17

Packaging and Deploying Java EE Applications

MentorLabs℠

Objectives

After completing this lesson, you should be able to do the following:

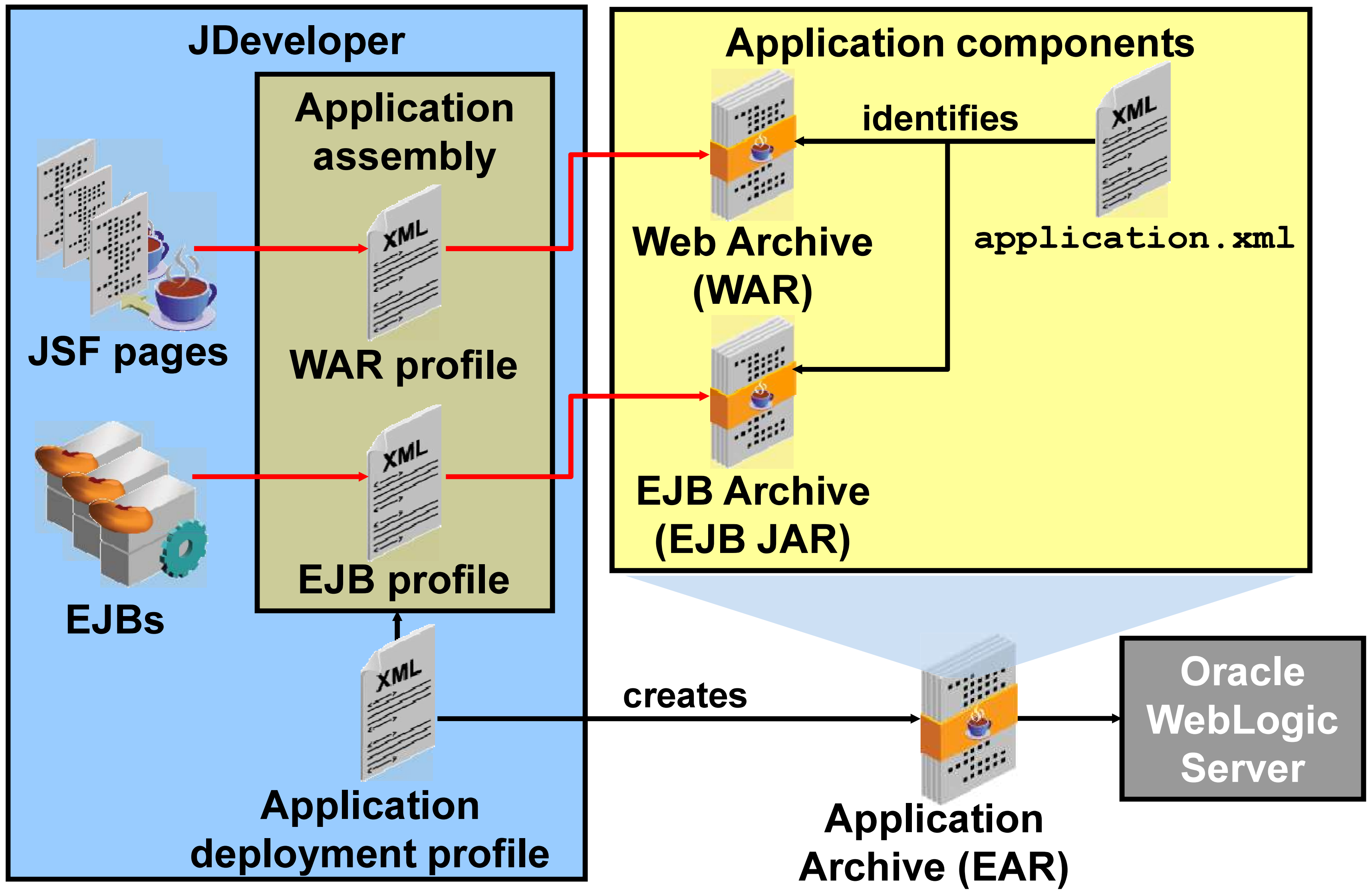
- Deploy Java EE applications to the WebLogic server environment
- Deploy applications by using :
  - Console
  - Command line
  - JDeveloper

Deploying Java EE Applications

Oracle WebLogic Server 10*g* release 3 (10.3):

- Is a Java EE 5–compliant container that
  - Provides a Java EE–compliant infrastructure for deployment
  - Supports deploying, undeploying, and redeploying Java EE applications and modules
  - Supports Java SE 6 specification
- Implements the Java EE Application Deployment API (JSR-88)
- Supports deployment with the following tools:
  - `weblogic.Deployer`
  - Administration Console
  - WLST

Packaging Business-Tier Components



Packaging Web Applications

1. Arrange resources in a prescribed directory structure.
2. Develop the `web.xml` deployment descriptor (or copy as required).
3. Develop the `weblogic.xml` deployment descriptor (WLS-specific).
4. Archive Web App into a `.war` file using JAR.
5. Deploy Web App onto WLS.
6. Configure Web App with the WLS Administration Console.

Web Application Structure

- The structure of Web applications is defined by the Servlet specification.
- A Web application can be either:
  - An archived file (`.war` file)
  - An expanded directory structure

Directory/Files	Description
MyWebApplication	Document root of Web application
META-INF	Information for archive tools (manifest)
WEB-INF	Private files that will not be served to clients
classes	Server-side classes such as servlets and applet
lib	.jar files used by Web app
web.xml	Web app deployment descriptor
weblogic.xml	WLS-specific deployment descriptor

MentorLabs℠

Configuring Web Applications

Web applications are configured through deployment descriptors `web.xml` and `weblogic.xml` which:

- Define run-time environment
- Map URLs to servlets and JSPs
- Define application defaults such as welcome and error pages
- Specify Java EE security constraints
- Define work managers for applications
- Set the `context-root` for the application

What Is `web.xml`?

The `web.xml` file is a deployment descriptor for configuring:

- Servlets and JSP registration
- Servlet initialization parameters
- JSP tag libraries
- MIME type mappings
- Welcome file list
- Error pages
- Security constraints and roles
- Resources
- EJB references

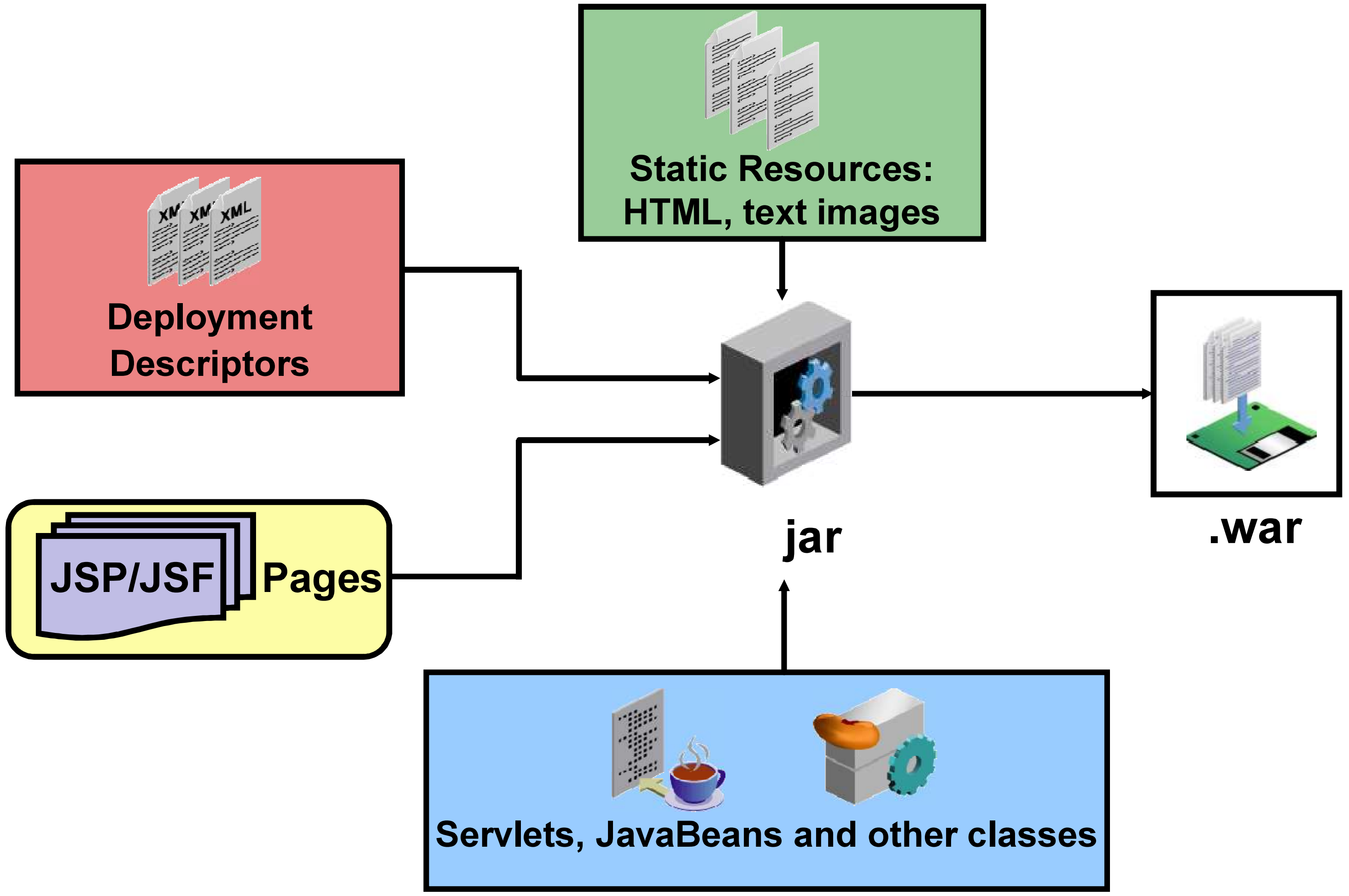
What Is `weblogic.xml`?

The `weblogic.xml` is a WebLogic Server-specific deployment descriptor for configuring:

- JSP properties
- JNDI mappings
- security role mappings
- HTTP session parameters
- Work managers
- Context root
- Virtual directory mappings
- Logging parameters
- Library modules

Web Application Archive

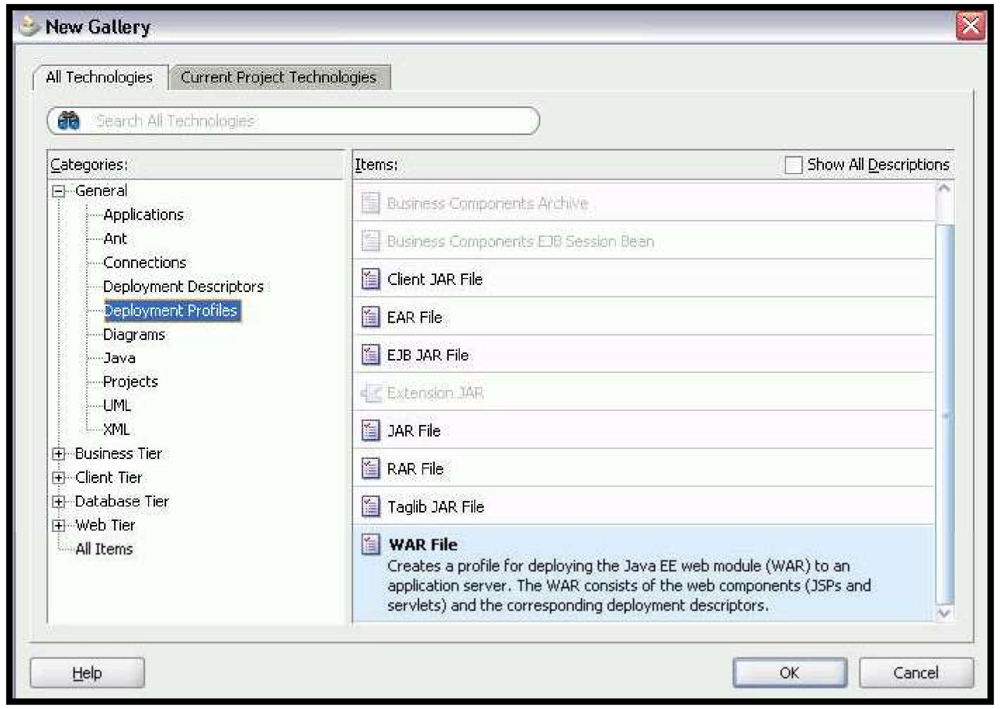
Web archives are created using the `jar` utility:


Static Resources: HTML, text images
XML
Deployment Descriptors
JSP/JSF
Pages
jar
.war
Servlets, JavaBeans and other classes

Creating Web Application Archives

To create a Web Archive (WAR) file by using JDeveloper, perform the following steps:

1. Right-click the Web Project node and select New.
2. Double-click the WAR file item in the Deployment Profiles category.
3. Configure and save profile settings.
4. Right-click the Web profile and select "Deploy to WAR file."



WAR File
Creates a profile for deploying the Java EE web module (WAR) to an application server. The WAR consists of the web components (JSPs and servlets) and the corresponding deployment descriptors.

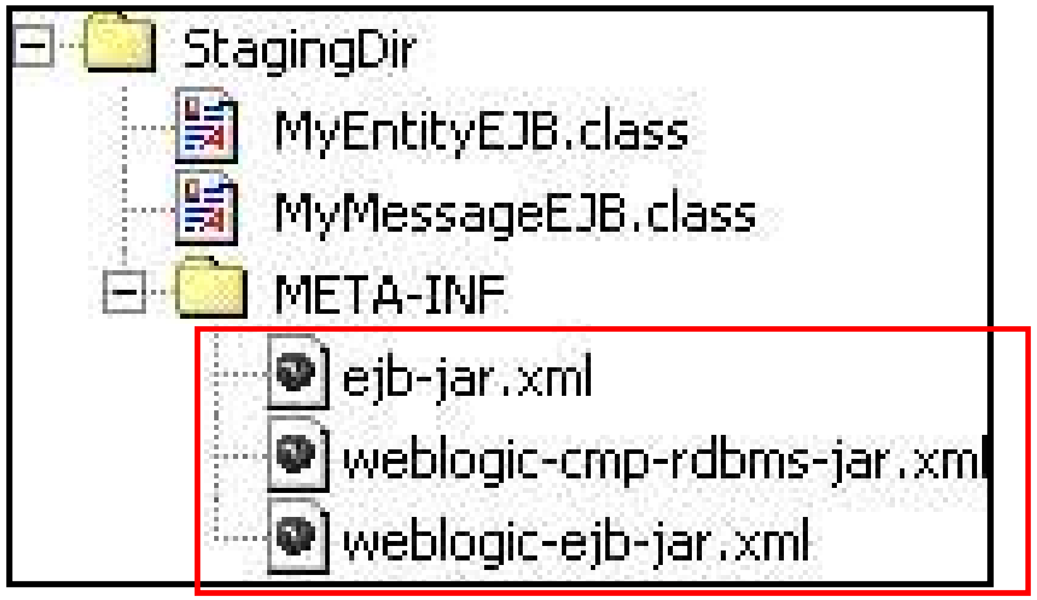
Quiz

What is the command-line interface that you can use to automate domain configuration tasks, application deployment configuration, and deployment operations in WebLogic Server?

1. `weblogic.Deployer`
2. WebLogic Scripting Tool
3. `jar` utility

EJB Application Directory Structure

- EJB components come packaged in JAR files.
- EJBs are configured by modifying deployment descriptors.



```
StagingDir
    MyEntityEJB.class
    MyMessageEJB.class
    META-INF
        ejb-jar.xml
        weblogic-cmp-rdbms-jar.xml
        weblogic-ejb-jar.xml
```

Java EE Enterprise Application (EAR)

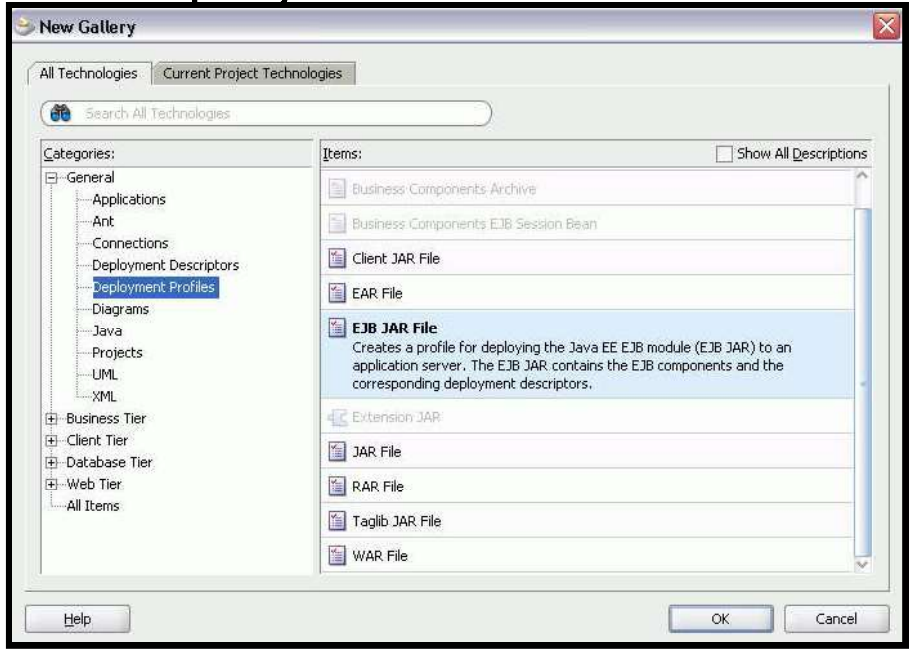
Example of the directory structure of an enterprise application:

Directory/File	Description
MyEnterpriseApplication	Document root of enterprise application
META-INF	META-INF directory
application.xml	Enterprise application deployment descriptor
weblogic-application	WLS Enterprise application deployment descriptor
myEJBs1.jar	EJB module
myEJBs2.jar	Another EJB module
myJavaClasses1.jar	Java module
myJavaClasses2.jar	Another Java module
myWebApp1.war	Web application module
myWebApp2.war	Another Web application module

Creating EJB Archives

To create an EJB JAR file by using Oracle JDeveloper, perform the following steps:

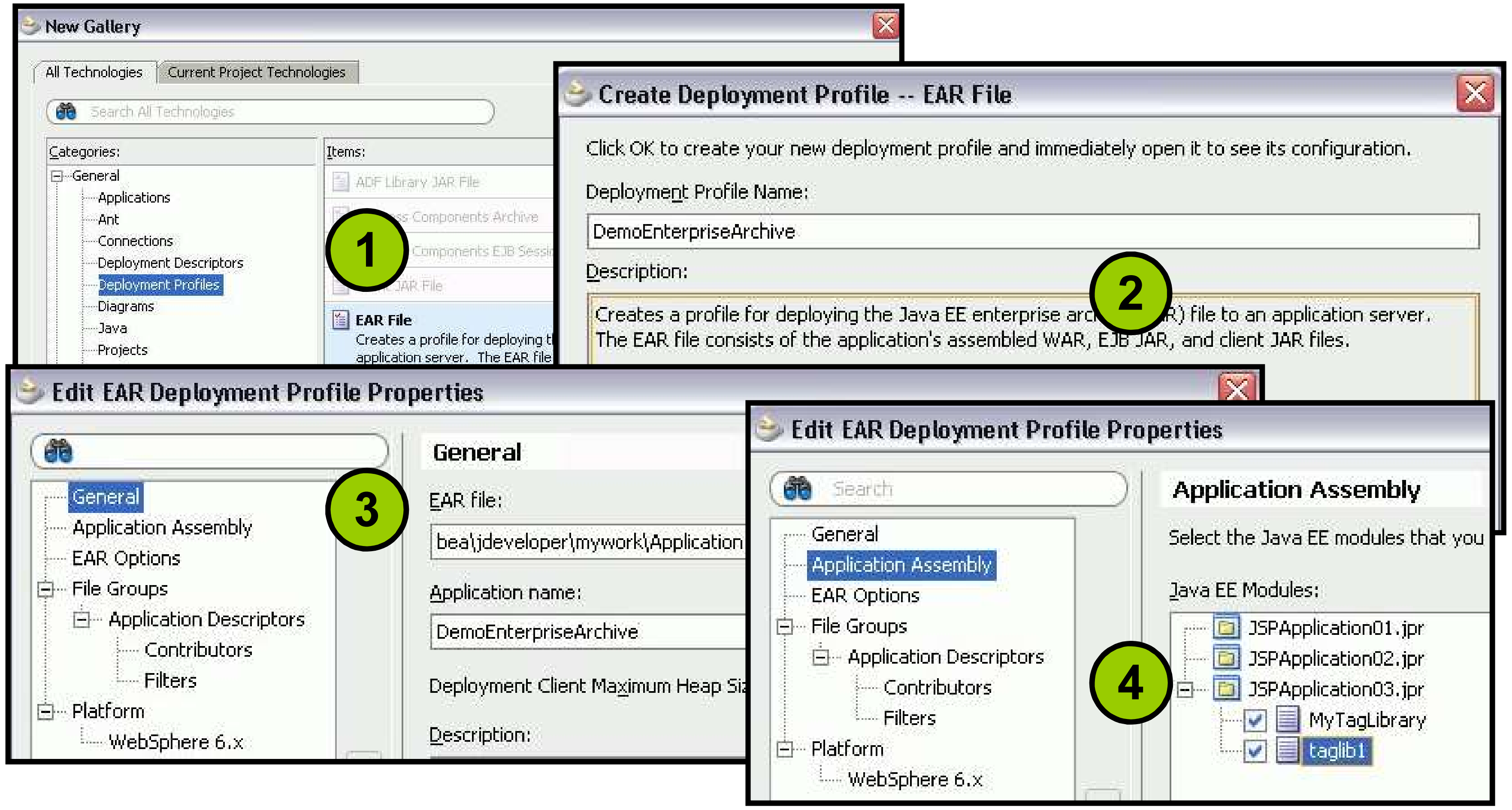
1. Right-click the EJB Model Project node and select New.
2. Double-click the EJB JAR file item in the Deployment Profiles category.
3. Configure and save profile settings.
4. Right-click the EJB-JAR profile and select "Deploy to JAR/EAR file."



Creates a profile for deploying the Java EE EJB module (EJB JAR) to an application server. The EJB JAR contains the EJB components and the corresponding deployment descriptors.

Creating Enterprise Archives

To create the EAR file by Using Oracle JDeveloper:



Create Deployment Profile -- EAR File

Click OK to create your new deployment profile and immediately open it to see its configuration.

Deployment Profile Name: DemoEnterpriseArchive

Description: Creates a profile for deploying the Java EE enterprise arc… R) file to an application server. The EAR file consists of the application's assembled WAR, EJB JAR, and client JAR files.

General

EAR file: bea\jdeveloper\mywork\Application

Application name: DemoEnterpriseArchive

Deployment Client Maximum Heap Si

Description:

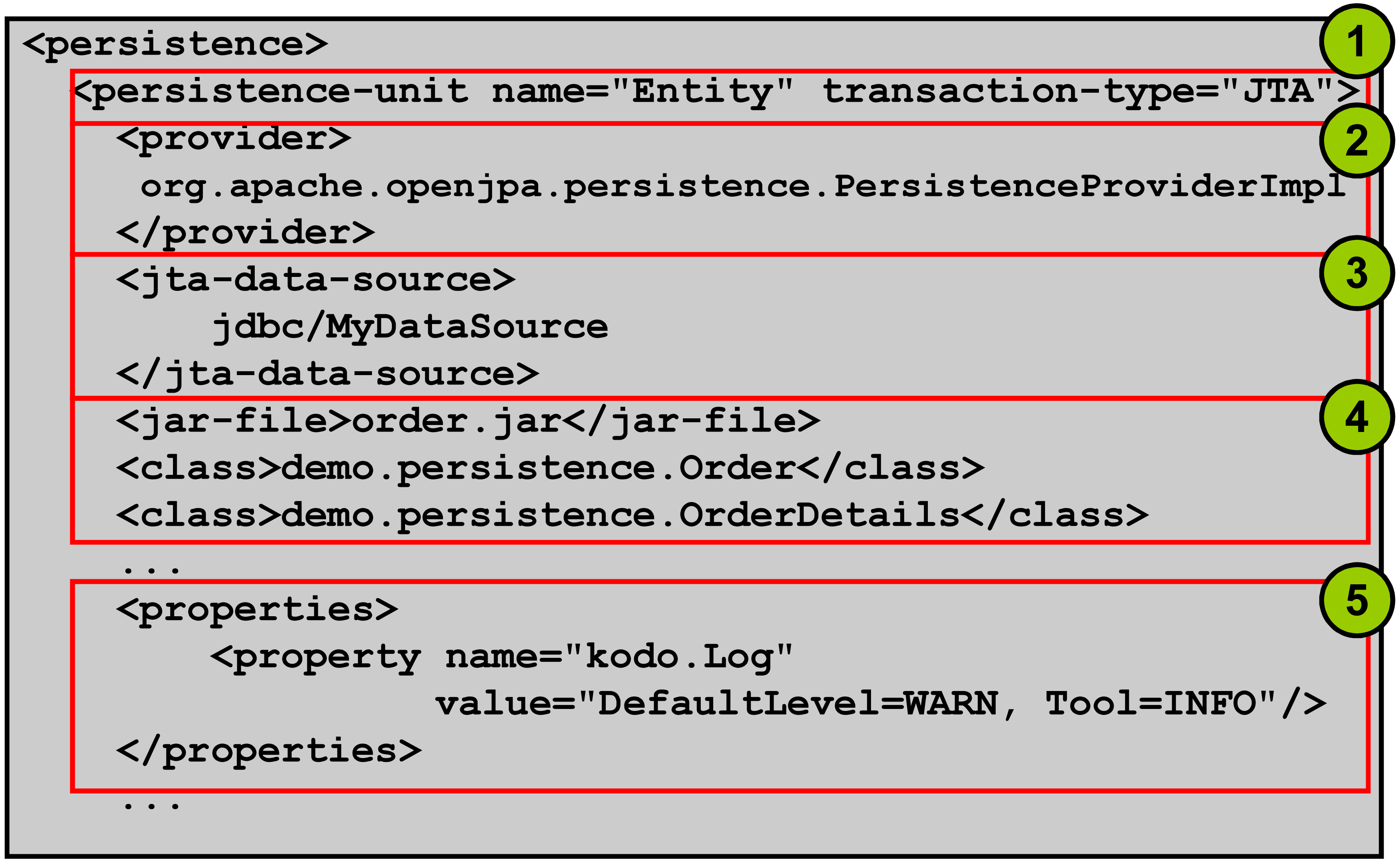
Application Assembly

Select the Java EE modules that you

Deploying Entities

- Expose the persistence module in an EJB-JAR, WAR, or JAR file depending on its execution environment.
- Configure the `persistence.xml` file to define the persistence unit name and to specify the following information:
  - Specify a data source.
  - Specify the transaction type.
  - Specify vendor-specific extensions.

Persistence.xml File



```
<persistence>
   <persistence-unit name="Entity" transaction-type="JTA">
     <provider>
      org.apache.openjpa.persistence.PersistenceProviderImpl
     </provider>
     <jta-data-source>
         jdbc/MyDataSource
     </jta-data-source>
     <jar-file>order.jar</jar-file>
     <class>demo.persistence.Order</class>
     <class>demo.persistence.OrderDetails</class>
      ...
     <properties>
         <property name="kodo.Log"
                   value="DefaultLevel=WARN, Tool=INFO"/>
     </properties>
      ...
```

Configuring Oracle WebLogic Server–Specific Features

Configure enterprisewide Oracle WebLogic Server–specific features with the `weblogic-application.xml` deployment descriptor:

- XML parsers
- XML entity mappings
- JDBC data sources
- JMS connection factories and destinations
- Security realms

Deploying with the

Oracle WebLogic Server Control enables:

- Applications to be deployed from
  - EAR files
  - WAR files
  - EJB JAR files
- Applications to be undeployed
- Applications to be redeployed
- Creation and editing of deployment plans during deployment

Deploying with Oracle JDeveloper

To deploy an application with JDeveloper, perform the following steps:

1. Create the deployment profile.
2. Configure the deployment profile.
3. Create an application server connection to the target environment.
4. Right-click the application and select “Deploy to <application_server_connection_name>.”

What Is Ant?

- Ant is:
  - A Java build tool similar to GNU’s make utility
  - Written in Java and is open source
  - Developed and maintained by the Apache organization
  - Downloadable from and documented at `http://jakarta.apache.org/ant`
- Ant consists of built-in tasks for:
  - Compiling and executing Java applications
  - Building archives
  - File and directory manipulation

Ant Build Files

- Ant searches for a build file to determine what work should be performed.
- By default, Ant looks for a file named `build.xml` in the current directory.
- Example of specifying a build file other than `build.xml`:

```
ant.bat -buildfile MyBuildFile.xml
ant.bat -buildfile /demo/BuildApplication.xml
```

A Sample `build.xml` File

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test Project" default="run" basedir=".">
   <property name="sourceDir" value="source"/>

   <target name="compile">
      <javac srcdir="${sourceDir}" destdir="classes"/>
   </target>

   <target name="run" depends="compile">
      <java classname="test.MyTester">
         <classpath>
            <pathelement path="classes"/>
         </classpath>
      </java>
   </target>
</project>
```

Creating a JAR File by Using Ant Task

- The `jar` task archives a set of files.
- The resulting archives can update existing ones or replace them.
- Use Ant if an external, scripted solution is required.
- Example of creating a Java archive file:

```
<jar jarfile="myArchive.jar" basedir="myproject/root"/>

<jar jarfile="myArchive.jar" basedir="myproject/root"
     excludes="*.html" update="yes"/>

<jar jarfile="myArchive.jar" basedir="myproject/root"
     compress="false"/>
```

Creating a WAR File by Using Ant Task

- The war task archives a set of files into the appropriate J2EE Web Application format.
- The webxml attribute defines the file to use as the deployment descriptor, web.xml.
- Use the <classes> and <lib> elements to define the application's class files.
- Example of creating a Web archive file:

```
<war warfile="myWebApp.war" basedir="myproject/root"
     webxml="myproject/myWebApp.xml">
   <lib dir="myproject/libraries"/>
   <classes dir="myproject/classes"/>
</war>
```

Creating an EAR File by Using Ant Task

- The `ear` task archives a set of files into the appropriate Java EE Enterprise Application format.
- The `appxml` attribute defines the file to use as the deployment descriptor, `application.xml`.
- Example of creating an Enterprise archive file:

```
<ear earfile="myApp.ear" basedir="myproject/root"
     appxml="myproject/myApp.xml"
     includes="*.jar,*.war"/>
```

Deploying an Application by Using Ant Task

- The `wldeploy` task deploys an archive (EAR, WAR, and so on) to one or more servers.
- The `source` attribute defines the archive to deploy.
- The `adminurl` attribute defines the URL of the Admin Server.
- The targets attribute defines the server to which the archive is deployed.
- Example of deploying a Web Archive (WAR):

```
<wldeploy source="myApp.war" name="MyApp"
     user="x" password="y"
      adminurl="t3://localhost:7001"
     targets="adminserver" />
```

Packaging Best Practices for Production Environments

- No duplicate JAR files in EARs
- JAR files that are supplied by the system or WebLogic classpath should not be included in the EAR
- Ensure all EAR files include a `weblogic-application.xml` descriptor
- Ensure all WAR files include a `weblogic.xml` descriptor
- Ensure all EJB files include a `weblogic-ejb-jar.xml` descriptor
- Test artifacts must not be included in the EAR

Summary

In this lesson, you should have learned how to:

- Deploy Java EE applications to the WebLogic server environment
- Deploy applications by using the:
  - Console Deployment
  - Command-line Deployment
  - Deployment by using JDeveloper