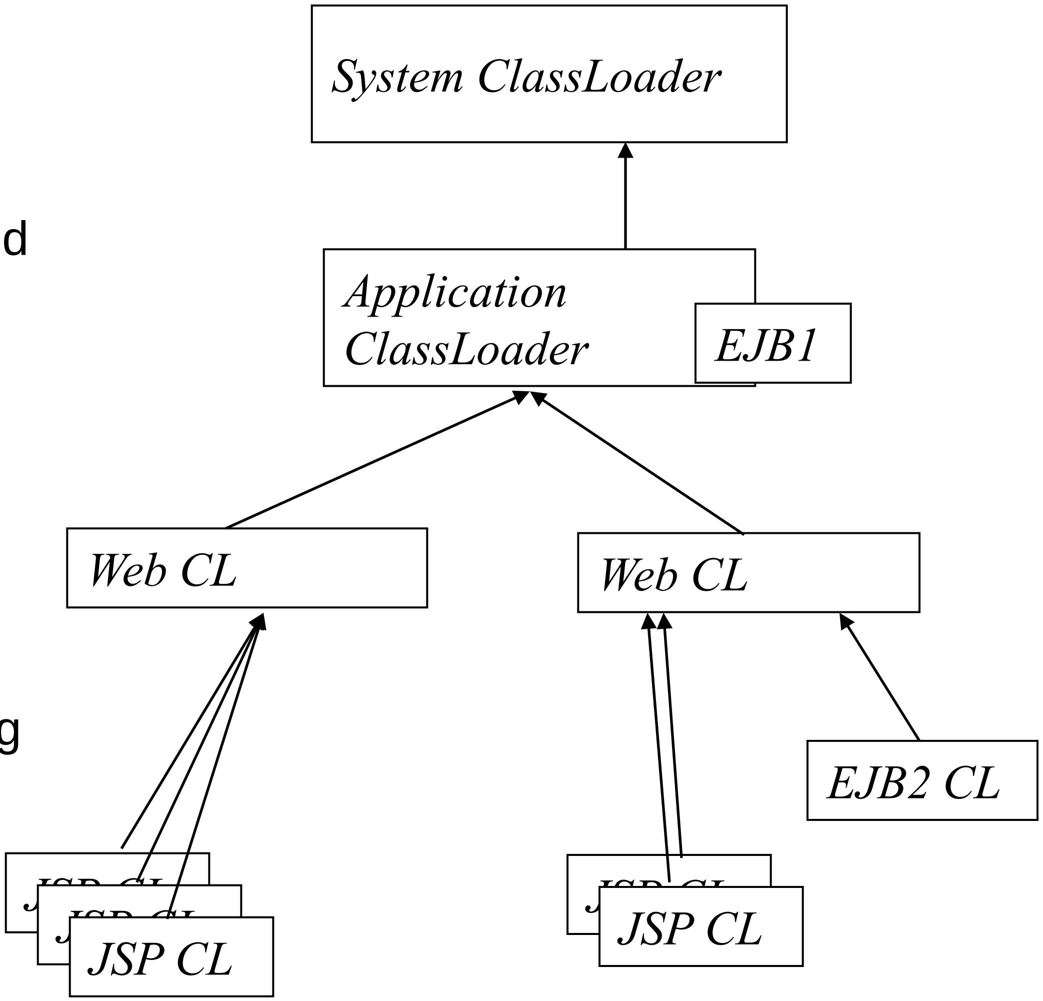
Production Redeployment

- To support the production redeployment strategy, Oracle WebLogic Server now recognizes a unique version string entry in the Enterprise `MANIFEST` file.
- When a redeployment operation is requested, Oracle WebLogic Server checks the version string to determine whether to deploy a new version of the application.
- Production redeployment is performed automatically if:
  - An application supports production redeployment
  - Its deployment configuration is updated with changes to resource bindings
- This occurs even if no version string is specified in the application's manifest file.

ORACLE

In-place Partial Redeployment

- Classloader hierarchy enables redeployment flexibility
- Web applications can be redeployed without redeploying the EJB tier
- The JSP class has its own classloader, which is a child of the Web application classloader. This allows JSPs to be individually reloaded.
- Newer versions of application modules such as EJBs can be deployed while the server is running
- Custom classloader hierarchies provide even more flexibility


System ClassLoader
Application ClassLoader
EJB1
Web CL
Web CL
JSP CL
JSP CL
EJB2 CL