SKILLBUILDERS

WebLogic 8.1 Student Guide

V1.1

WebLogic 8.1 Student Guide

SkillBuilders, Inc.
24 Salt Pond Road, Unit C-4
South Kingstown, RI 02880
www.skillbuilders.com
knowledge@skillbuilders.com

Author: Carl Wurtzel

Legal Notice

Copyrights

Trademarks

The product/service names mentioned herein are manufacturer/publisher trademarks and are used only for purposes of identification.

Microsoft, Microsoft Windows 95, Microsoft Windows 98, Microsoft Windows NT, and Microsoft Windows are registered trademarks of Microsoft Corporation. Netscape, Netscape Navigator, and the Netscape logo are registered trademarks of Netscape Communications Corporation in the United States and other countries. Oracle, Oracle8i and Oracle9i are registered trademarks of Oracle Corporation.

All other brand, product and service names are the trademarks of their respective companies or owners as mentioned herein.

Disclaimer

Information within this publication is subject to change without notice. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. The author(s) and SkillBuilders, Inc. make no representation or warranty with respect to the contents herein and specifically disclaims any implied warranties of fitness for any particular purpose. SkillBuilders, Inc. disclaims all liability for any direct, indirect, incidental or consequential, special or exemplary damages resulting from the use of the information in this document or from the use of any products described in this document.

Table of Contents

1. Introduction to WebLogic 8.1 .... 1.1

WebLogic 8.1 Overview 1.2
WebLogic 8.1 & J2EE 1.3
WLS Architecture 1.4
Servers 1.5
Domains 1.6
Clustering 1.8
Server Management 1.9
WebLogic Console 1.11
Miscellaneous New Features 1.13
Where We've Been 1.14

2. Installation & Configuration of WebLogic 8.1 .... 2.1

Acquisition and Installation 2.2
Configuration 2.3
Testing Installation 2.4
Testing the JDK 2.5
Ant 2.6
XML Spy 2.7
Where We've Been 2.8

3. Deploying Web Apps in WebLogic 8.1 .... 3.1

Deploying Servlets 3.2
Deploying JSPs 3.3
Deploying to WebLogic 3.4
Copy Jar to Server 3.5
Deploying to WebLogic 3.6

Deploying to WebLogic Examples....3.7
WebLogic Builder Tool....3.8
Using Support Classes....3.10
The Console....3.11
Server config.xml....3.13
Undeploying Web Apps....3.14
Target Servers....3.16
Where We've Been....3.17

4. Deploying Enterprise Beans in WebLogic 8.1 .... 4.1

Deploying EJBs in WL8.1....4.2
1. Create DDs....4.3
weblogic-ejb-jar.xml....4.5
Deployment Descriptors....4.7
2. Jar the Beans & DDs....4.13
3. Make Container Classes....4.14
4. Deploy the EJB Jar File....4.15
Copy Jar to Server....4.16
The Deployer Utility....4.18
Deploying via the Console....4.19
Server config.xml....4.20
Managing Deployment....4.21
Updating a Deployment....4.22
Undeploying EJBs....4.23
Stopping EJBs....4.25
Target Servers....4.26
JNDI Registry....4.27
Where We've Been....4.28

5. Managing EJB Persistence in WebLogic 8.1 .... 5.1

WebLogic and EJB Persistence....5.2
Setting Up Persistence....5.3
Data Sources....5.4
WebLogic & CMP....5.5
Persistence Type....5.6
Using Persistence Types....5.8
Use WebLogic Builder Tool....5.10
Where We've Been....5.11

6. JDBC DataSources in WebLogic 8.1 .... 6.1

Configuring Database Connections....6.2

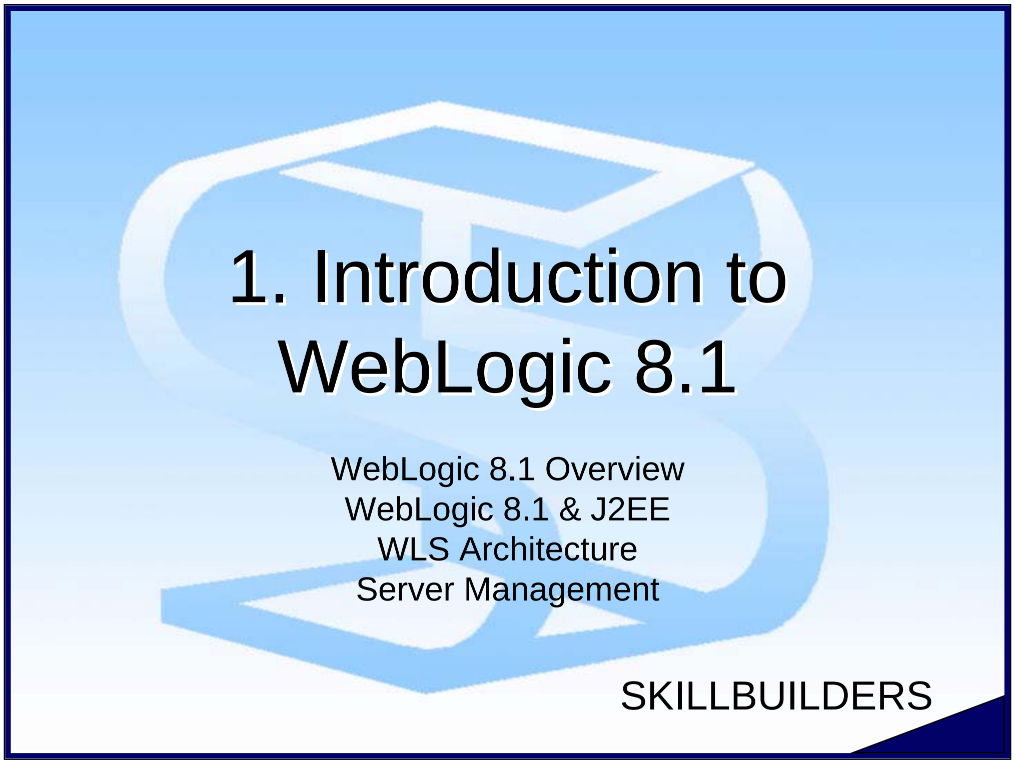
Configuring a Connection Pool..6.3
Sample Connection Pools: PointBase...6.7
Sample Connection Pools: MySql 6.8
Configuring a Data Source 6.9
Where We've Been 6.12

7. A Quick Introduction to the WebLogic 8.1 Builder Tool 7.1

Builder Tool Makes Deployment Easier 7.2
Creates DDs...7.3
A Quick Tour 7.4
EJB Attributes: Session Bean 7.5
EJB Attributes: Entity Bean 7.6
A Few Other EJB Attributes 7.7
Limitations of Builder..7.9
Deploying from Builder..7.10
Where We've Been 7.11

A. Using MySql with WebLogic 8.1 ....A.1

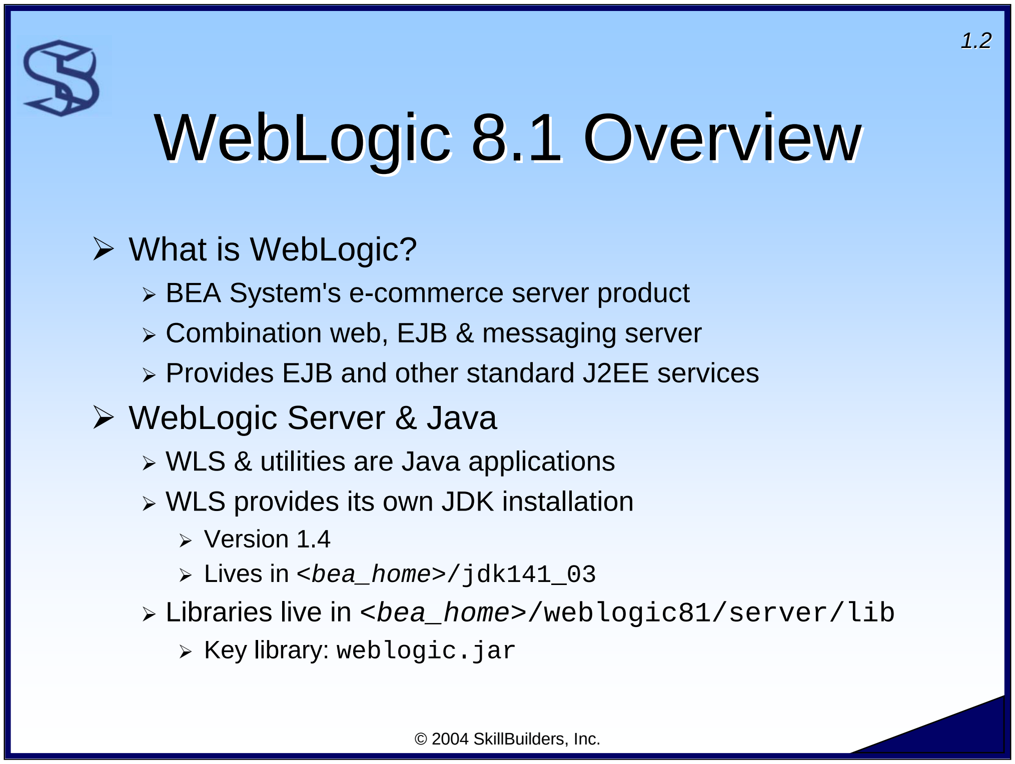
Download MySql A.2
Running MySQL...A.3
Configuring a Connection Pool...A.4
Sample Connection Pools A.7
Configuring a Data Source A.8
Where We've Been A.11



1. Introduction to WebLogic 8.1

WebLogic 8.1 Overview
WebLogic 8.1 & J2EE
WLS Architecture
Server Management

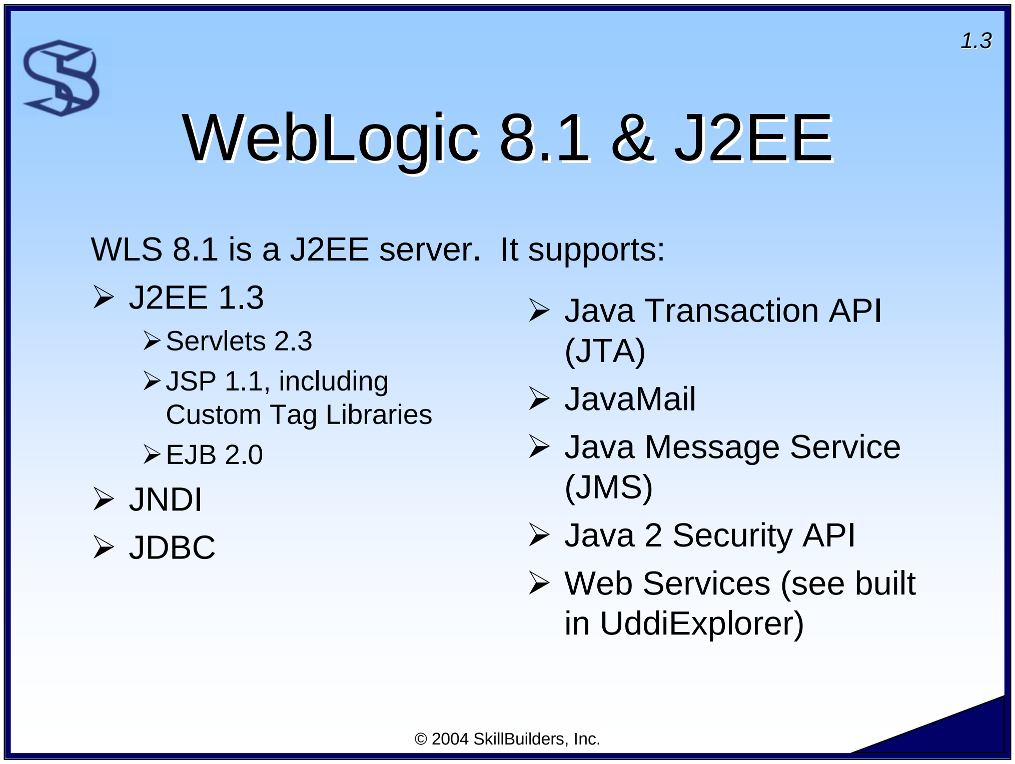
SKILLBUILDERS



1.2

WebLogic 8.1 Overview

- What is WebLogic?
  - BEA System's e-commerce server product
  - Combination web, EJB & messaging server
  - Provides EJB and other standard J2EE services
- WebLogic Server & Java
  - WLS & utilities are Java applications
  - WLS provides its own JDK installation
    - Version 1.4
    - Lives in <*bea_home*>`/jdk141_03`
  - Libraries live in <*bea_home*>`/weblogic81/server/lib`
    - Key library: `weblogic.jar`

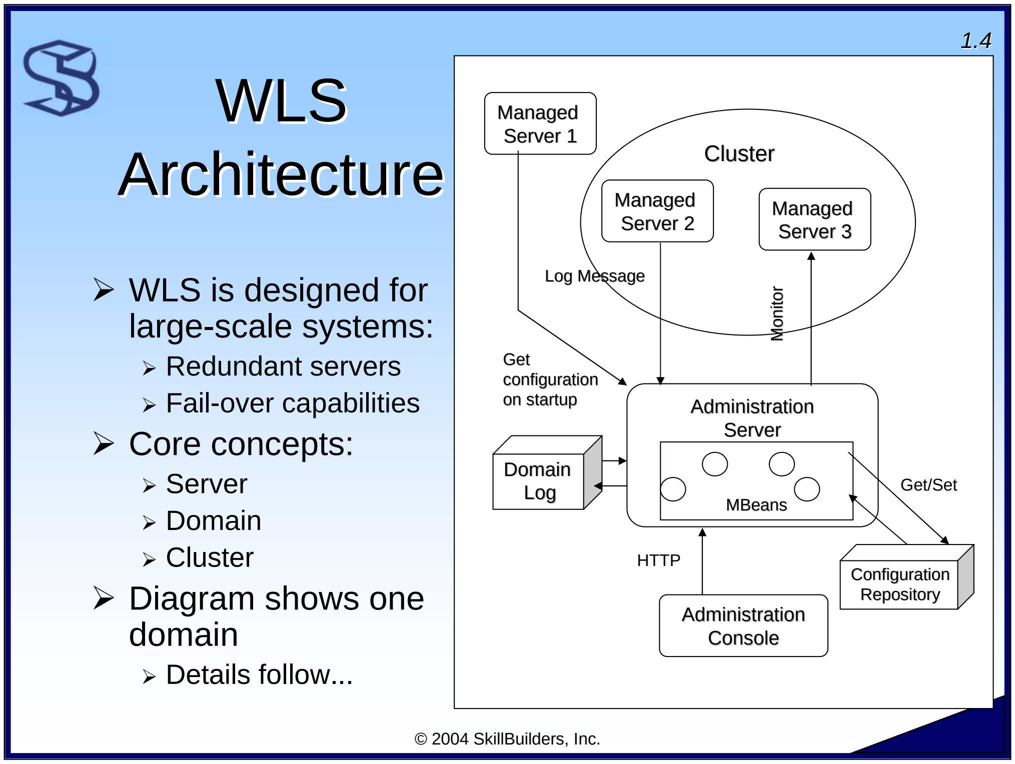


1.3

WebLogic 8.1 & J2EE

WLS 8.1 is a J2EE server. It supports:

- J2EE 1.3
  - Servlets 2.3
  - JSP 1.1, including Custom Tag Libraries
  - EJB 2.0
- JNDI
- JDBC
- Java Transaction API (JTA)
- JavaMail
- Java Message Service (JMS)
- Java 2 Security API
- Web Services (see built in UddiExplorer)



1.4

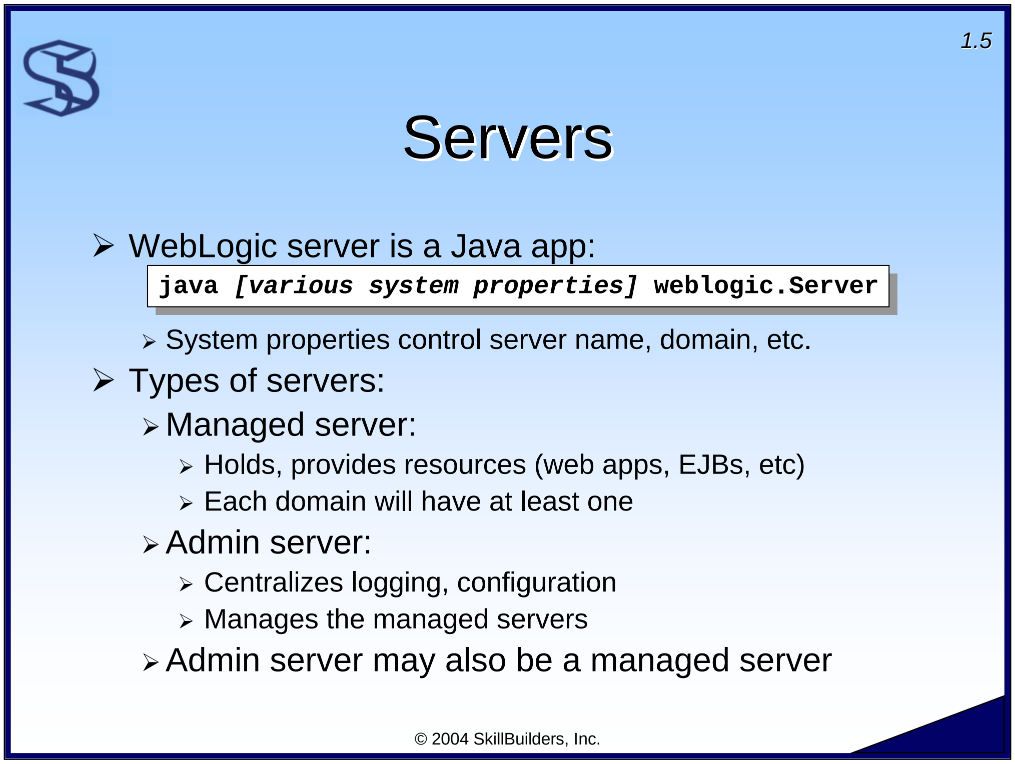
WLS Architecture

- WLS is designed for large-scale systems:
  - Redundant servers
  - Fail-over capabilities
- Core concepts:
  - Server
  - Domain
  - Cluster
- Diagram shows one domain
  - Details follow...

© 2004 SkillBuilders, Inc.

This diagram, from the documentation accompanying WebLogic server, illustrates generally how servers are organized.

For full documentation, browse to `<`*`bea_home`*`>/docs/index.html.`



1.5

Servers

- WebLogic server is a Java app:

```
java [various system properties] weblogic.Server
```

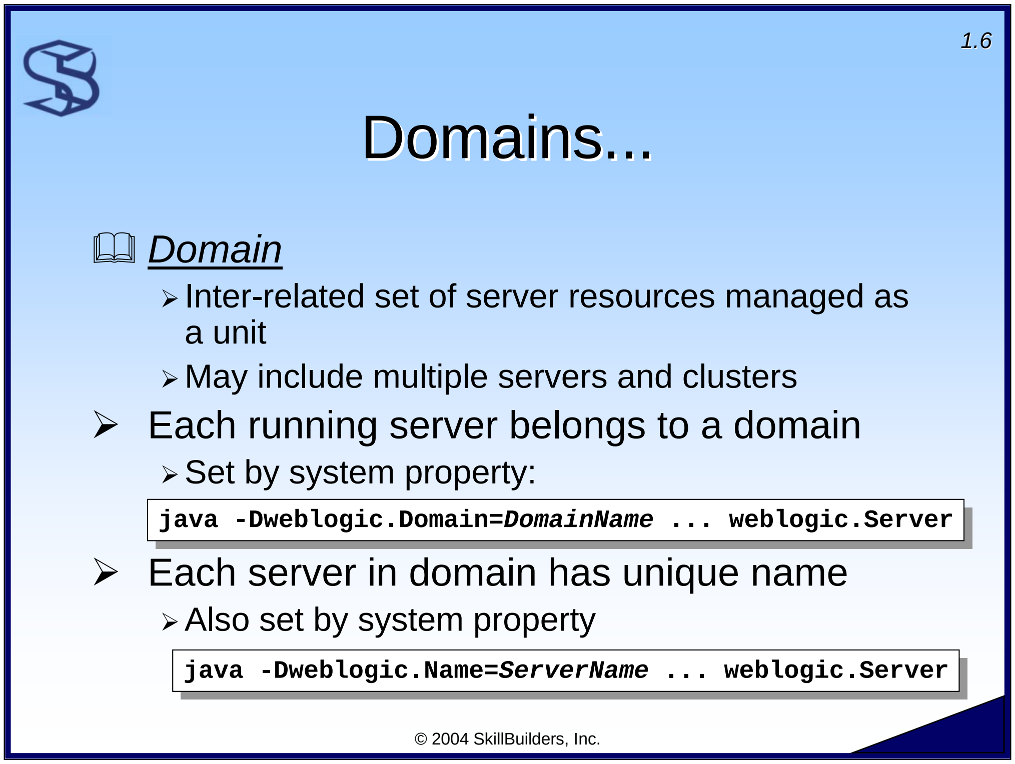
  - System properties control server name, domain, etc.
- Types of servers:
  - Managed server:
    - Holds, provides resources (web apps, EJBs, etc)
    - Each domain will have at least one
  - Admin server:
    - Centralizes logging, configuration
    - Manages the managed servers
  - Admin server may also be a managed server

© 2004 SkillBuilders, Inc.

A given WebLogic server instance will be a managed server or an admin server. Managed servers do all the work; admin servers centralize configuration information and management of the managed servers.

The same WebLogic server application is used in either case. To run an instance as a managed server, specify the URL of the admin server that will manage it:

```
java -Dweblogic.management.server=AdminHost:AdminPort ...
           weblogic.Server
```



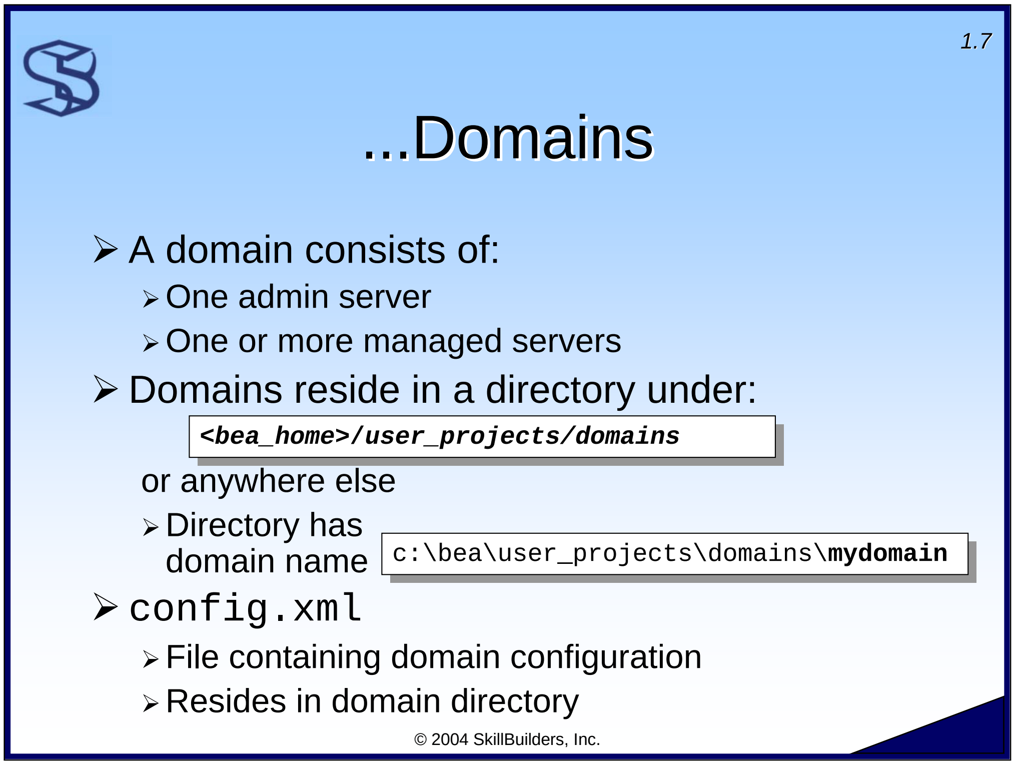
Domains...

- 📖 *Domain*
  - Inter-related set of server resources managed as a unit
  - May include multiple servers and clusters
- Each running server belongs to a domain
  - Set by system property:

```
java -Dweblogic.Domain=DomainName ... weblogic.Server
```

- Each server in domain has unique name
  - Also set by system property

```
java -Dweblogic.Name=ServerName ... weblogic.Server
```



1.7

...Domains

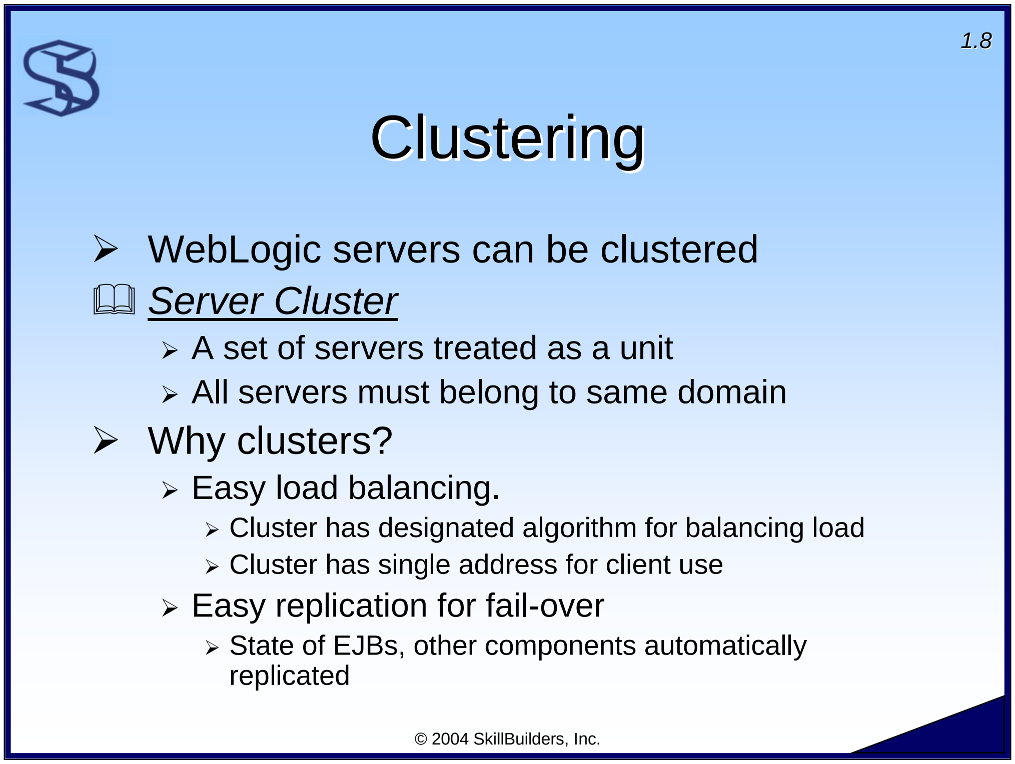
- A domain consists of:
  - One admin server
  - One or more managed servers
- Domains reside in a directory under:

  ```
  <bea_home>/user_projects/domains
  ```

  or anywhere else
  - Directory has domain name

    ```
    c:\bea\user_projects\domains\mydomain
    ```
- `config.xml`
  - File containing domain configuration
  - Resides in domain directory

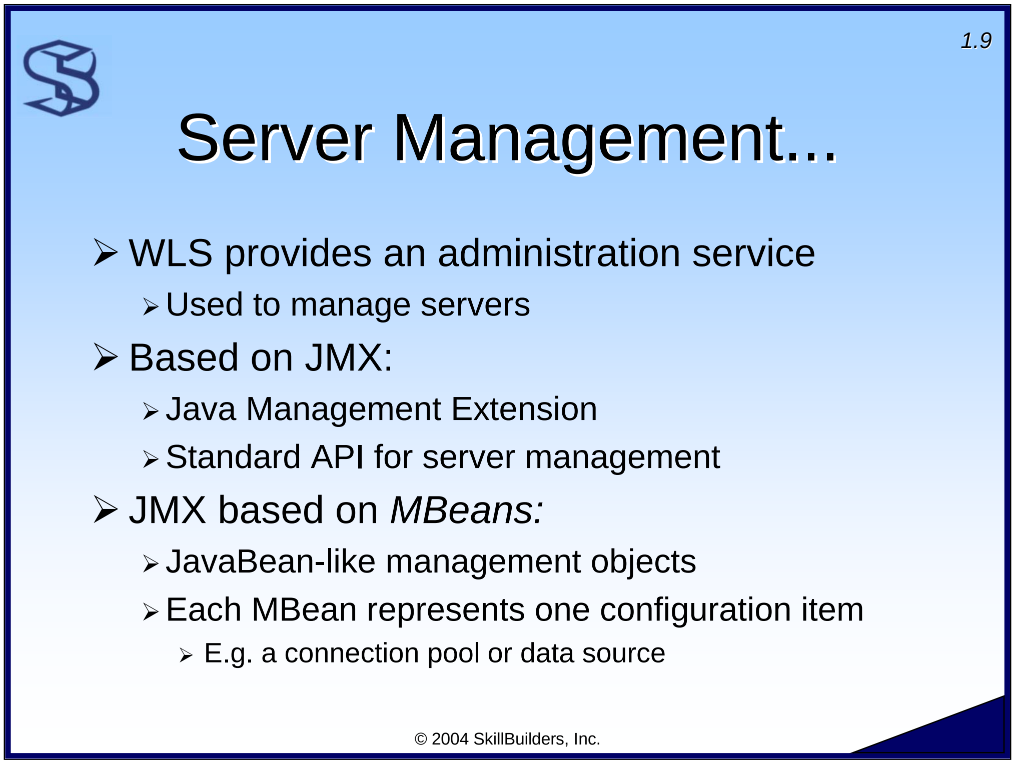
© 2004 SkillBuilders, Inc.



1.8

Clustering

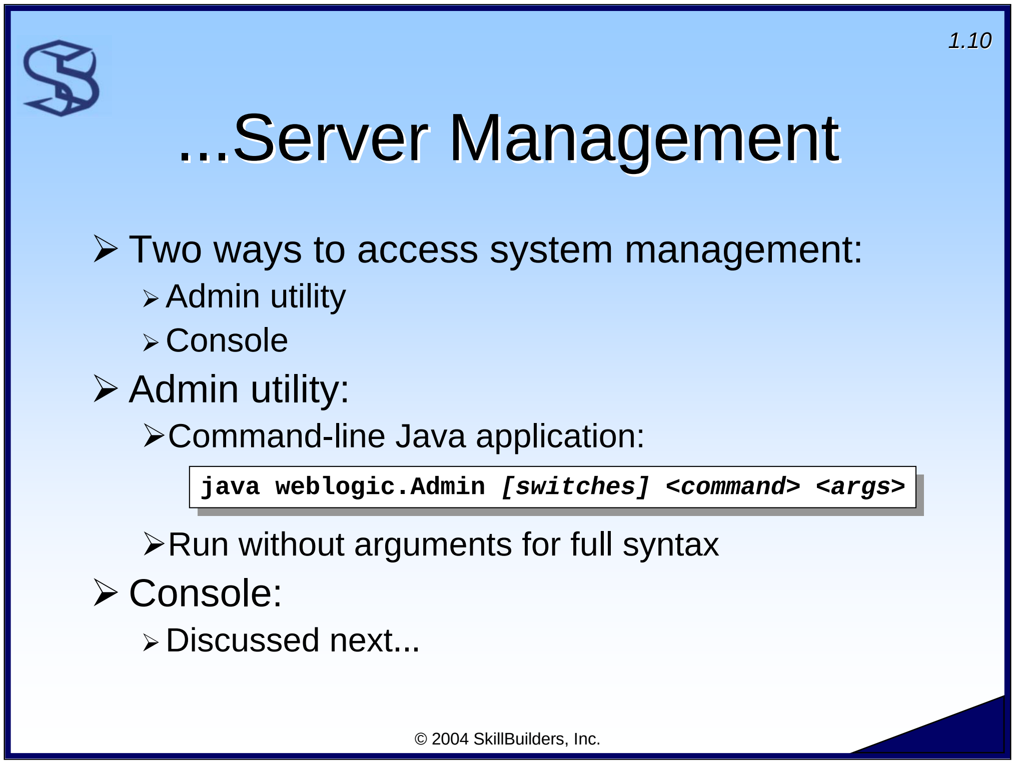
- WebLogic servers can be clustered
- 📖 *Server Cluster*
  - A set of servers treated as a unit
  - All servers must belong to same domain
- Why clusters?
  - Easy load balancing.
    - Cluster has designated algorithm for balancing load
    - Cluster has single address for client use
  - Easy replication for fail-over
    - State of EJBs, other components automatically replicated



1.9

Server Management...

- WLS provides an administration service
  - Used to manage servers
- Based on JMX:
  - Java Management Extension
  - Standard API for server management
- JMX based on *MBeans:*
  - JavaBean-like management objects
  - Each MBean represents one configuration item
    - E.g. a connection pool or data source



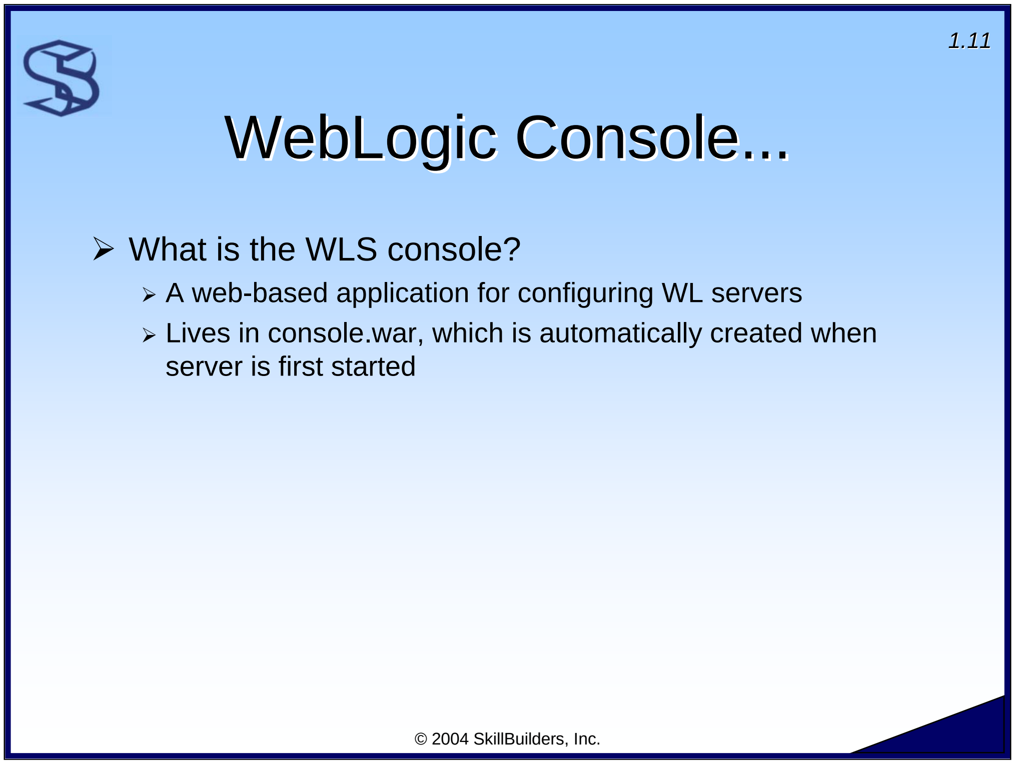
1.10

...Server Management

- Two ways to access system management:
  - Admin utility
  - Console
- Admin utility:
  - Command-line Java application:

```
java weblogic.Admin [switches] <command> <args>
```

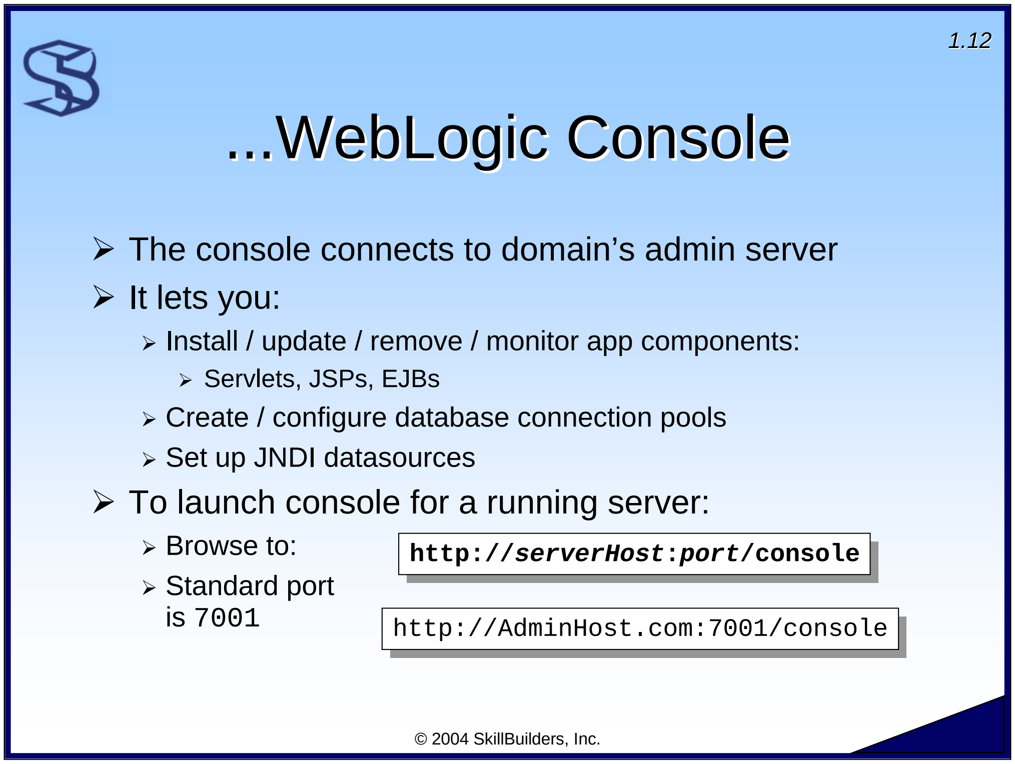
  - Run without arguments for full syntax
- Console:
  - Discussed next...



1.11

WebLogic Console...

- What is the WLS console?
  - A web-based application for configuring WL servers
  - Lives in console.war, which is automatically created when server is first started



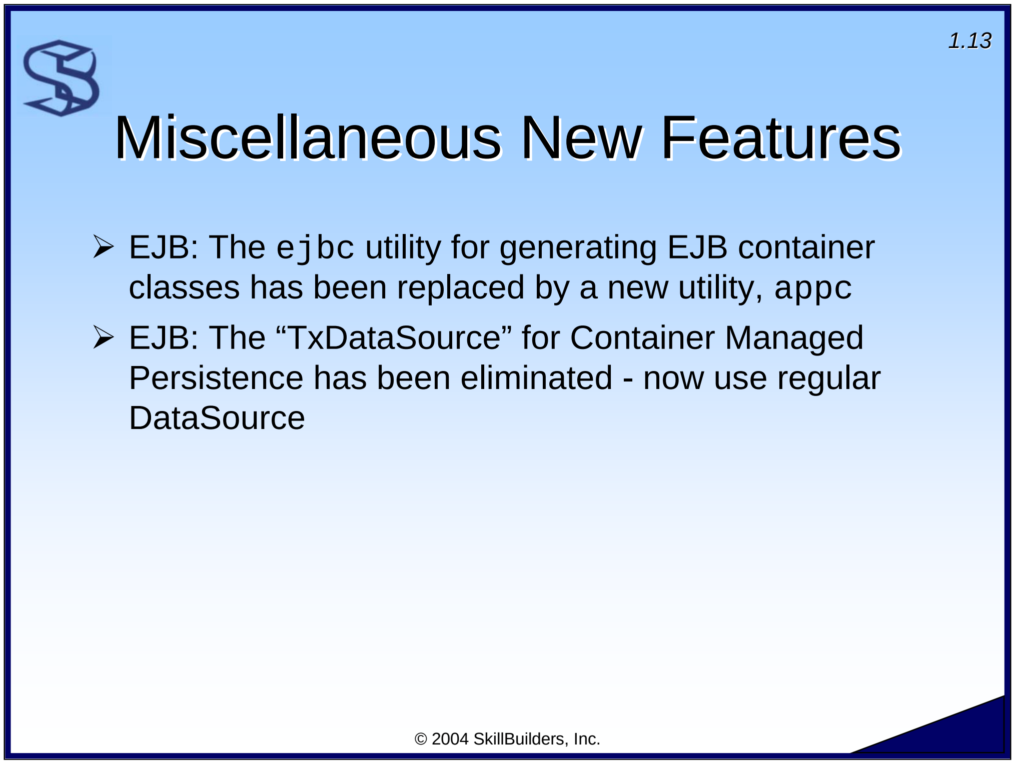
1.12

...WebLogic Console

- The console connects to domain's admin server
- It lets you:
  - Install / update / remove / monitor app components:
    - Servlets, JSPs, EJBs
  - Create / configure database connection pools
  - Set up JNDI datasources
- To launch console for a running server:
  - Browse to:
  - Standard port is `7001`

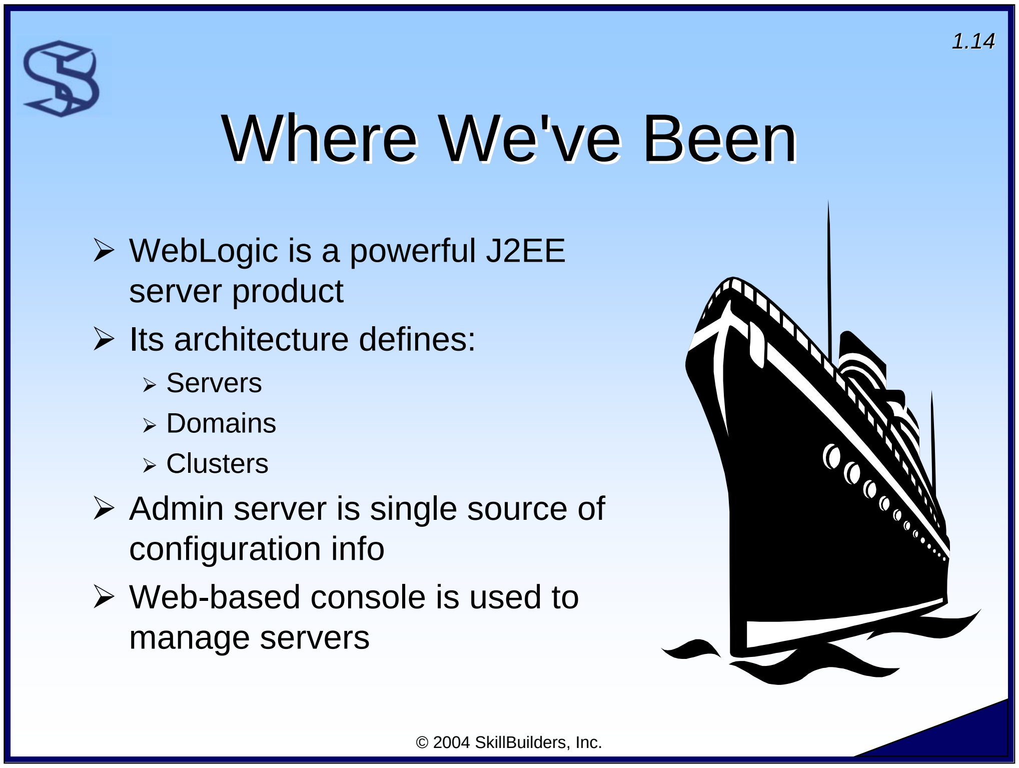
```
http://serverHost:port/console
```

```
http://AdminHost.com:7001/console
```



Miscellaneous New Features

- EJB: The `ejbc` utility for generating EJB container classes has been replaced by a new utility, `appc`
- EJB: The “TxDataSource” for Container Managed Persistence has been eliminated - now use regular DataSource



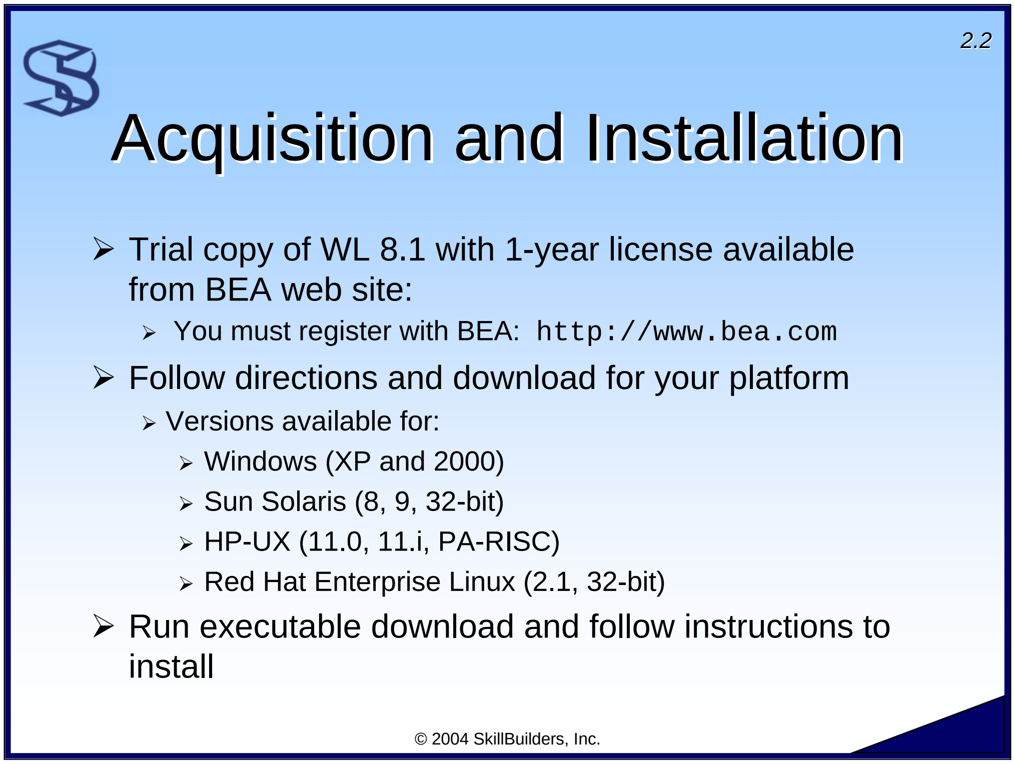
1.14

Where We've Been

- WebLogic is a powerful J2EE server product
- Its architecture defines:
  - Servers
  - Domains
  - Clusters
- Admin server is single source of configuration info
- Web-based console is used to manage servers

2. Installation & Configuration of WebLogic 8.1

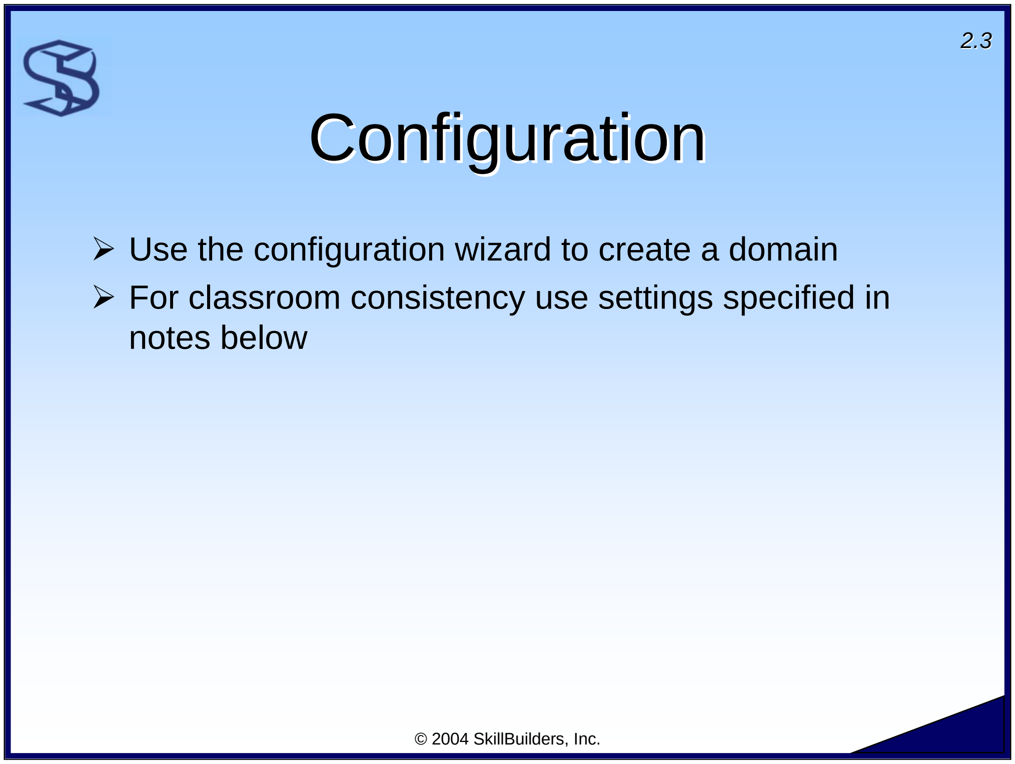
SKILLBUILDERS



2.2

Acquisition and Installation

- Trial copy of WL 8.1 with 1-year license available from BEA web site:
  - You must register with BEA: `http://www.bea.com`
- Follow directions and download for your platform
  - Versions available for:
    - Windows (XP and 2000)
    - Sun Solaris (8, 9, 32-bit)
    - HP-UX (11.0, 11.i, PA-RISC)
    - Red Hat Enterprise Linux (2.1, 32-bit)
- Run executable download and follow instructions to install



2.3

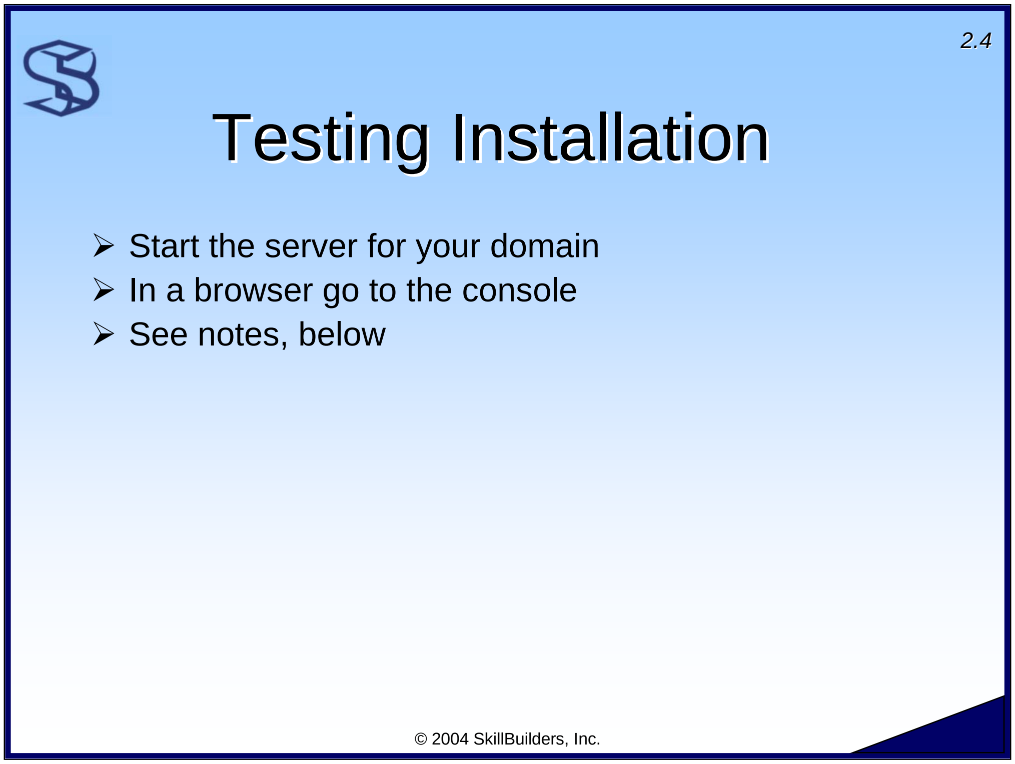
Configuration

- ➢ Use the configuration wizard to create a domain
- ➢ For classroom consistency use settings specified in notes below

Configuring a Domain

Before starting WebLogic 8.1 you should create a domain. The easiest way is to run the *WebLogic Configuration Wizard*. Click on: `start`→`All Programs`→`BEA WebLogic Platform 8.1`→`Configuration Wizard`. Follow the screen prompts, responding as noted below. Accept the suggested default for anything not mentioned.

1. Select `Create a new WebLogic configuration`
2. template: `Basic WebLogic Server Domain`
3. Choose `Express`
4. Username: `admin`
5. Password: `adminadmin`
6. Choose `Development Mode`
7. Choose `BEA Supplied SDKs`
8. Choose `Sun SDK`
9. Configuration Name: `mydomain`



2.4

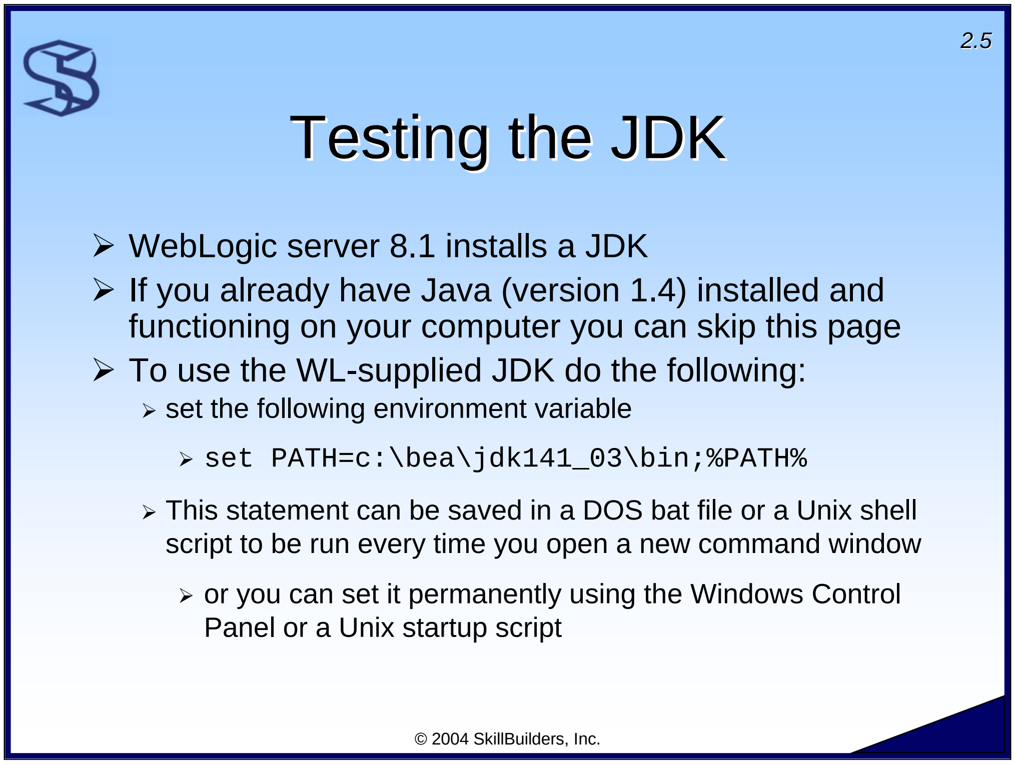
Testing Installation

- Start the server for your domain
- In a browser go to the console
- See notes, below

Testing the WebLogic Server Installation

To test that your WebLogic installation is correct, do the following:

- Open a DOS window and cd to the following directory: `c:\bea\user_projects\domains\mydomain`
- Type `startWebLogic` and hit the Enter key.
- Open the default console. Open a browser window and enter the URL `http://localhost:7001/console`. A screen should appear prompting for user and password.
- Enter the username and password you specified when you ran the domain configuration wizard (`admin` and `adminadmin`). Then click the Sign In button.
- If the live console screen appears, your installation and configuration were successful.



2.5

Testing the JDK

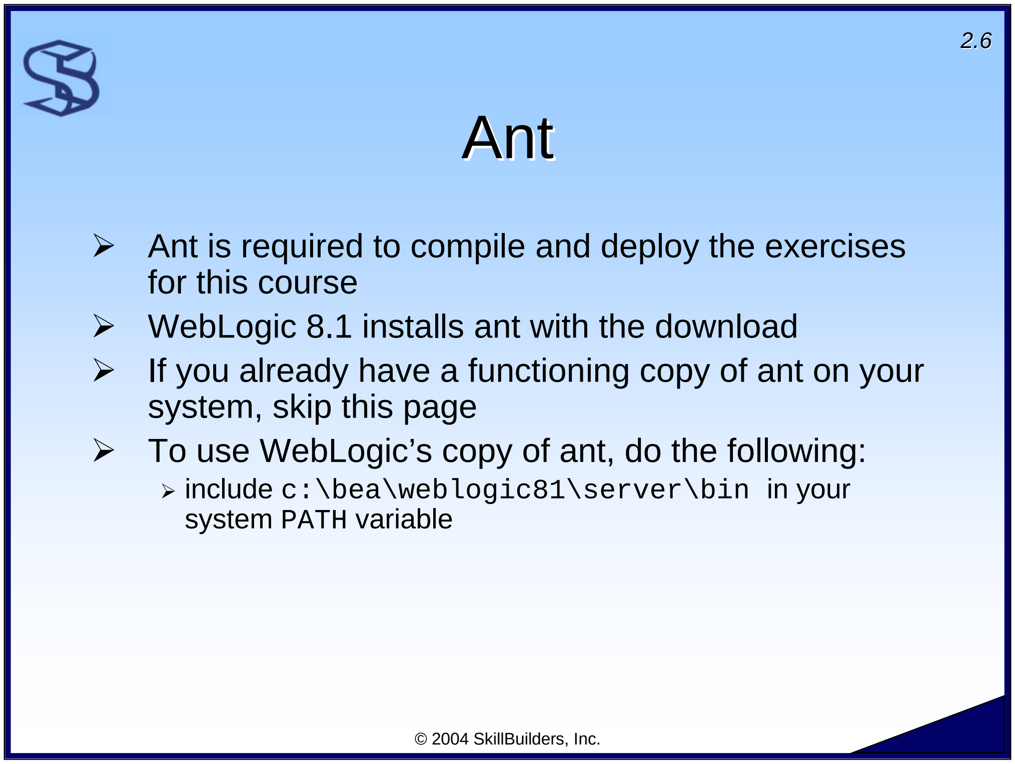
- WebLogic server 8.1 installs a JDK
- If you already have Java (version 1.4) installed and functioning on your computer you can skip this page
- To use the WL-supplied JDK do the following:
  - set the following environment variable
    - `set PATH=c:\bea\jdk141_03\bin;%PATH%`
  - This statement can be saved in a DOS bat file or a Unix shell script to be run every time you open a new command window
    - or you can set it permanently using the Windows Control Panel or a Unix startup script

To test your JDK configuration enter the following on the DOS command line:

```
java -version
```

The response should look something like the following:

```
java version "1.4.2"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2-b28)
Java HotSpot(TM) Client VM (build 1.4.2-b28, mixed mode)
```



2.6

Ant

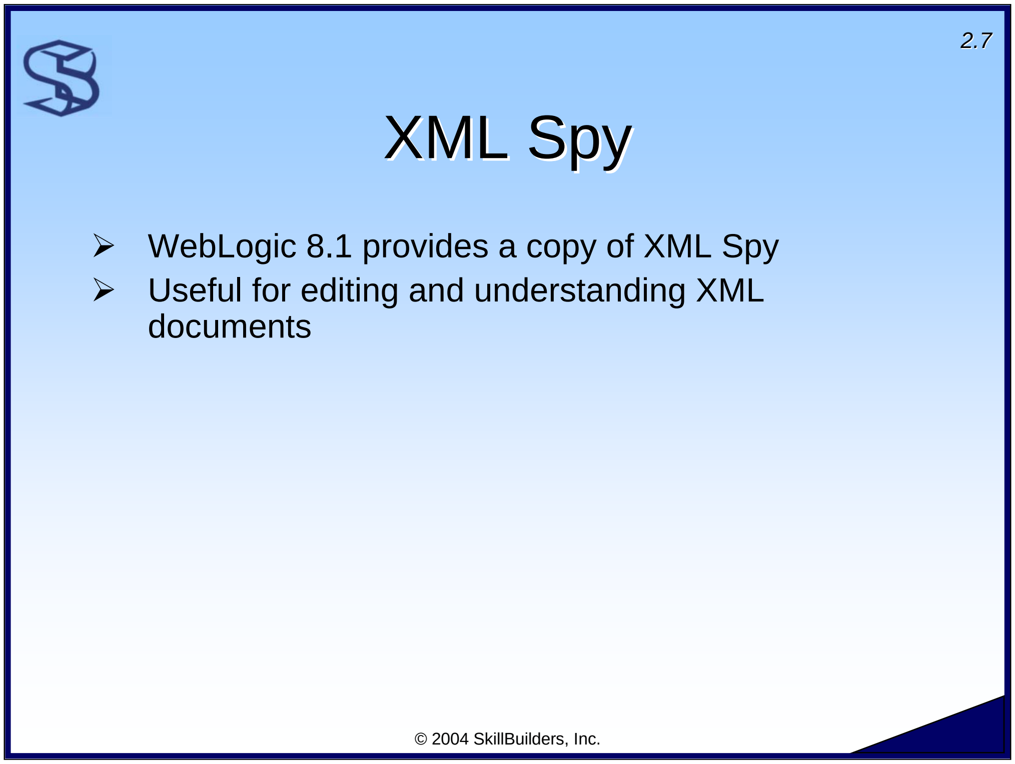
- Ant is required to compile and deploy the exercises for this course
- WebLogic 8.1 installs ant with the download
- If you already have a functioning copy of ant on your system, skip this page
- To use WebLogic’s copy of ant, do the following:
  - include `c:\bea\weblogic81\server\bin` in your system PATH variable

You can add the directory to your path by entering the following statement on the command line:

```
set PATH=c:\bea\weblogic81\server\bin;%PATH%
```

You can add this statement to a DOS bat file or Unix shell script along with the statements on the previous pages.

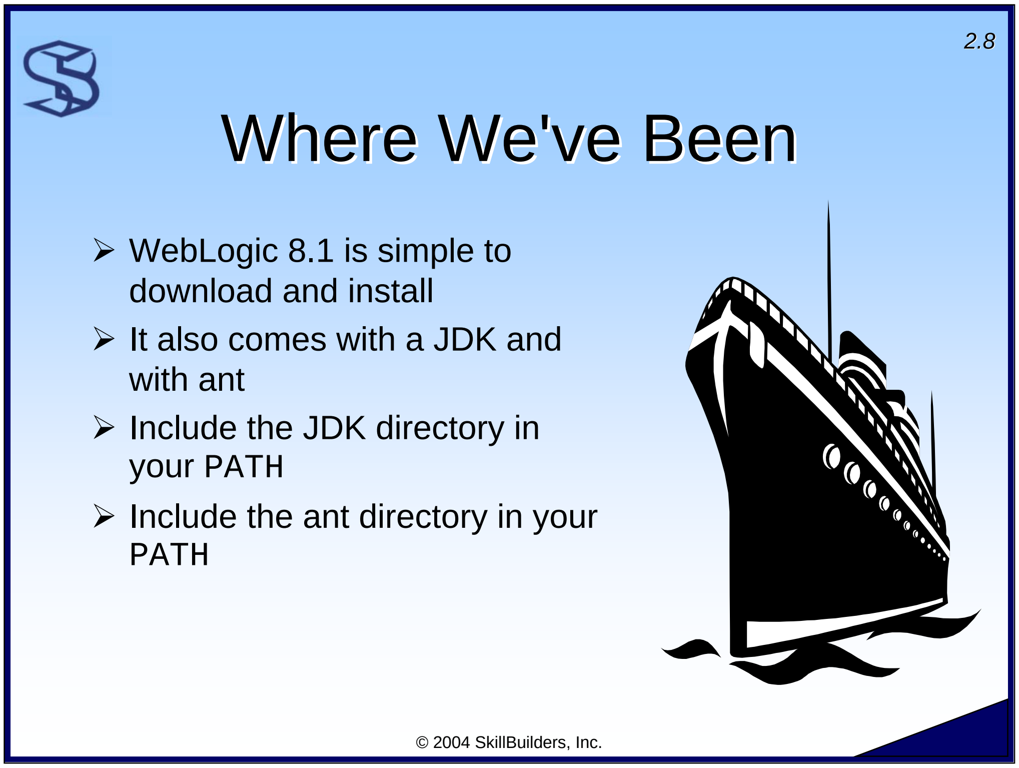
You can also download ant from `www.apache.org`



2.7

XML Spy

- WebLogic 8.1 provides a copy of XML Spy
- Useful for editing and understanding XML documents



2.8

Where We've Been

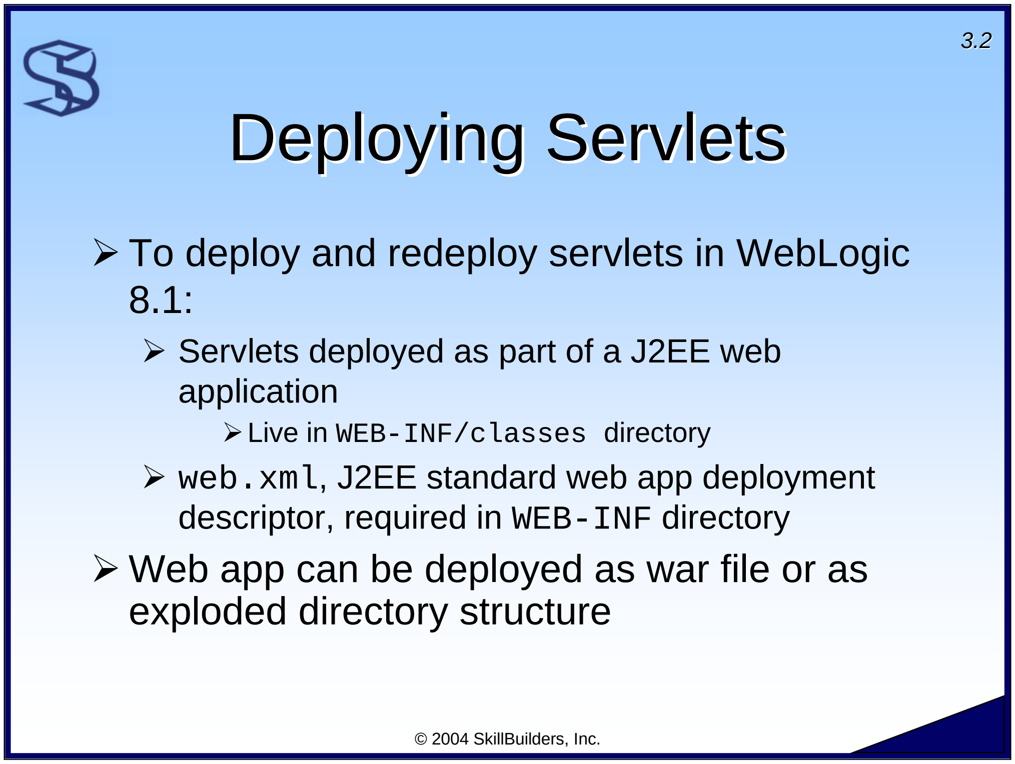
- WebLogic 8.1 is simple to download and install
- It also comes with a JDK and with ant
- Include the JDK directory in your PATH
- Include the ant directory in your PATH

© 2004 SkillBuilders, Inc.

3. Deploying Web Apps in WebLogic 8.1

WebLogic Overview
Deploying Servlets
Installing Web Applications
Managing Deployments

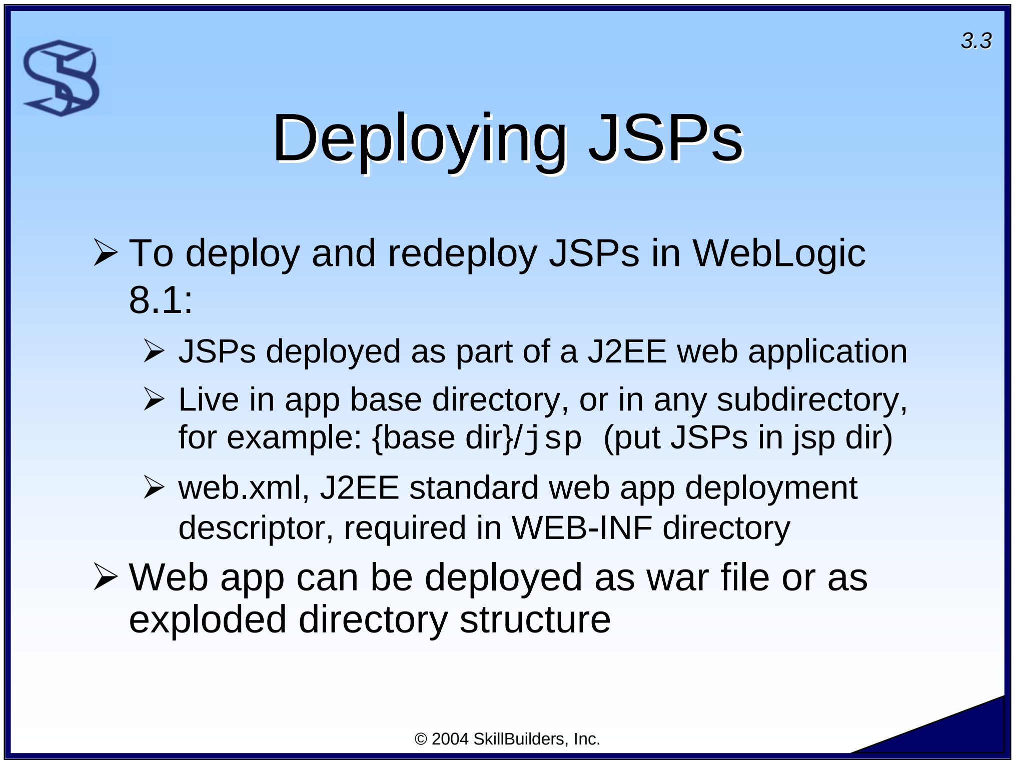
SKILLBUILDERS



3.2

Deploying Servlets

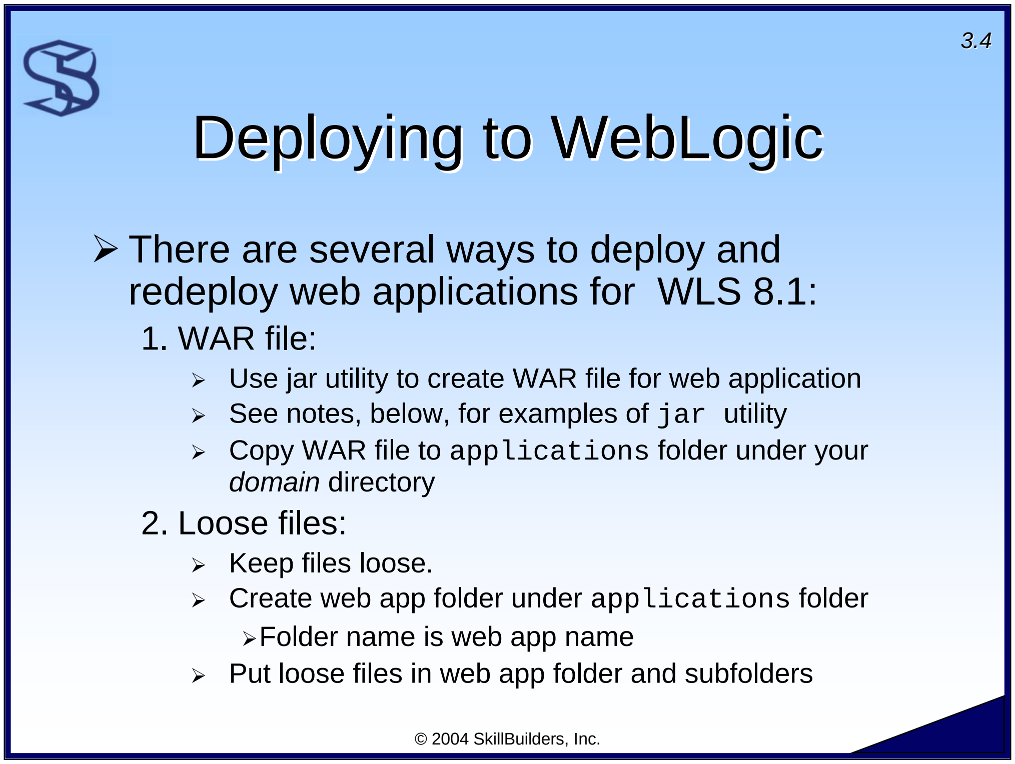
- To deploy and redeploy servlets in WebLogic 8.1:
  - Servlets deployed as part of a J2EE web application
    - Live in `WEB-INF/classes` directory
  - `web.xml`, J2EE standard web app deployment descriptor, required in `WEB-INF` directory
- Web app can be deployed as war file or as exploded directory structure



3.3

Deploying JSPs

- To deploy and redeploy JSPs in WebLogic 8.1:
  - JSPs deployed as part of a J2EE web application
  - Live in app base directory, or in any subdirectory, for example: {base dir}/`jsp` (put JSPs in jsp dir)
  - web.xml, J2EE standard web app deployment descriptor, required in WEB-INF directory
- Web app can be deployed as war file or as exploded directory structure



3.4

Deploying to WebLogic

- There are several ways to deploy and redeploy web applications for WLS 8.1:
  1. WAR file:
     - Use jar utility to create WAR file for web application
     - See notes, below, for examples of `jar` utility
     - Copy WAR file to `applications` folder under your *domain* directory
  2. Loose files:
     - Keep files loose.
     - Create web app folder under `applications` folder
       - Folder name is web app name
     - Put loose files in web app folder and subfolders

© 2004 SkillBuilders, Inc.

In development mode, every few seconds WebLogic checks the applications directory under the *domain* folder for a new or changed war file or a new or changed file in an exploded application. When it detects that something has changed, it will attempt to dynamically deploy or redeploy the application.

The environment variable `STARTMODE` in the WebLogic startup script (`startWebLogic.cmd`) determines whether WebLogic will run in development mode or production mode. To use production mode set the value of `STARTMODE` to `TRUE`. To use development mode with autodeployment, set it to `FALSE`, leave it blank (or omit the statement).

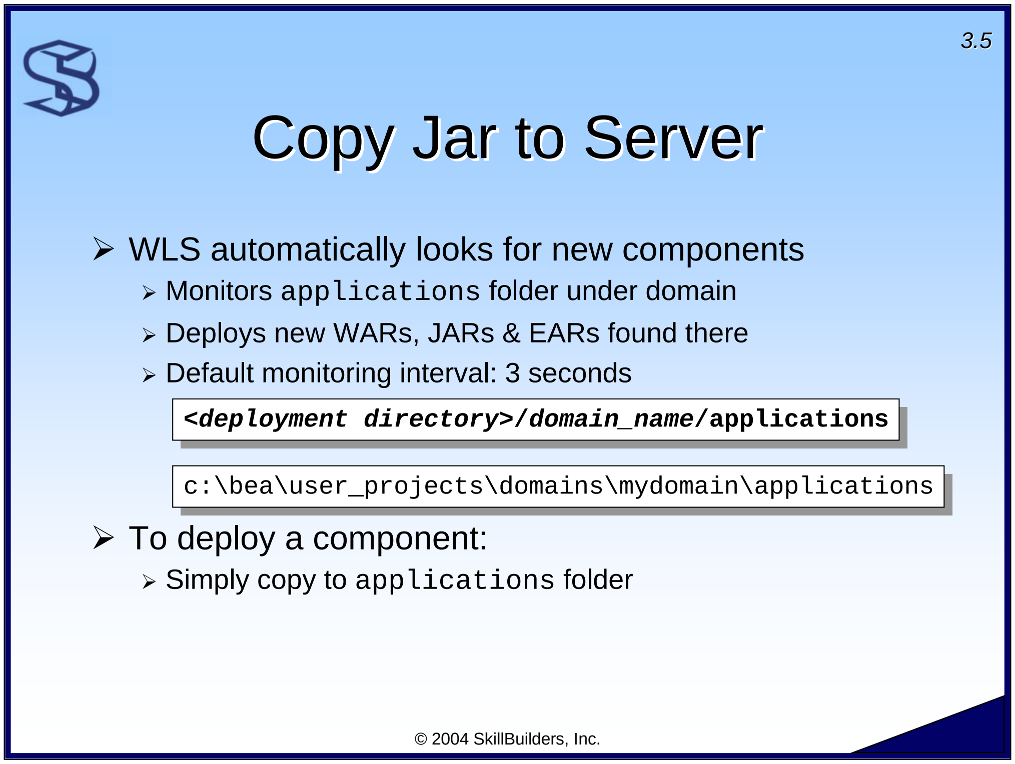
```
(nothing)             development/autodeploy
set STARTMODE=        development/autodeploy
set STARTMODE=FALSE   development/autodeploy
set STARTMODE=TRUE    production mode/NO autodeploy
```

You can use the jar utility to create a war file as follows:

```
     jar cvf {appname}.war {filespec1} {filespec2} . . .
```

You can use the jar utility to update a war file as follows:

```
     jar uvf {appname}.war {filespec1} {filespec2} . . .
```



3.5

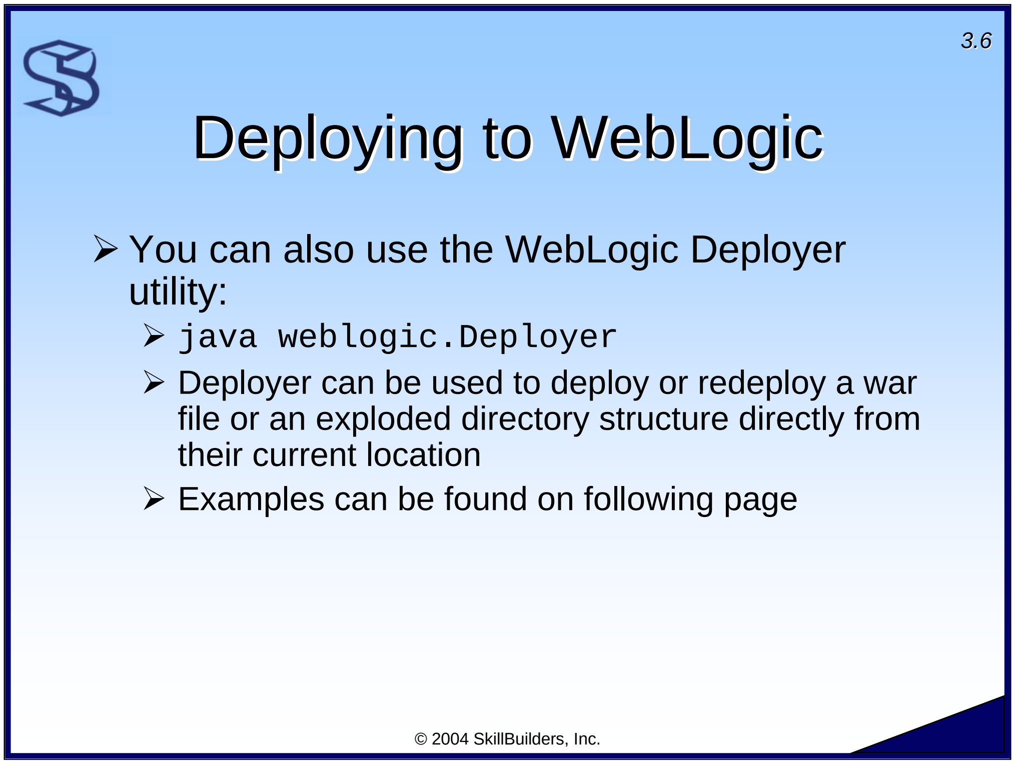
Copy Jar to Server

- WLS automatically looks for new components
  - Monitors `applications` folder under domain
  - Deploys new WARs, JARs & EARs found there
  - Default monitoring interval: 3 seconds

```
<deployment directory>/domain_name/applications
```

```
c:\bea\user_projects\domains\mydomain\applications
```

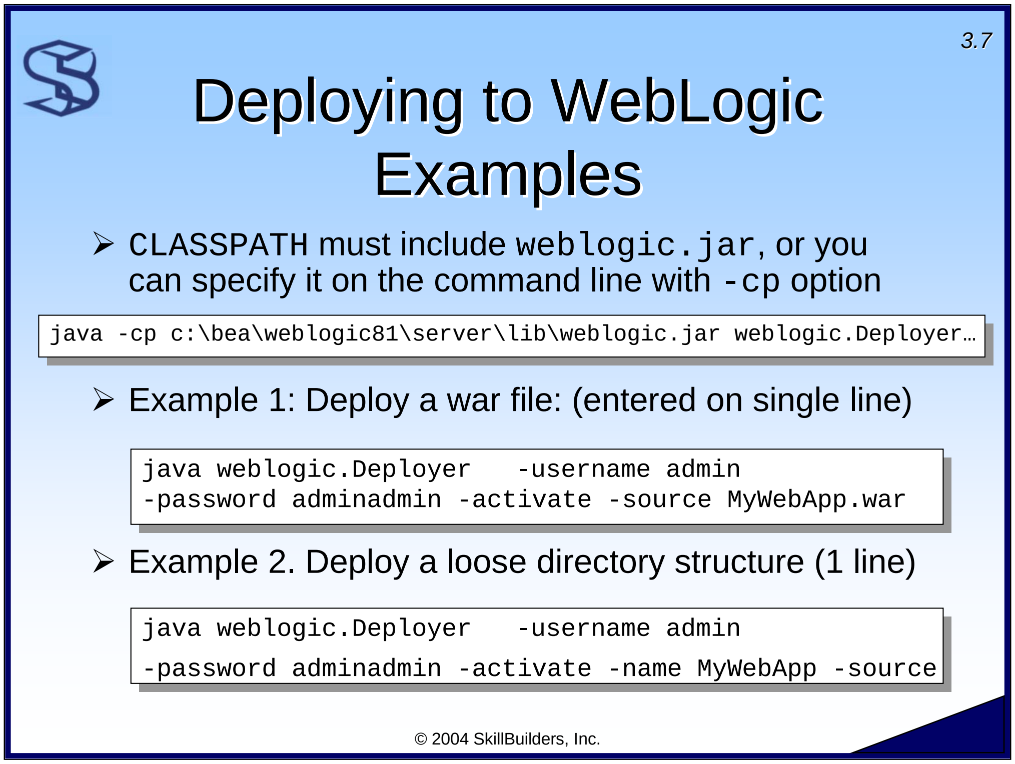
- To deploy a component:
  - Simply copy to `applications` folder



3.6

Deploying to WebLogic

- You can also use the WebLogic Deployer utility:
  - `java weblogic.Deployer`
  - Deployer can be used to deploy or redeploy a war file or an exploded directory structure directly from their current location
  - Examples can be found on following page



3.7

Deploying to WebLogic Examples

- CLASSPATH must include weblogic.jar, or you can specify it on the command line with -cp option

```
java -cp c:\bea\weblogic81\server\lib\weblogic.jar weblogic.Deployer…
```

- Example 1: Deploy a war file: (entered on single line)

```
java weblogic.Deployer   -username admin
-password adminadmin -activate -source MyWebApp.war
```

- Example 2. Deploy a loose directory structure (1 line)

```
java weblogic.Deployer   -username admin
-password adminadmin -activate -name MyWebApp -source
```

© 2004 SkillBuilders, Inc.

The following are some examples of using the WebLogic Deployer utility. Your CLASSPATH must include weblogic.jar, or you can specify it on the command line with the -cp option as shown above and again here:

```
java -cp c:\bea\weblogic81\server\lib\weblogic.jar weblogic.Deployer…
```

The examples above use these options shown on separate lines for clarity. They must be coded on a single line when run.

1. Deploy a war file:

```
     java weblogic.Deployer  -username {admin username}
                             -password {admin password}
                             -activate -source {warfile name}
```

E.g.:

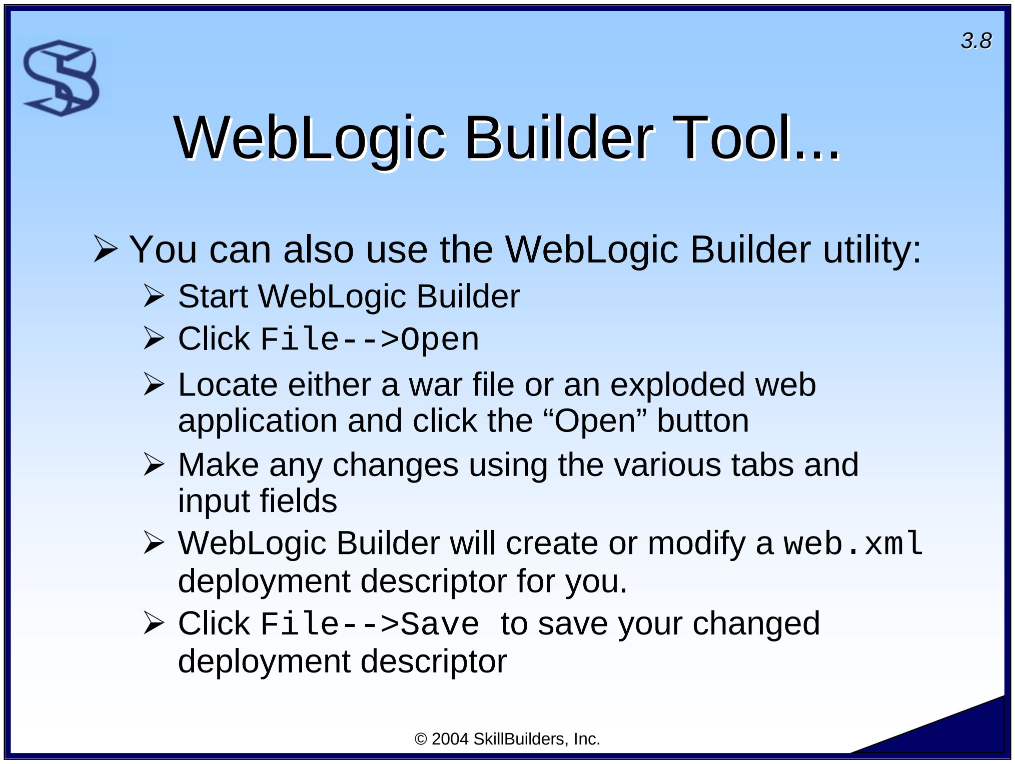
```
 -username admin -password adminadmin -activate -source MyWebApp.war
```

2. Deploy a loose directory structure

```
     java weblogic.Deployer  -username {admin username}
                             -password {admin password}
                             -activate
                             -name {app name} -source {base dir}
```

E.g:

```
 -activate -name MyWebApp -source
```

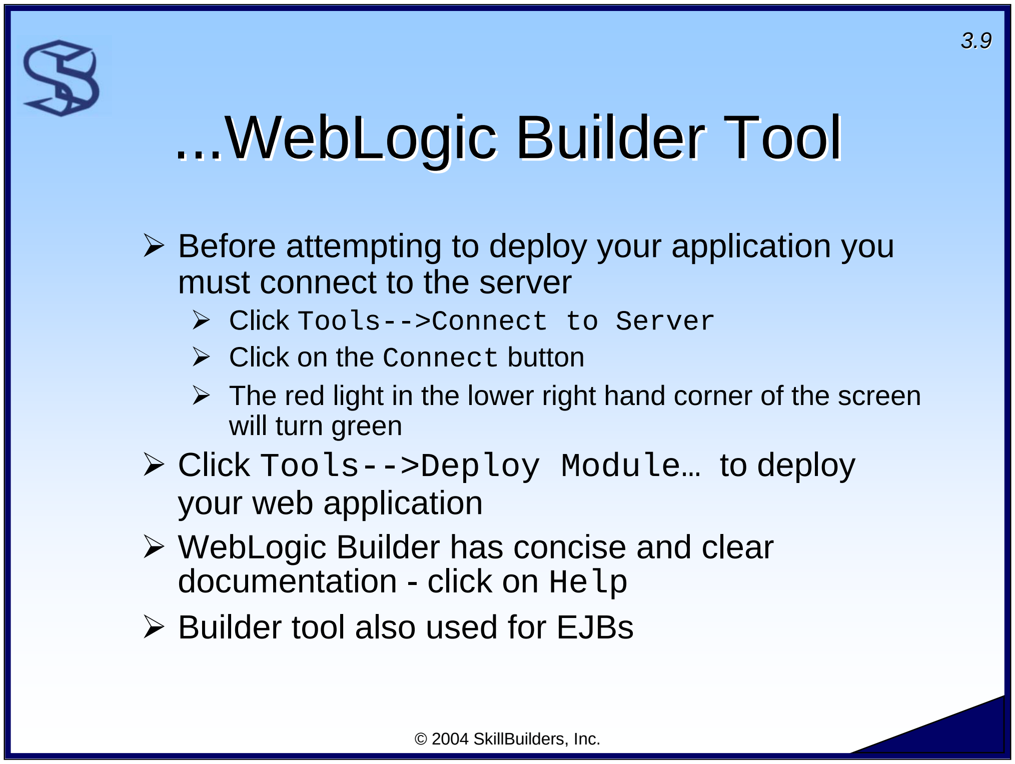


3.8

WebLogic Builder Tool...

- You can also use the WebLogic Builder utility:
  - Start WebLogic Builder
  - Click `File-->Open`
  - Locate either a war file or an exploded web application and click the “Open” button
  - Make any changes using the various tabs and input fields
  - WebLogic Builder will create or modify a `web.xml` deployment descriptor for you.
  - Click `File-->Save` to save your changed deployment descriptor

You can view the `web.xml` and `weblogic.xml` files created or modified by the WebLogic Builder by clicking on `View-->XML Source` and using the tabs to view each file.

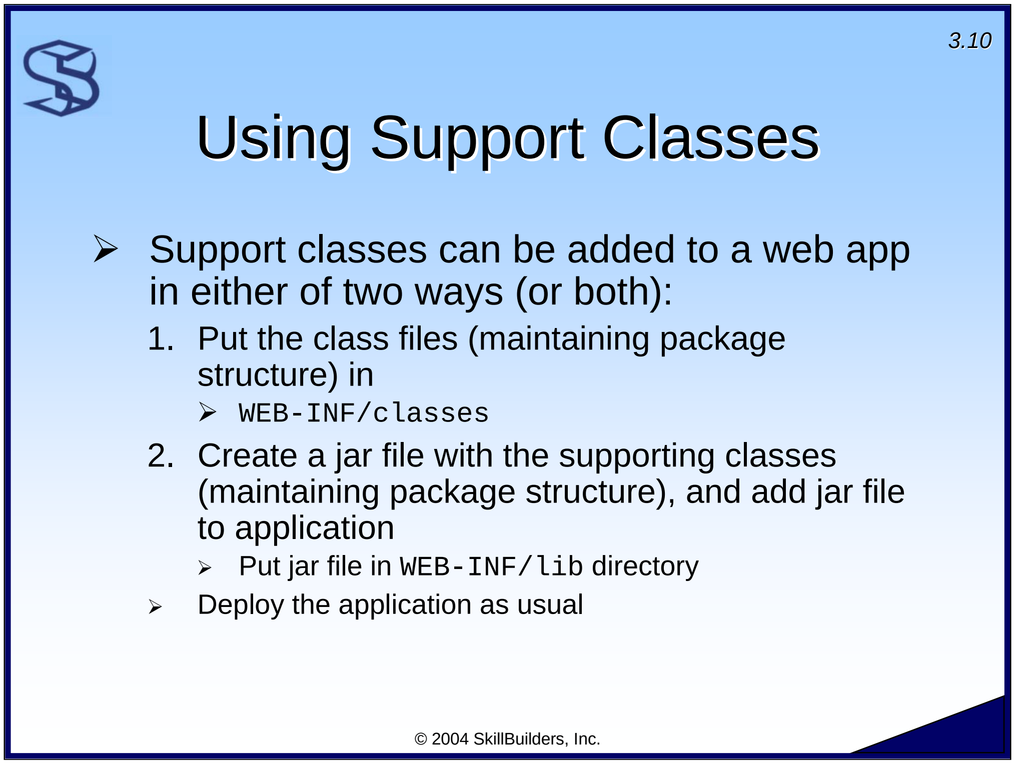


3.9

...WebLogic Builder Tool

- Before attempting to deploy your application you must connect to the server
  - Click `Tools-->Connect to Server`
  - Click on the `Connect` button
  - The red light in the lower right hand corner of the screen will turn green
- Click `Tools-->Deploy Module…` to deploy your web application
- WebLogic Builder has concise and clear documentation - click on `Help`
- Builder tool also used for EJBs

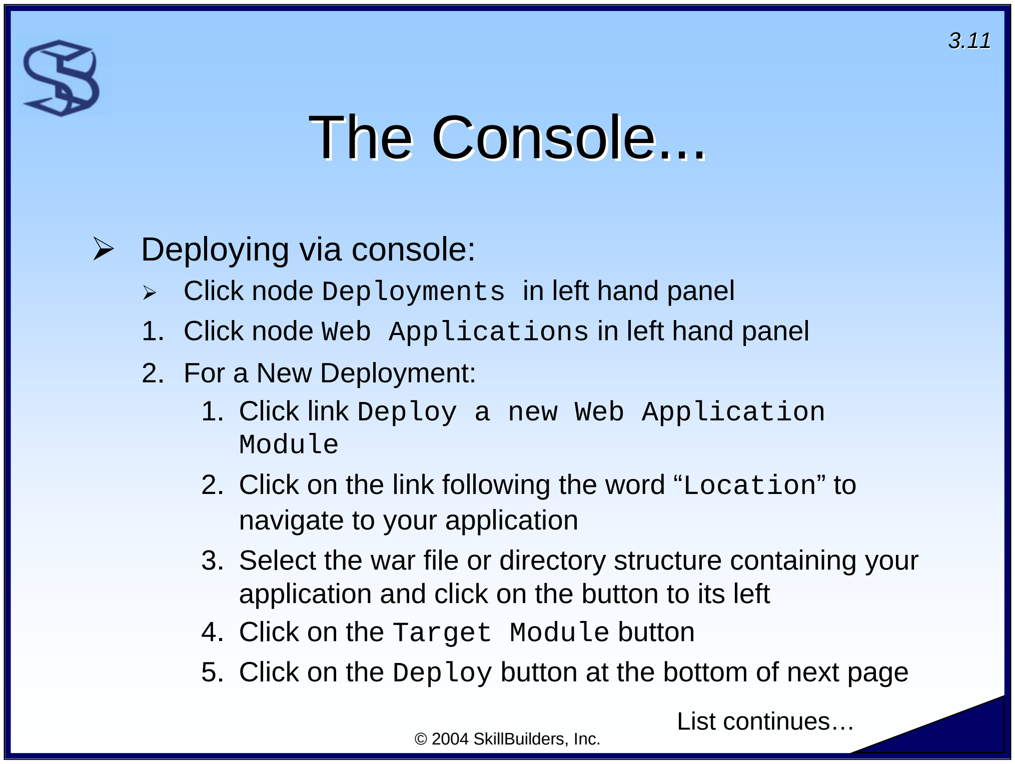
You can view the `web.xml` and `weblogic.xml` files created or modified by the WebLogic Builder by clicking on `View-->XML Source` and using the tabs to view each file.



3.10

Using Support Classes

- Support classes can be added to a web app in either of two ways (or both):
  1. Put the class files (maintaining package structure) in
     - `WEB-INF/classes`
  2. Create a jar file with the supporting classes (maintaining package structure), and add jar file to application
     - Put jar file in `WEB-INF/lib` directory
  - Deploy the application as usual



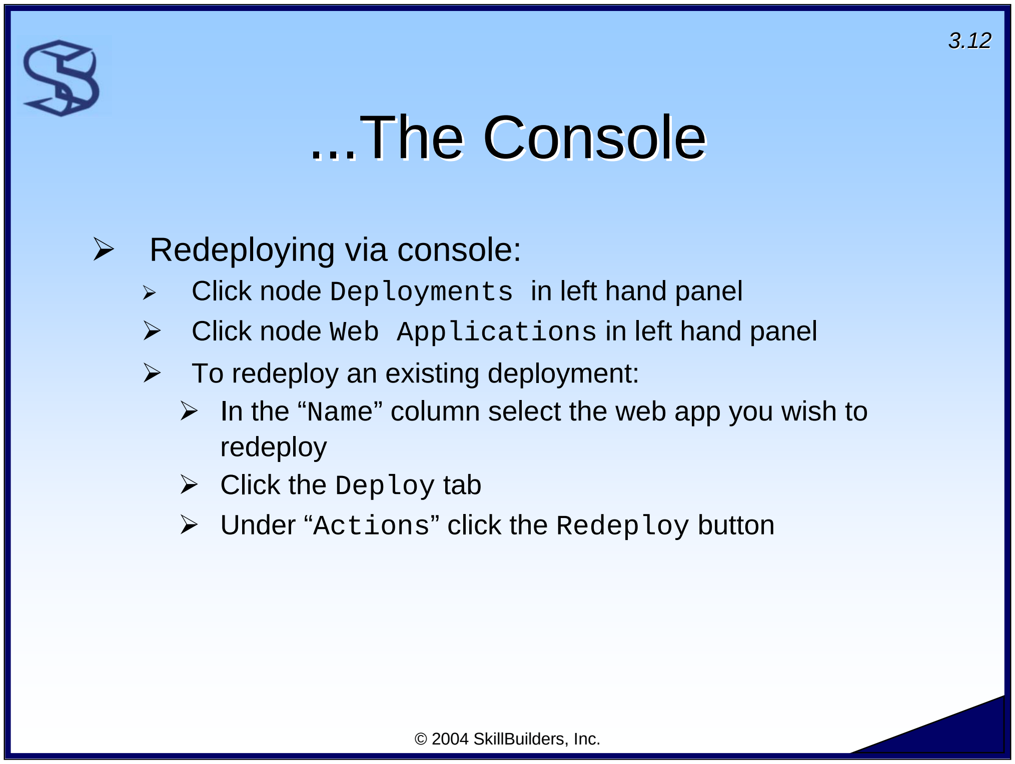
3.11

The Console...

- Deploying via console:
  - Click node `Deployments` in left hand panel
  1. Click node `Web Applications` in left hand panel
  2. For a New Deployment:
     1. Click link `Deploy a new Web Application Module`
     2. Click on the link following the word “`Location`” to navigate to your application
     3. Select the war file or directory structure containing your application and click on the button to its left
     4. Click on the `Target Module` button
     5. Click on the `Deploy` button at the bottom of next page

List continues…

© 2004 SkillBuilders, Inc.

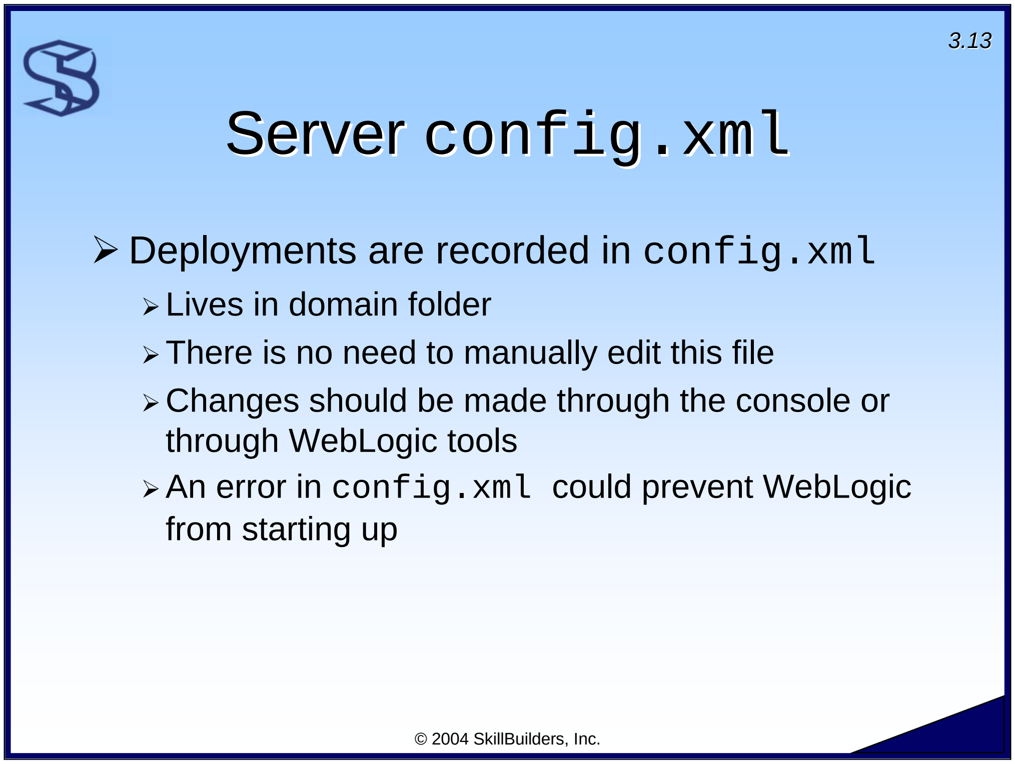


3.12

...The Console

- Redeploying via console:
  - Click node `Deployments` in left hand panel
  - Click node `Web Applications` in left hand panel
  - To redeploy an existing deployment:
    - In the “`Name`” column select the web app you wish to redeploy
    - Click the `Deploy` tab
    - Under “`Actions`” click the `Redeploy` button

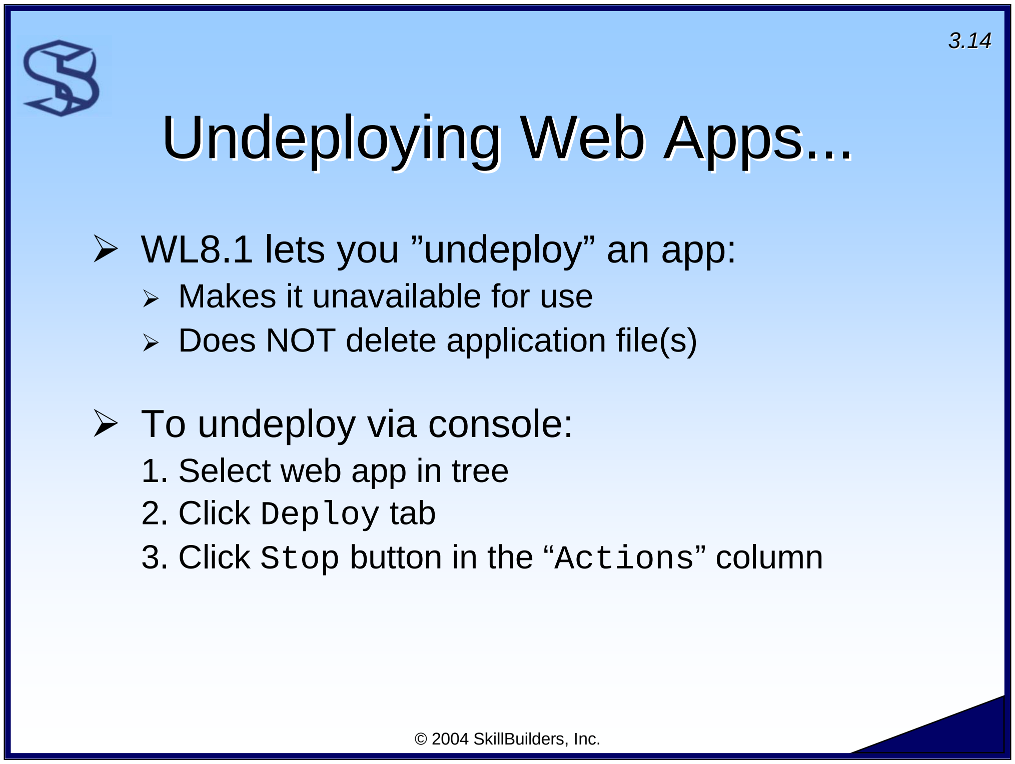
© 2004 SkillBuilders, Inc.



3.13

Server `config.xml`

- Deployments are recorded in `config.xml`
  - Lives in domain folder
  - There is no need to manually edit this file
  - Changes should be made through the console or through WebLogic tools
  - An error in `config.xml` could prevent WebLogic from starting up

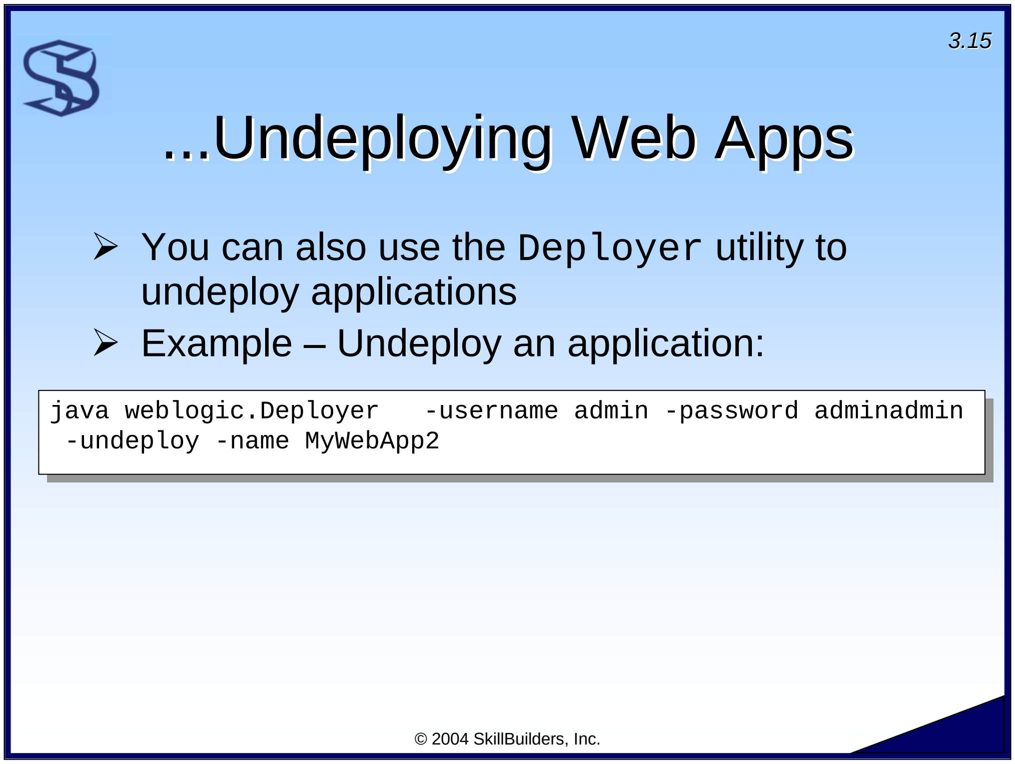


3.14

Undeploying Web Apps...

- WL8.1 lets you ”undeploy” an app:
  - Makes it unavailable for use
  - Does NOT delete application file(s)

- To undeploy via console:
  1. Select web app in tree
  2. Click `Deploy` tab
  3. Click `Stop` button in the “`Actions`” column



3.15

...Undeploying Web Apps

- You can also use the `Deployer` utility to undeploy applications
- Example – Undeploy an application:

```
java weblogic.Deployer   -username admin -password adminadmin
 -undeploy -name MyWebApp2
```

© 2004 SkillBuilders, Inc.

The following is an example of using the WebLogic Deployer utility. As noted earlier, your `CLASSPATH` must include `weblogic.jar`, or you can specify it on the command line with the `-cp` option. E.g.:

```
java -cp c:\bea\weblogic81\server\lib\weblogic.jar weblogic.Deployer…
```

The following example shows options on separate lines for clarity. They must be coded on a single line when run.

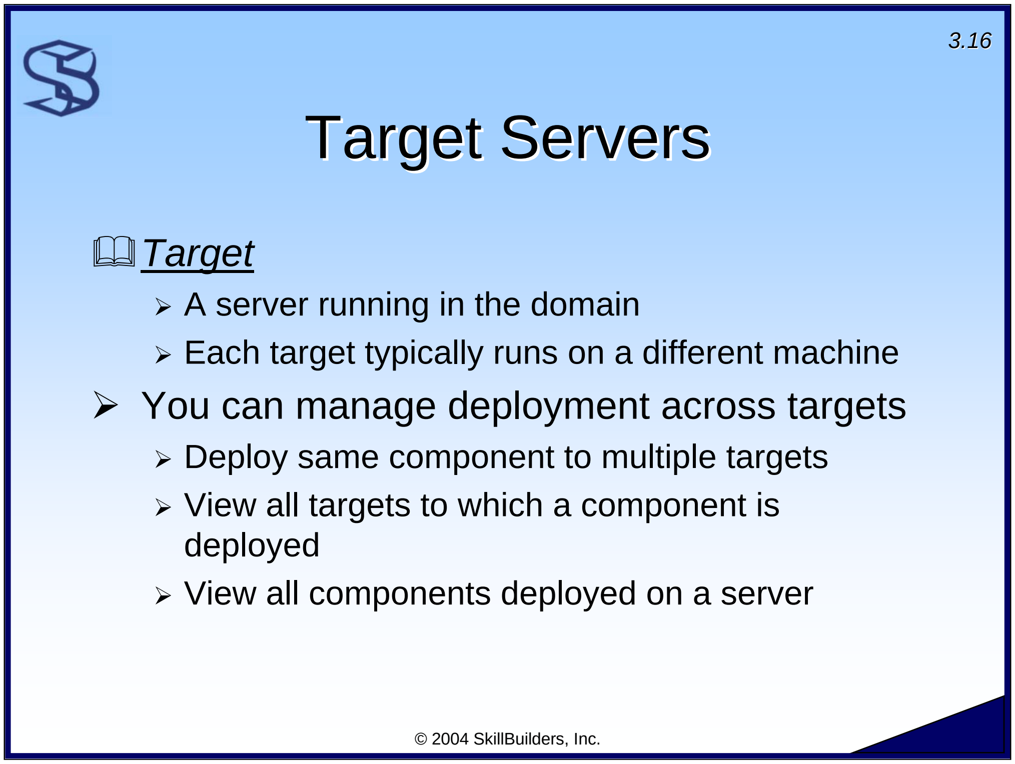
1. Undeploy an application options:

```
java weblogic.Deployer   -username {admin username}
                         -password {admin password}
                         -undeploy -name {appname}
```

E.g.:

```
-username admin -password adminadmin -undeploy -name MyWebApp2
```

.



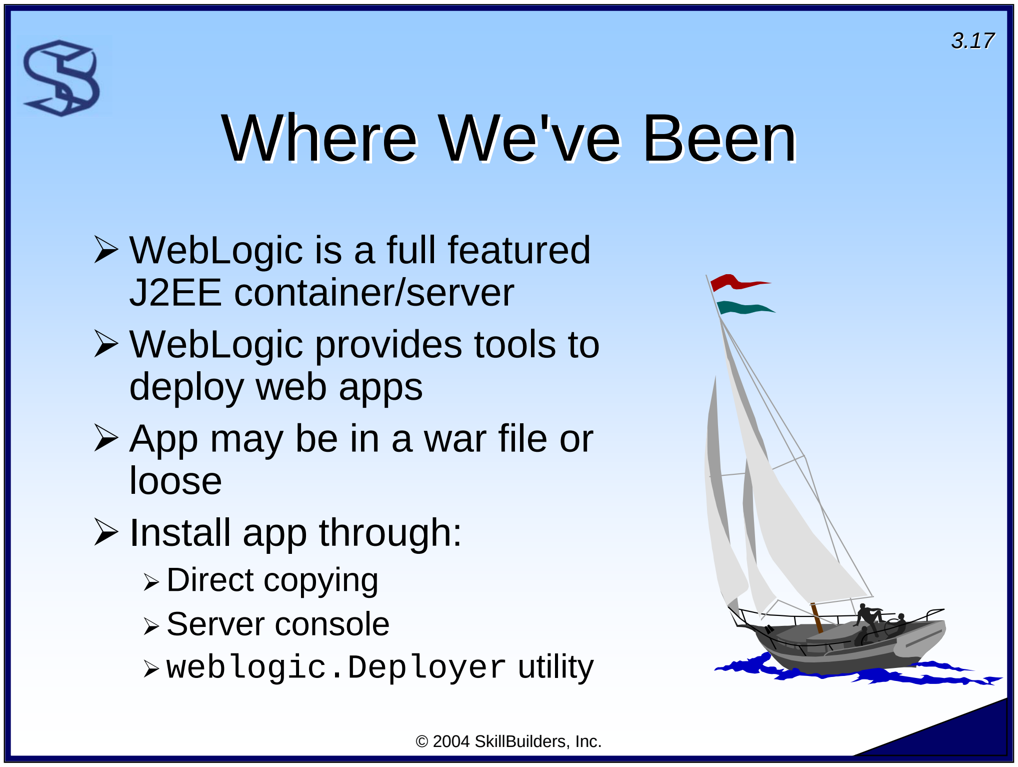
3.16

Target Servers

📖 *Target*

- A server running in the domain
- Each target typically runs on a different machine

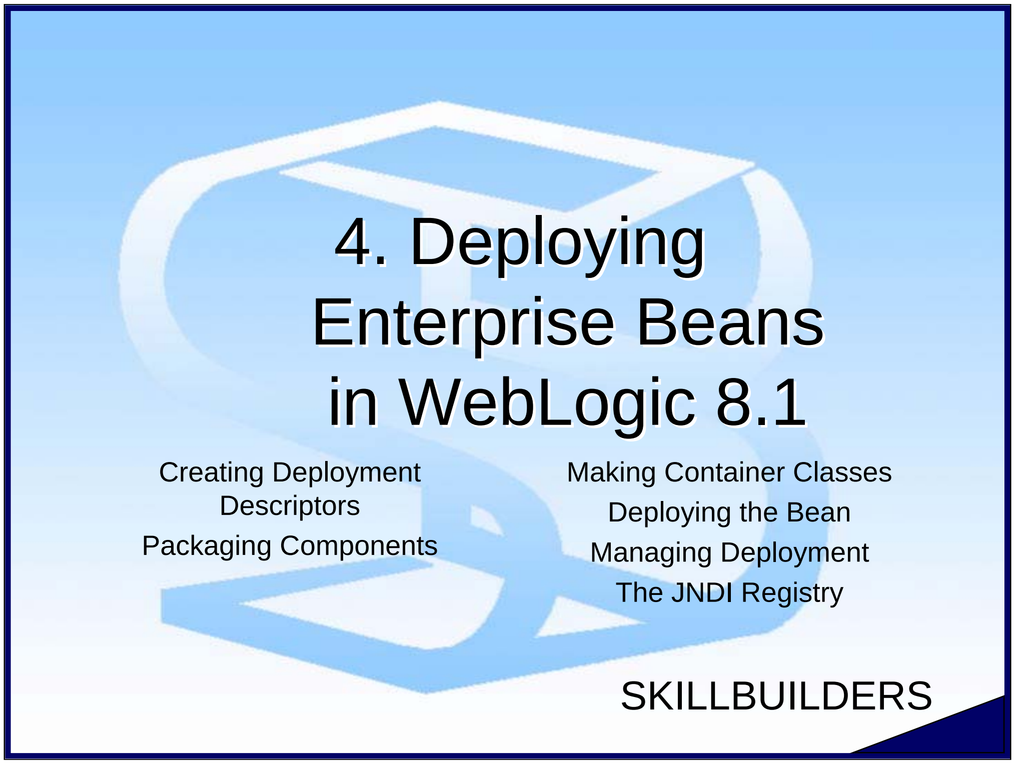
- You can manage deployment across targets
  - Deploy same component to multiple targets
  - View all targets to which a component is deployed
  - View all components deployed on a server



3.17

Where We've Been

- ➢ WebLogic is a full featured J2EE container/server
- ➢ WebLogic provides tools to deploy web apps
- ➢ App may be in a war file or loose
- ➢ Install app through:
  - ➢ Direct copying
  - ➢ Server console
  - ➢ `weblogic.Deployer` utility



4. Deploying Enterprise Beans in WebLogic 8.1

Creating Deployment Descriptors

Packaging Components

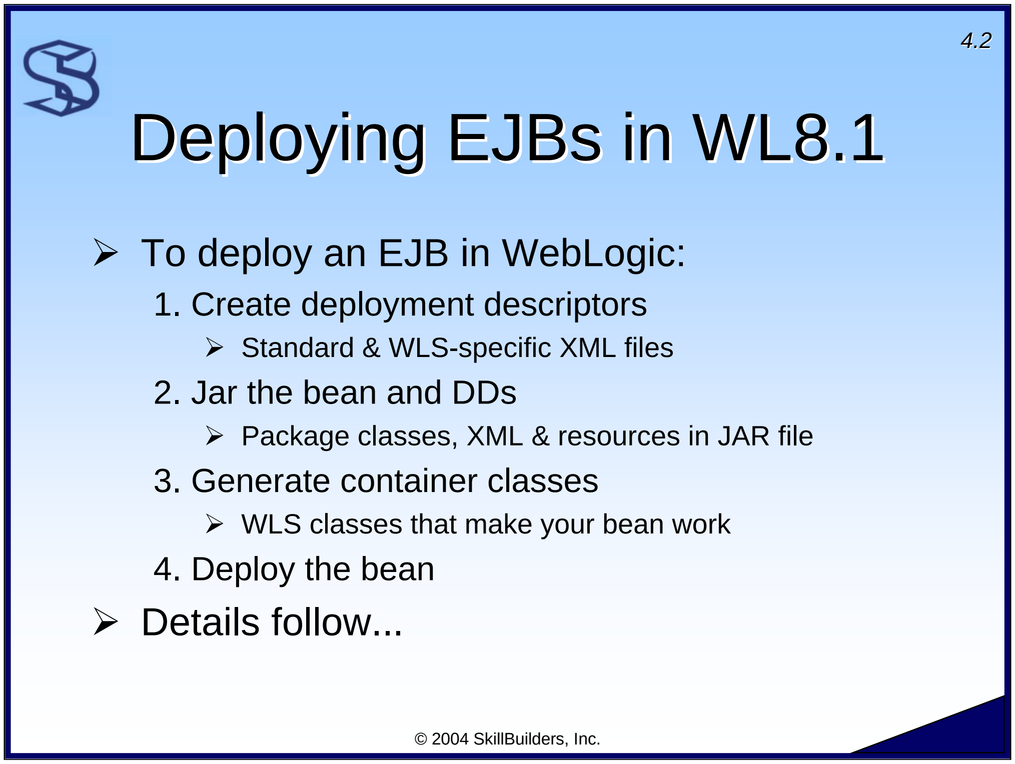
Making Container Classes

Deploying the Bean

Managing Deployment

The JNDI Registry

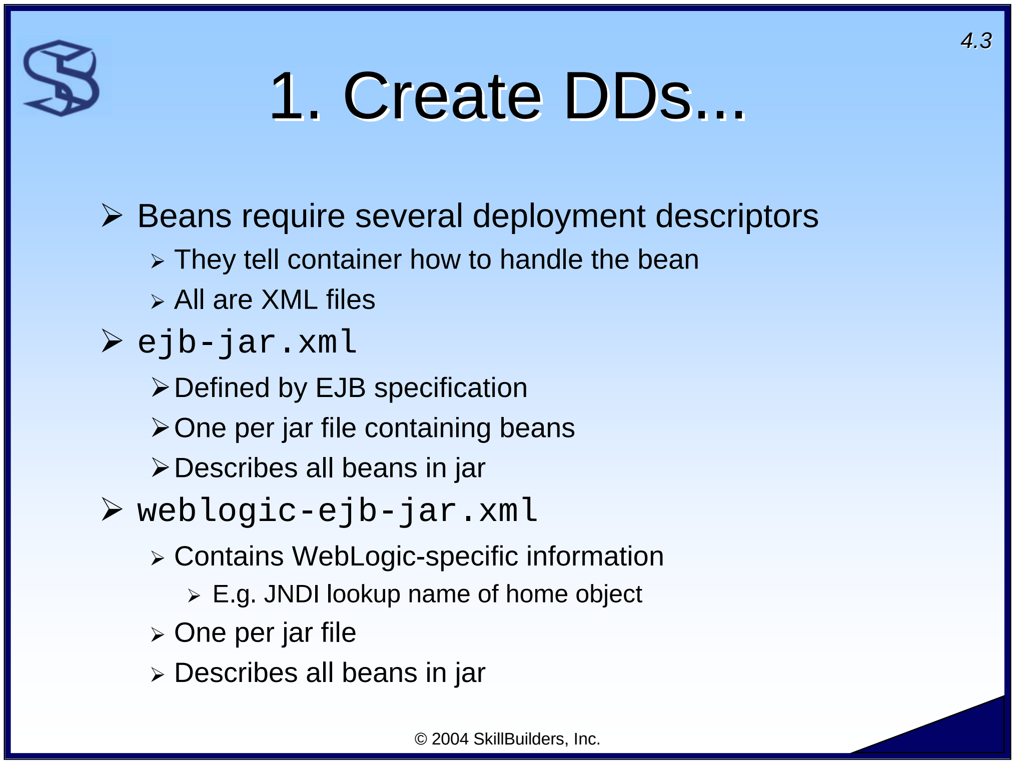
SKILLBUILDERS



4.2

Deploying EJBs in WL8.1

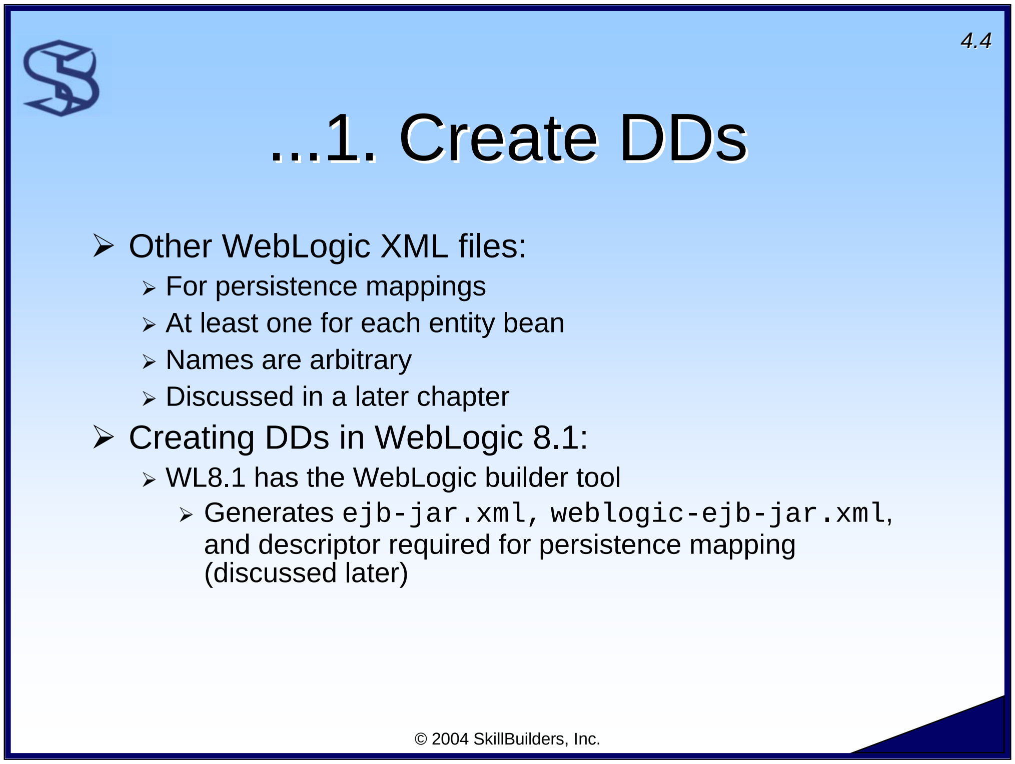
- To deploy an EJB in WebLogic:
  1. Create deployment descriptors
     - Standard & WLS-specific XML files
  2. Jar the bean and DDs
     - Package classes, XML & resources in JAR file
  3. Generate container classes
     - WLS classes that make your bean work
  4. Deploy the bean
- Details follow...



4.3

1. Create DDs...

- Beans require several deployment descriptors
  - They tell container how to handle the bean
  - All are XML files
- `ejb-jar.xml`
  - Defined by EJB specification
  - One per jar file containing beans
  - Describes all beans in jar
- `weblogic-ejb-jar.xml`
  - Contains WebLogic-specific information
    - E.g. JNDI lookup name of home object
  - One per jar file
  - Describes all beans in jar

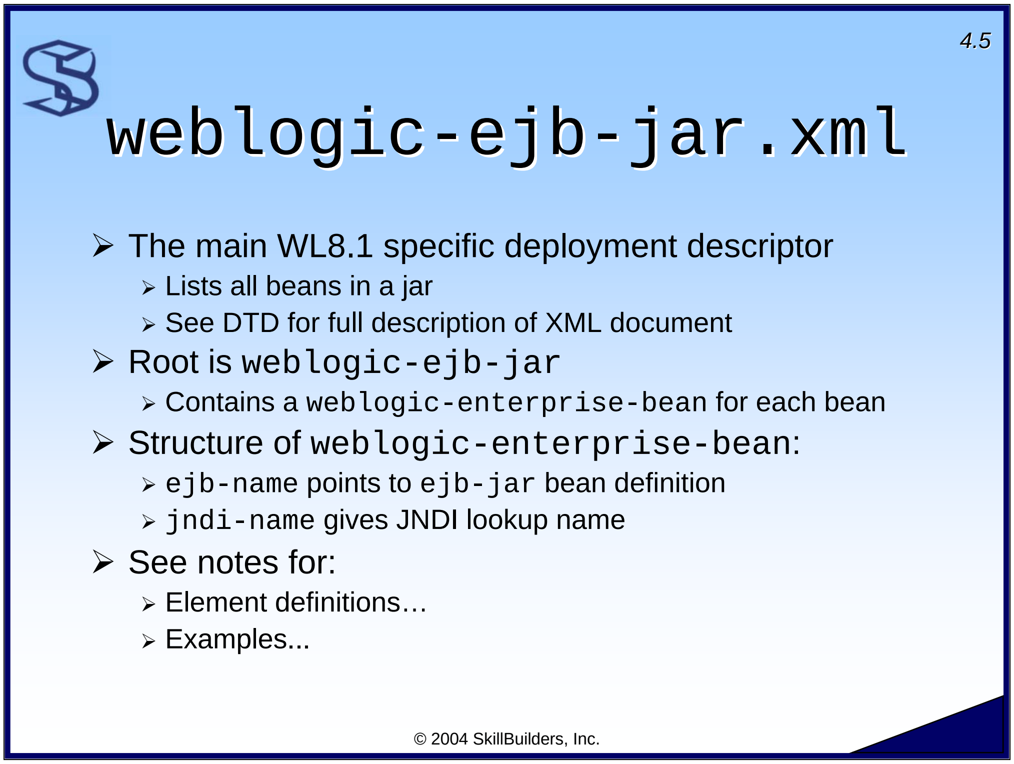


4.4

...1. Create DDs

- Other WebLogic XML files:
  - For persistence mappings
  - At least one for each entity bean
  - Names are arbitrary
  - Discussed in a later chapter
- Creating DDs in WebLogic 8.1:
  - WL8.1 has the WebLogic builder tool
    - Generates `ejb-jar.xml`, `weblogic-ejb-jar.xml`, and descriptor required for persistence mapping (discussed later)

The WebLogic 8.1 builder tool will generate deployment descriptors. To avoid writing the deployment descriptor xml files from scratch, you can use the WebLogic builder tool to generate them.



4.5

weblogic-ejb-jar.xml

- The main WL8.1 specific deployment descriptor
  - Lists all beans in a jar
  - See DTD for full description of XML document
- Root is `weblogic-ejb-jar`
  - Contains a `weblogic-enterprise-bean` for each bean
- Structure of `weblogic-enterprise-bean`:
  - `ejb-name` points to `ejb-jar` bean definition
  - `jndi-name` gives JNDI lookup name
- See notes for:
  - Element definitions…
  - Examples...

© 2004 SkillBuilders, Inc.

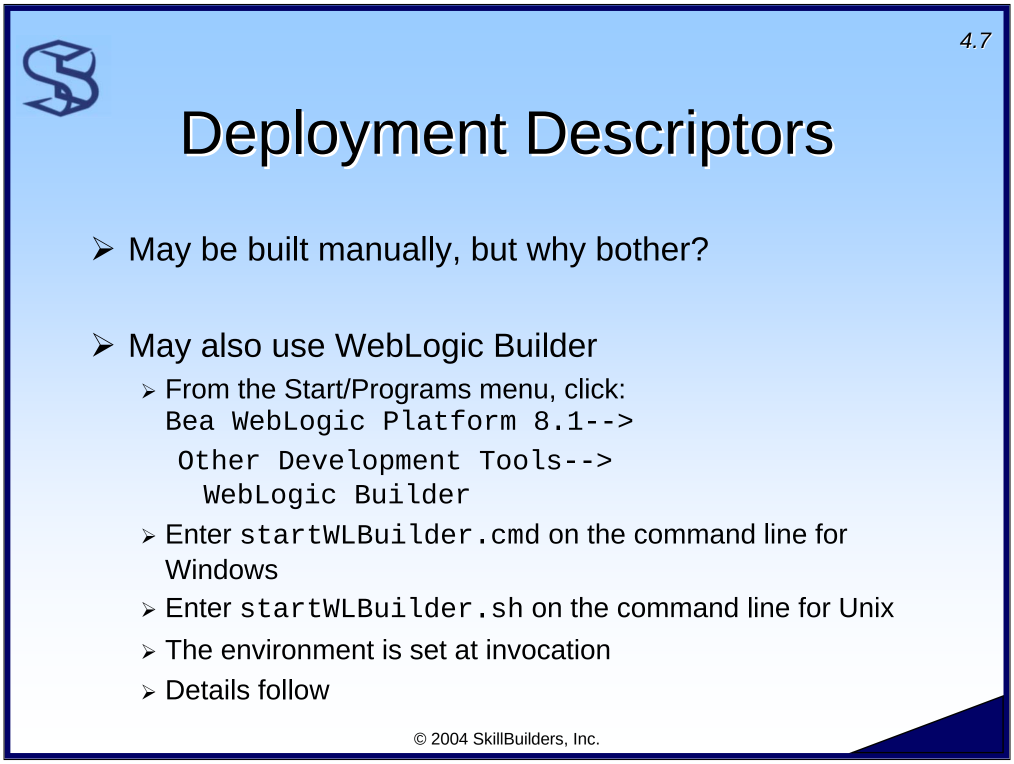
Here is the DTD definition for the root element (`weblogic-ejb-jar`) of the `weblogic- ejb-jar.xml` deployment descriptor. Note that you can get complete descriptions of the elements in the actual DTD file, which comes with WebLogic:

```
<!ELEMENT weblogic-ejb-jar (description?, weblogic-enterprise-bean*,
            security-role-assignment*,
            security-permission?,
            transaction-isolation*,
            idempotent-methods?)
```

Here is the definition for `weblogic-enterprise-bean`. Although it seems complex at first glance, notice that the only required element is `ejb-name`.

```
<!ELEMENT weblogic-enterprise-bean ( ejb-name,
 (entity-descriptor| stateless-session-descriptor| stateful-session-descriptor|
  message-driven-descriptor )?,
 transaction-descriptor?, iiop-security-descriptor?, reference-descriptor?,
 enable-call-by-reference?, clients-on-same-server?, run-as-identity-principal?,
 jndi-name?, local-jndi-name?)
```

```
<!--
Simple WebLogic 8.1 Deployment Descriptor for a ShoppingCart session bean
and related Customer bean
-->

<weblogic-ejb-jar>
   <weblogic-enterprise-bean>

      <!-- EJB name must match that in ejb-jar.xml -->
      <ejb-name>ShoppingCartBean</ejb-name>

      <!-- JNDI name is used to look up home object in WL7 -->
      <jndi-name>oursystem.ShoppingCart</jndi-name>

   </weblogic-enterprise-bean>
   <weblogic-enterprise-bean>

      <!-- EJB name must match that in ejb-jar.xml -->
      <ejb-name>CustomerBean</ejb-name>

      <!-- JNDI name is used to look up home object in WL7 -->
      <jndi-name>oursystem.Customer</jndi-name>

   </weblogic-enterprise-bean>
</weblogic-ejb-jar>
```



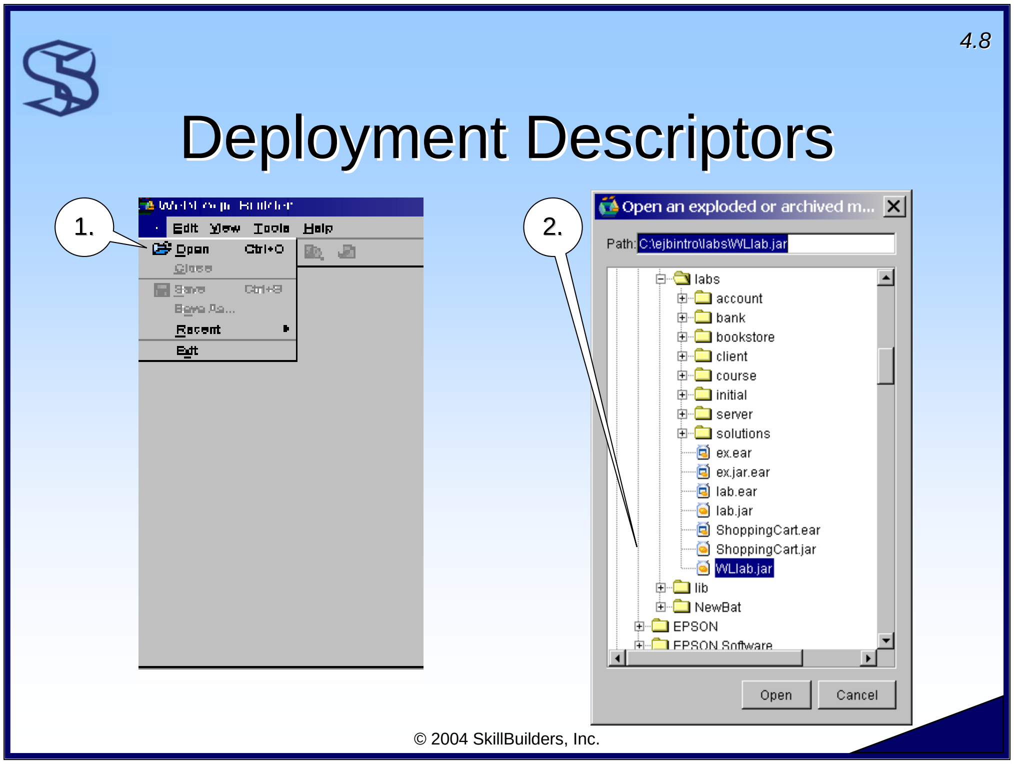
4.7

Deployment Descriptors

- May be built manually, but why bother?

- May also use WebLogic Builder
  - From the Start/Programs menu, click:
    ```
    Bea WebLogic Platform 8.1-->
     Other Development Tools-->
       WebLogic Builder
    ```
  - Enter `startWLBuilder.cmd` on the command line for Windows
  - Enter `startWLBuilder.sh` on the command line for Unix
  - The environment is set at invocation
  - Details follow

© 2004 SkillBuilders, Inc.



4.8

Deployment Descriptors

Path: C:\ejbintro\labs\WLlab.jar

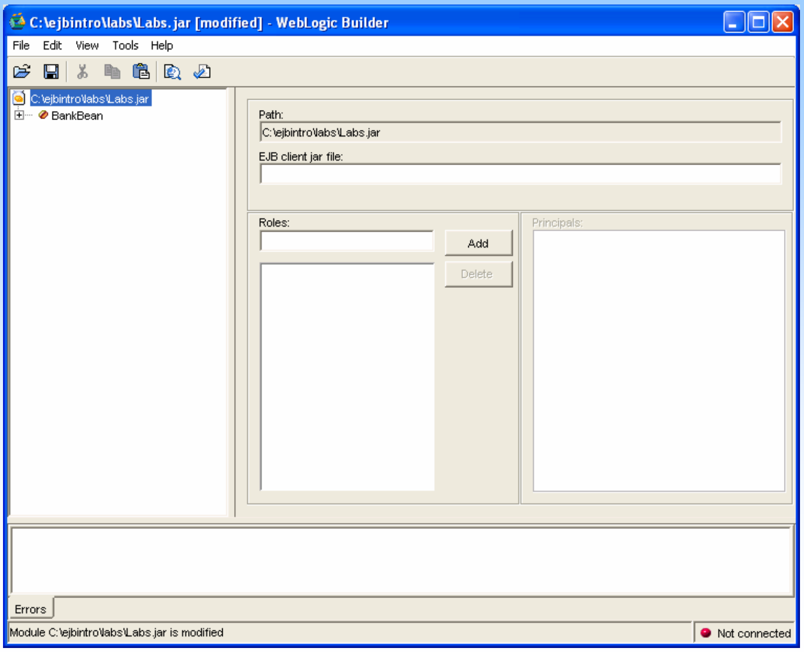
© 2004 SkillBuilders, Inc.

When the WebLogic Builder starts:

1. Click on `File` and then on `Open`
2. Select the jar file containing your EJB classes and click the `Open` button
3. If you do not already have valid deployment descriptors, a dialog box will open asking if you want new descriptors created for you. Click the `Yes` button.

1. Create DDs Screens...

4.9



C:\ejbintro\labs\Labs.jar [modified] - WebLogic Builder

Path: C:\ejbintro\labs\Labs.jar

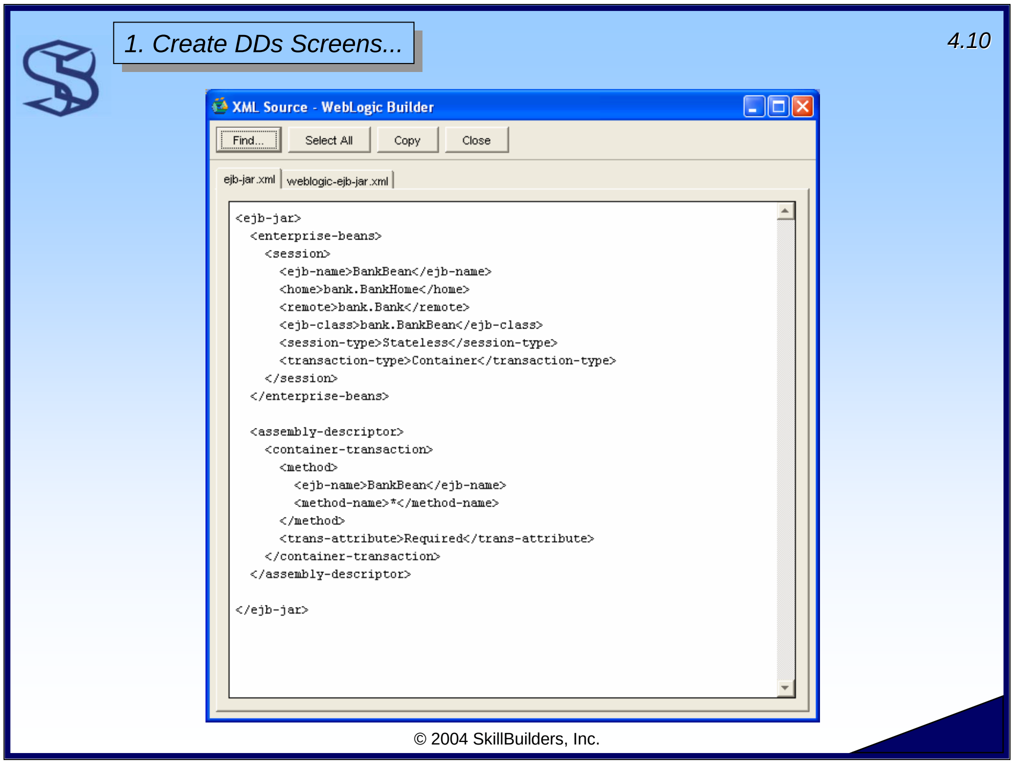
EJB client jar file:

Module C:\ejbintro\labs\Labs.jar is modified

Not connected

© 2004 SkillBuilders, Inc.

- ➢A screen similar to the one above will appear



1. Create DDs Screens...

4.10

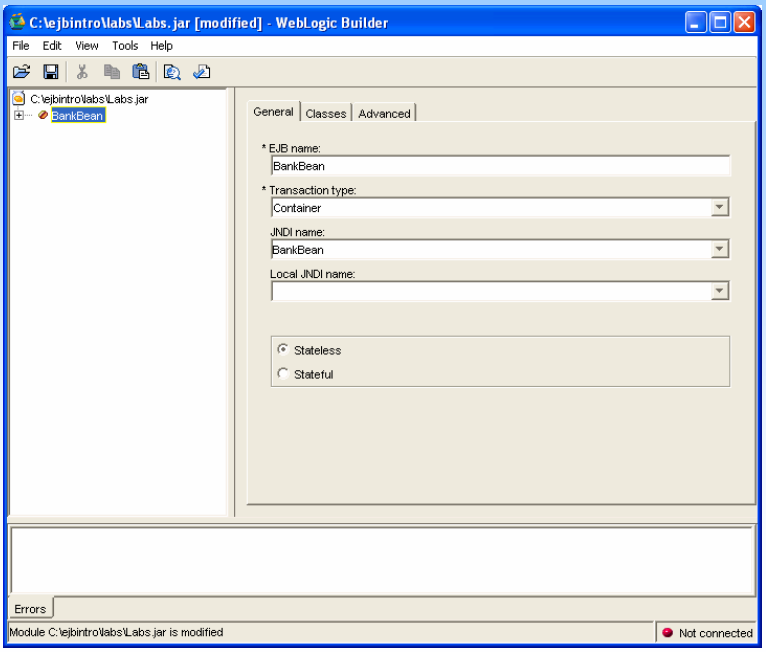
```
<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>BankBean</ejb-name>
      <home>bank.BankHome</home>
      <remote>bank.Bank</remote>
      <ejb-class>bank.BankBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>

  <assembly-descriptor>
    <container-transaction>
      <method>
        <ejb-name>BankBean</ejb-name>
        <method-name>*</method-name>
      </method>
      <trans-attribute>Required</trans-attribute>
    </container-transaction>
  </assembly-descriptor>

</ejb-jar>
```

© 2004 SkillBuilders, Inc.

To view the deployment descriptors click on `View` and then `XML Source`

- ➢ You can tab between the deployment descriptors `ejb-jar.xml` and `weblogic-ejb-jar.xml`.

1. Create DDs Screens...


C:\ejbintro\labs\Labs.jar [modified] - WebLogic Builder
File
Edit
View
Tools
Help
C:\ejbintro\labs\Labs.jar
BankBean
General
Classes
Advanced
Errors

* EJB name: BankBean

* Transaction type: Container

JNDI name: BankBean

Local JNDI name:

● Stateless

○ Stateful

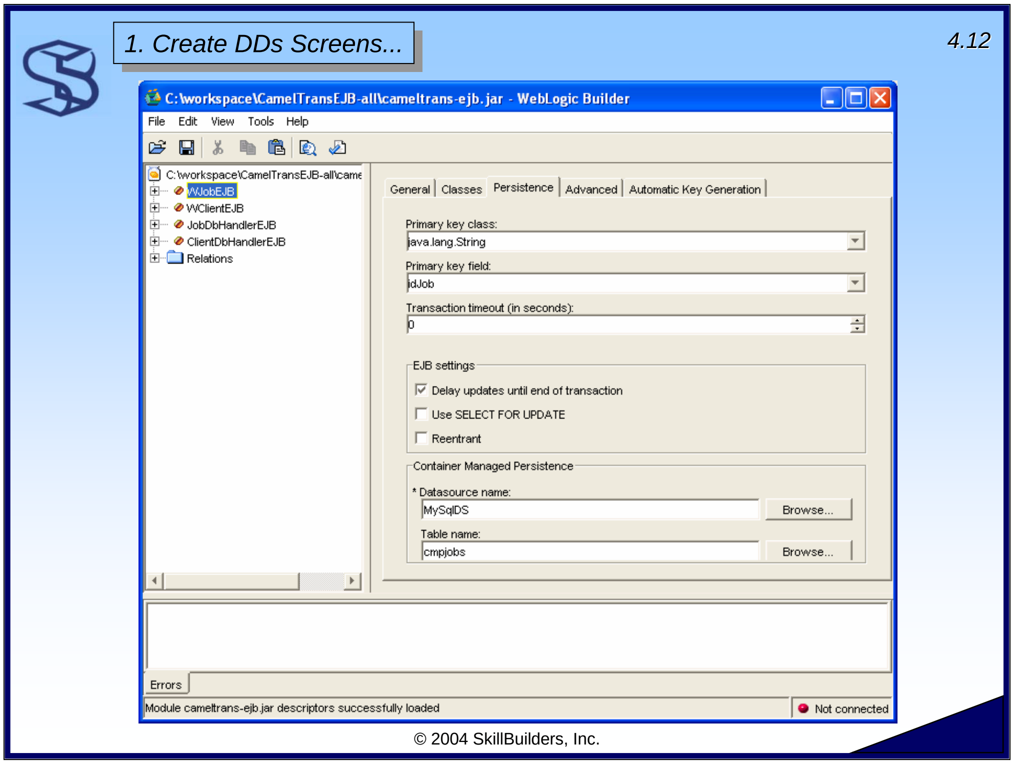
Module C:\ejbintro\labs\Labs.jar is modified

Not connected

Click on the name of your bean in the tree at the left

You can change some of the EJB's attributes at this time:

- The `ejb-name` in the deployment descriptors
- The transaction type (Bean or Container)
- The `jndi-name` of the bean
- If the bean has a local interface, the `local-jndi-name` of the bean
- The stateless or stateful attribute of a session bean



1. Create DDs Screens...

4.12

C:\workspace\CamelTransEJB-all\cameltrans-ejb.jar - WebLogic Builder

Primary key class: java.lang.String

Primary key field: idJob

Transaction timeout (in seconds): 0

EJB settings

☑ Delay updates until end of transaction

☐ Use SELECT FOR UPDATE

☐ Reentrant

Container Managed Persistence

* Datasource name: MySqlDS

Table name: cmpjobs

Module cameltrans-ejb.jar descriptors successfully loaded

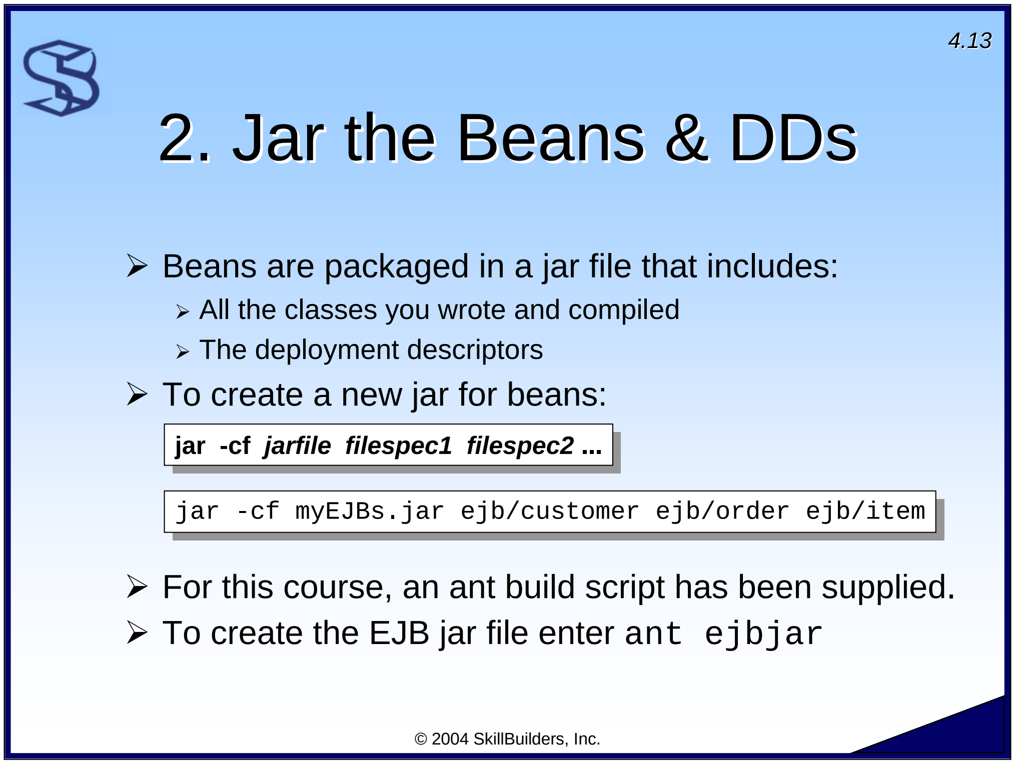
● Not connected

© 2004 SkillBuilders, Inc.

In addition to the `ejb-name`, `jndi-name`, and `local-jndi-name`, entity EJBs have some additional deployment attributes that can be changed in WebLogic builder, particularly relating to persistence:

- The `prim-key-class` (Primary key class)
- The `primkey-field` (Primary key field)
- The datasource name (for container managed persistence)
- The database table name (for container managed persistence)

When you are done setting the various attributes of your EJBs, be sure to click the “save” icon or click `File` and then `Save`, and the deployment descriptors will be saved in your jar file.



4.13

2. Jar the Beans & DDs

- Beans are packaged in a jar file that includes:
  - All the classes you wrote and compiled
  - The deployment descriptors
- To create a new jar for beans:

jar -cf *jarfile filespec1 filespec2* ...

```
jar -cf myEJBs.jar ejb/customer ejb/order ejb/item
```

- For this course, an ant build script has been supplied.
- To create the EJB jar file enter `ant ejbjar`

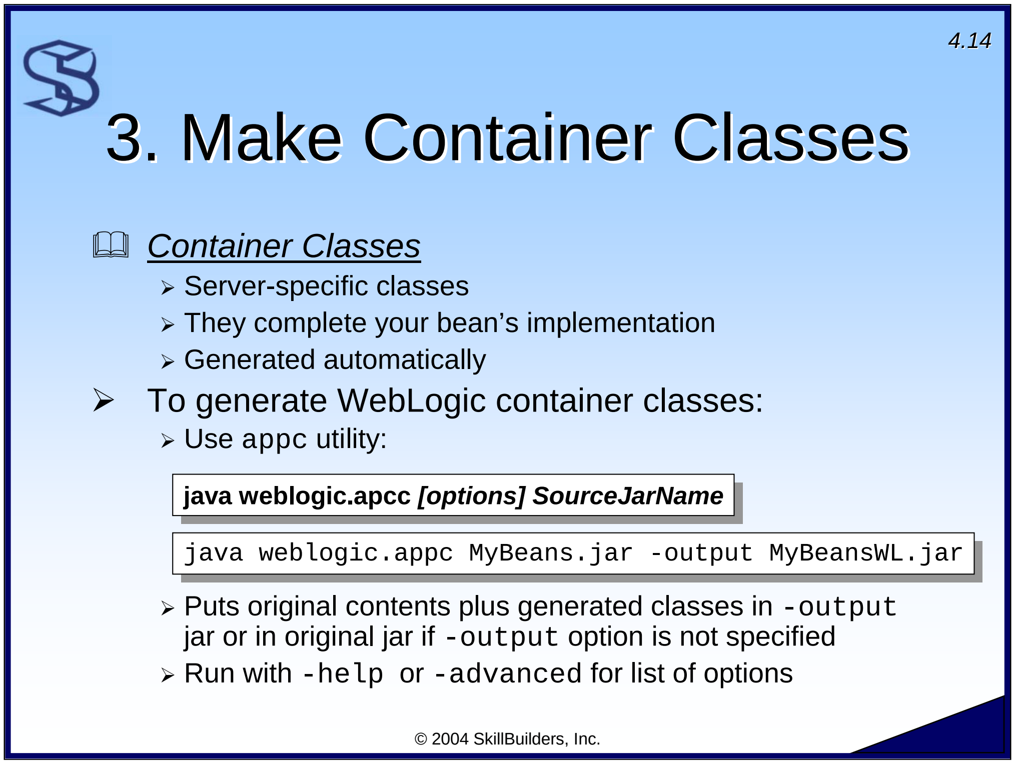
Each time you open your jar file in the WebLogic Builder tool you should save the deployment descriptors.

Use `File -> Export descriptors ...` to extract the descriptors to your working directory (“Labs” in most SkillBuilders courses).

By exporting the deployment descriptors to your working directory all your settings will remain intact each time you open your jar file in WebLogic Builder.

The ant build script provided will use the deployment descriptors located in the working directory and it will automatically place those deployment descriptors under `META-INF` in your EJB jar file. The next time you open your jar file in WebLogic Builder your settings will be intact.

If you fail to export your descriptors, you will lose the latest changes you made through the WebLogic Builder GUI.



4.14

3. Make Container Classes

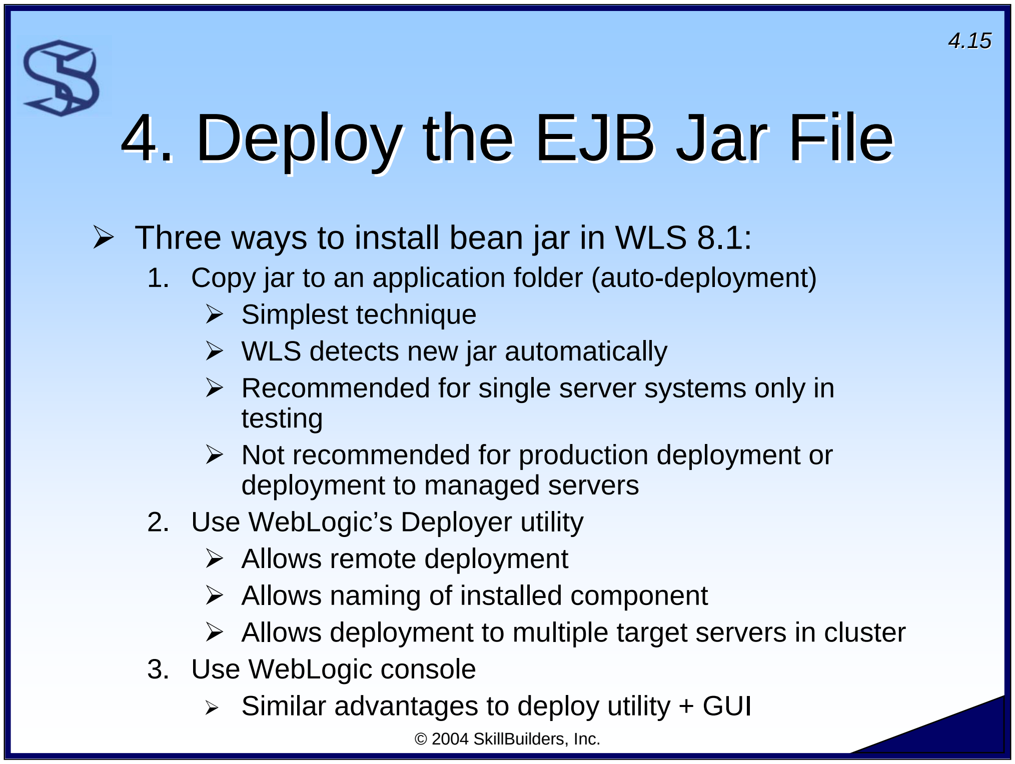
- *Container Classes*
  - Server-specific classes
  - They complete your bean's implementation
  - Generated automatically
- To generate WebLogic container classes:
  - Use `appc` utility:

java weblogic.apcc *[options] SourceJarName*

```
java weblogic.appc MyBeans.jar -output MyBeansWL.jar
```

  - Puts original contents plus generated classes in `-output` jar or in original jar if `-output` option is not specified
  - Run with `-help` or `-advanced` for list of options

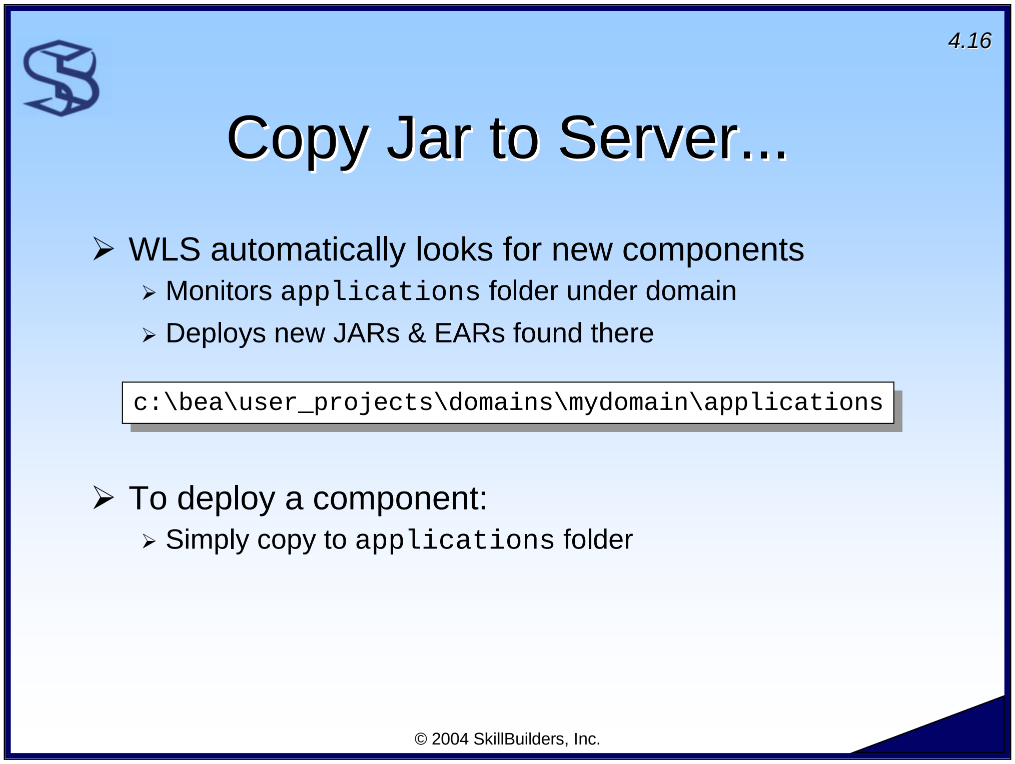
For SkillBuilders courses, `appc` will be run in an ant build script



4.15

4. Deploy the EJB Jar File

- Three ways to install bean jar in WLS 8.1:
  1. Copy jar to an application folder (auto-deployment)
     - Simplest technique
     - WLS detects new jar automatically
     - Recommended for single server systems only in testing
     - Not recommended for production deployment or deployment to managed servers
  2. Use WebLogic’s Deployer utility
     - Allows remote deployment
     - Allows naming of installed component
     - Allows deployment to multiple target servers in cluster
  3. Use WebLogic console
     - Similar advantages to deploy utility + GUI



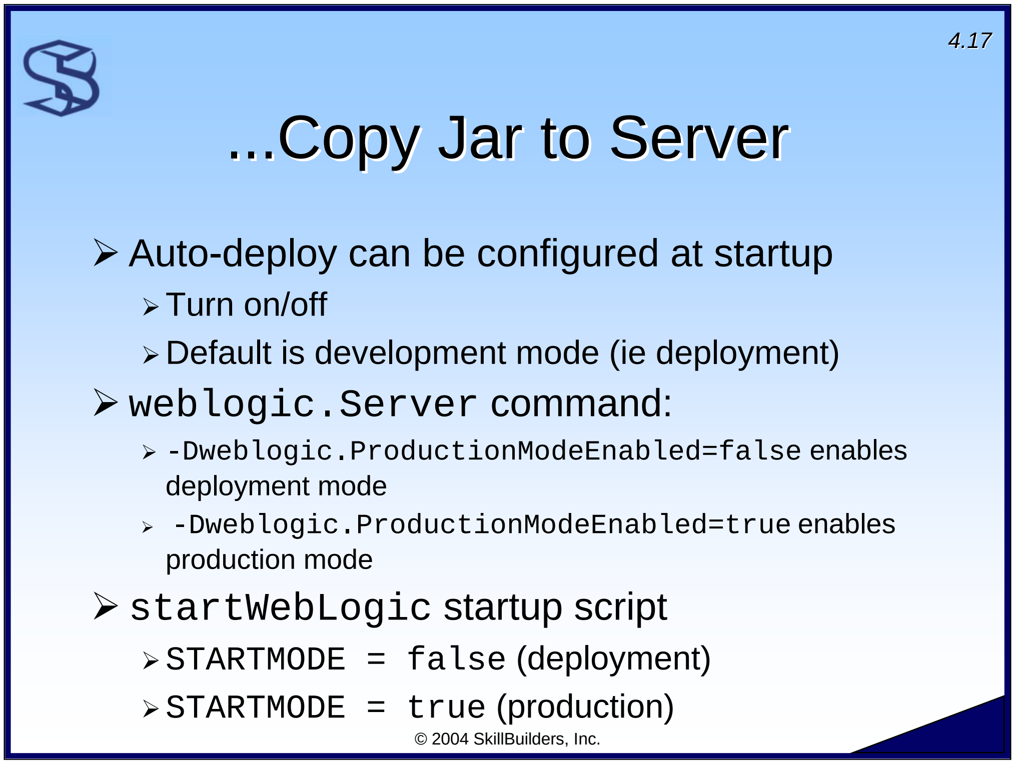
4.16

Copy Jar to Server...

- WLS automatically looks for new components
  - Monitors `applications` folder under domain
  - Deploys new JARs & EARs found there

```
c:\bea\user_projects\domains\mydomain\applications
```

- To deploy a component:
  - Simply copy to `applications` folder

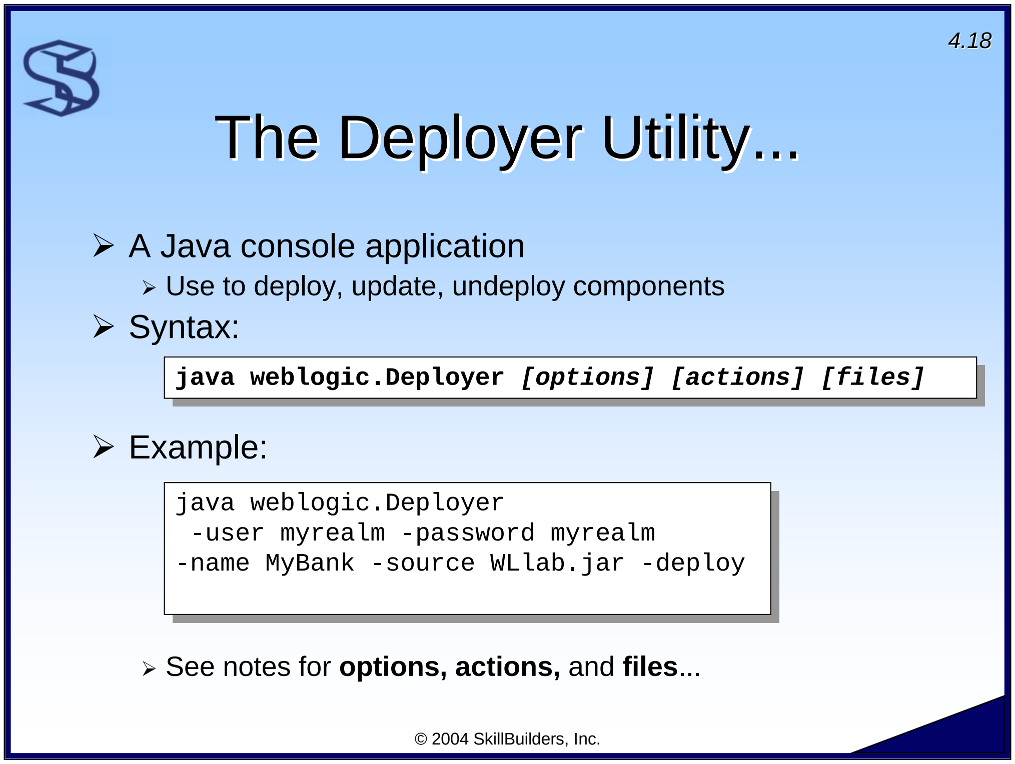


4.17

...Copy Jar to Server

- Auto-deploy can be configured at startup
  - Turn on/off
  - Default is development mode (ie deployment)
- `weblogic.Server` command:
  - `-Dweblogic.ProductionModeEnabled=false` enables deployment mode
  - `-Dweblogic.ProductionModeEnabled=true` enables production mode
- `startWebLogic` startup script
  - `STARTMODE = false` (deployment)
  - `STARTMODE = true` (production)

© 2004 SkillBuilders, Inc.



4.18

The Deployer Utility...

- A Java console application
  - Use to deploy, update, undeploy components
- Syntax:

```
java weblogic.Deployer [options] [actions] [files]
```

- Example:

```
java weblogic.Deployer
 -user myrealm -password myrealm
-name MyBank -source WLlab.jar -deploy
```

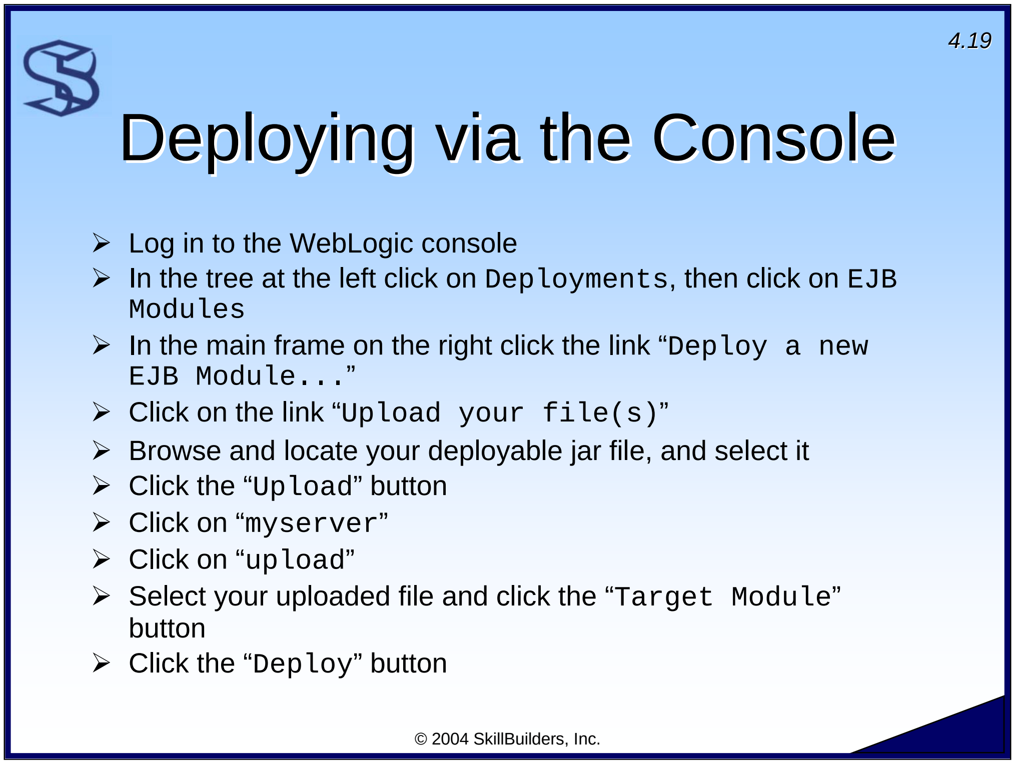
  - See notes for **options, actions,** and **files**...

ACTIONS:

- **`-deploy`** Deploys or redeploys the application (use with `-source` option)
- **`-undeploy`** Deactivates the application (use with `-source` option)
- **`-examples`** Shows tool usage examples
- **`-advanced`** Shows various actions
- **`-stop`** Makes a deployed application unavailable (use with `-name` option)
- **`-start`** Makes a deployed application available (use with `-name` option)
- See the WebLogic documentation for more complete information

OPTIONS:

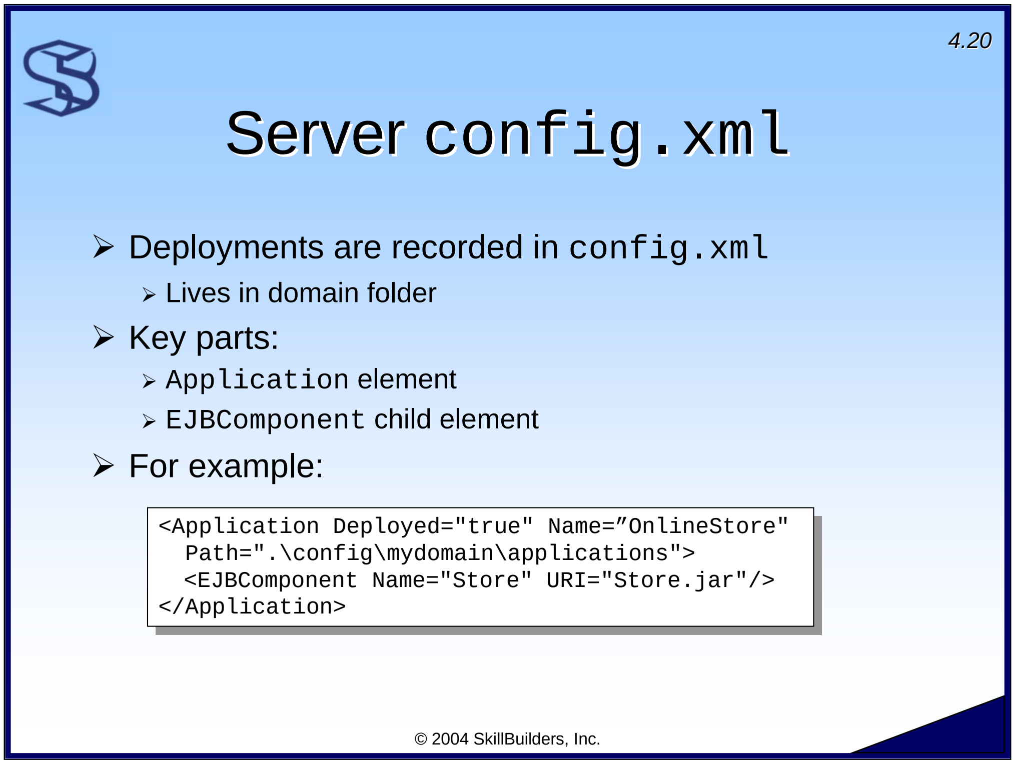
- **`-name`** Name of the application being deployed; itdefaults to the base name of the deployment file
- **`-user`** User name
- **`-password`** Password for above user
- **`-source`** Location of jar file or directory (may be relative or absolute)
- **`-debug`** Display messages
- See the WebLogic documentation for more complete information



4.19

Deploying via the Console

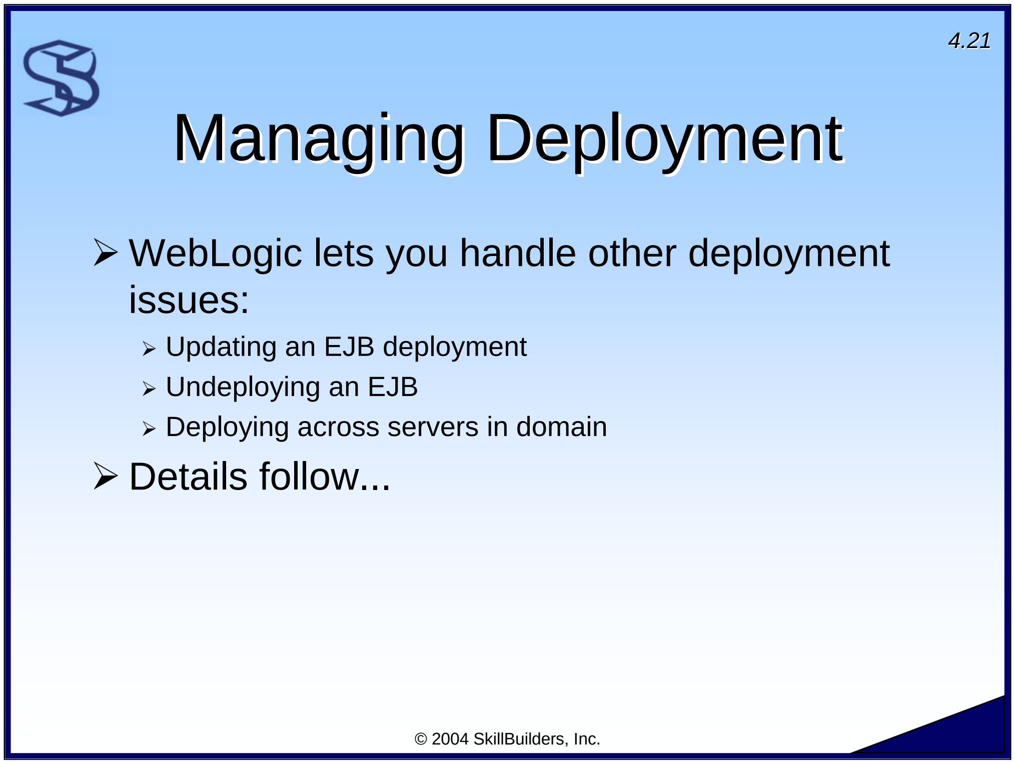
- Log in to the WebLogic console
- In the tree at the left click on `Deployments`, then click on `EJB Modules`
- In the main frame on the right click the link “`Deploy a new EJB Module...`”
- Click on the link “`Upload your file(s)`”
- Browse and locate your deployable jar file, and select it
- Click the “`Upload`” button
- Click on “`myserver`”
- Click on “`upload`”
- Select your uploaded file and click the “`Target Module`” button
- Click the “`Deploy`” button



Server config.xml

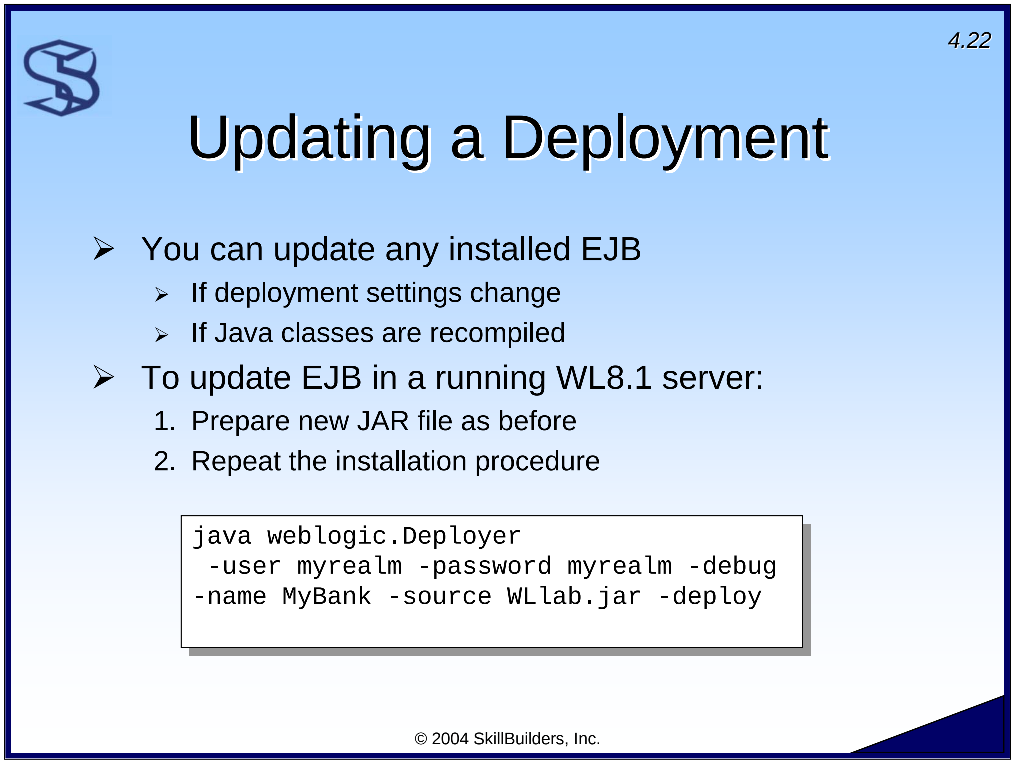
- Deployments are recorded in `config.xml`
  - Lives in domain folder
- Key parts:
  - `Application` element
  - `EJBComponent` child element
- For example:

```
<Application Deployed="true" Name=”OnlineStore"
  Path=".\config\mydomain\applications">
  <EJBComponent Name="Store" URI="Store.jar"/>
</Application>
```



Managing Deployment

- WebLogic lets you handle other deployment issues:
  - Updating an EJB deployment
  - Undeploying an EJB
  - Deploying across servers in domain
- Details follow...

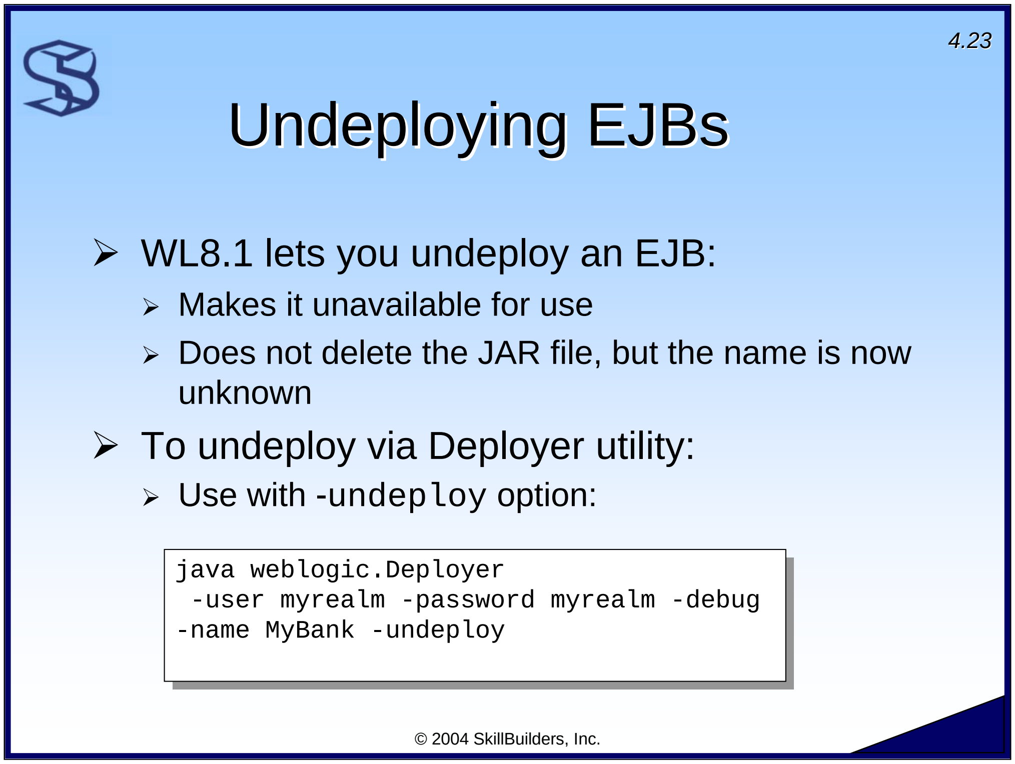


4.22

Updating a Deployment

- You can update any installed EJB
  - If deployment settings change
  - If Java classes are recompiled
- To update EJB in a running WL8.1 server:
  1. Prepare new JAR file as before
  2. Repeat the installation procedure

```
java weblogic.Deployer
 -user myrealm -password myrealm -debug
-name MyBank -source WLlab.jar -deploy
```

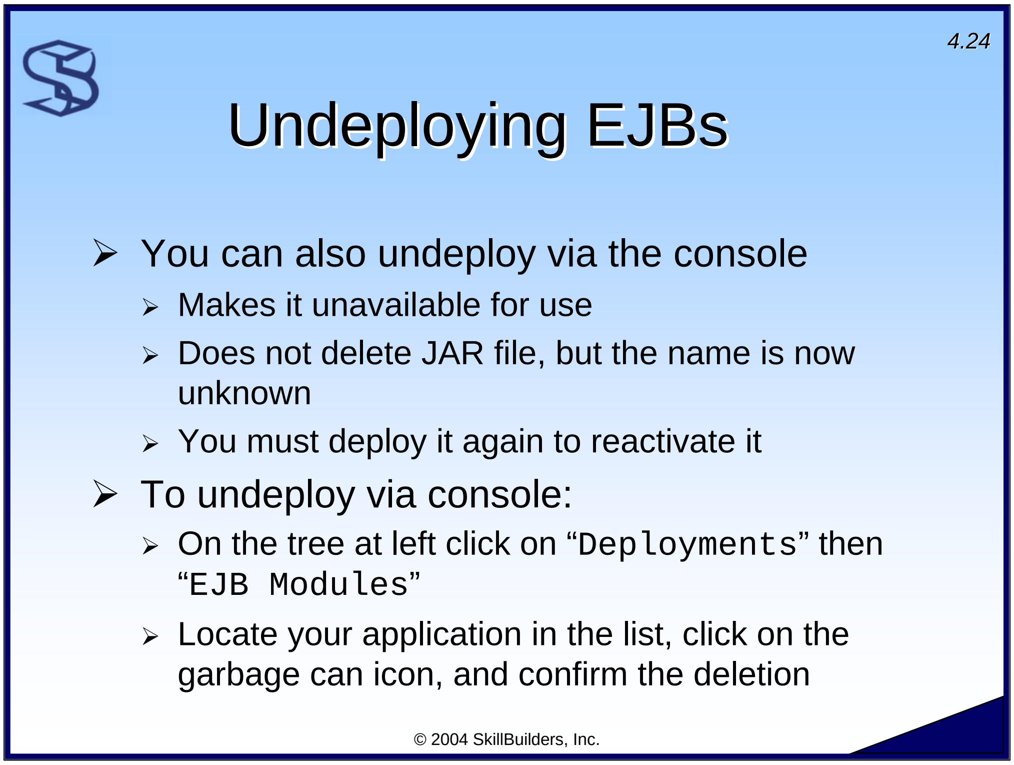


4.23

Undeploying EJBs

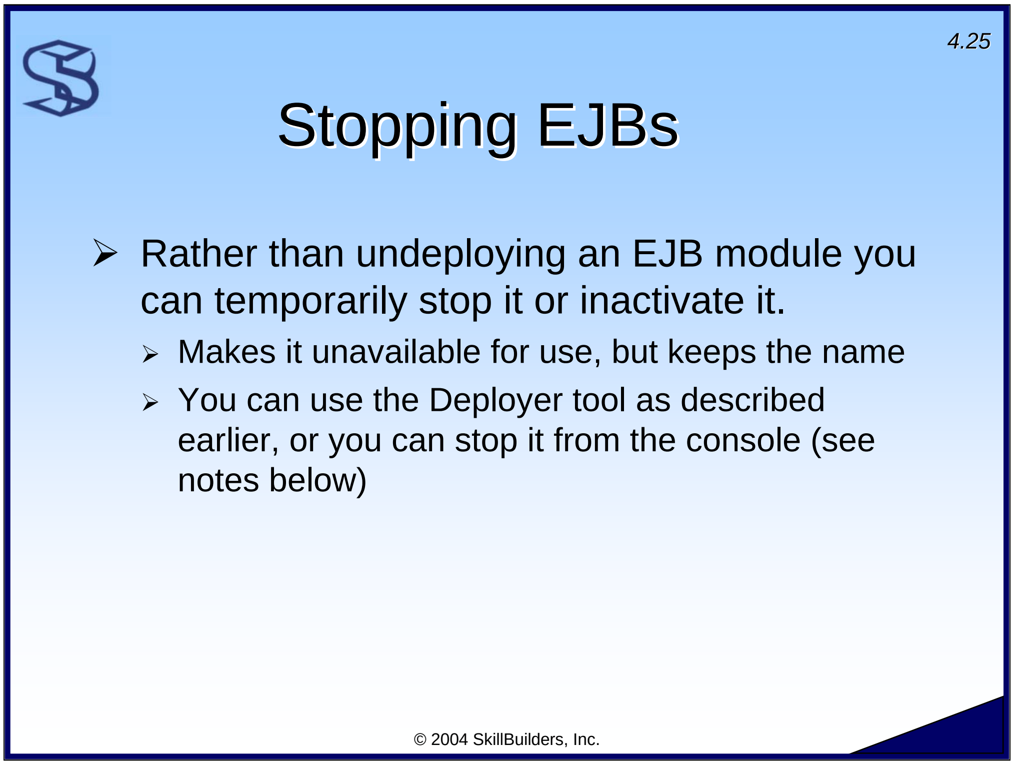
- WL8.1 lets you undeploy an EJB:
  - Makes it unavailable for use
  - Does not delete the JAR file, but the name is now unknown
- To undeploy via Deployer utility:
  - Use with `-undeploy` option:

```
java weblogic.Deployer
 -user myrealm -password myrealm -debug
-name MyBank -undeploy
```



Undeploying EJBs

- You can also undeploy via the console
  - Makes it unavailable for use
  - Does not delete JAR file, but the name is now unknown
  - You must deploy it again to reactivate it
- To undeploy via console:
  - On the tree at left click on “`Deployments`” then “`EJB Modules`”
  - Locate your application in the list, click on the garbage can icon, and confirm the deletion



4.25

Stopping EJBs

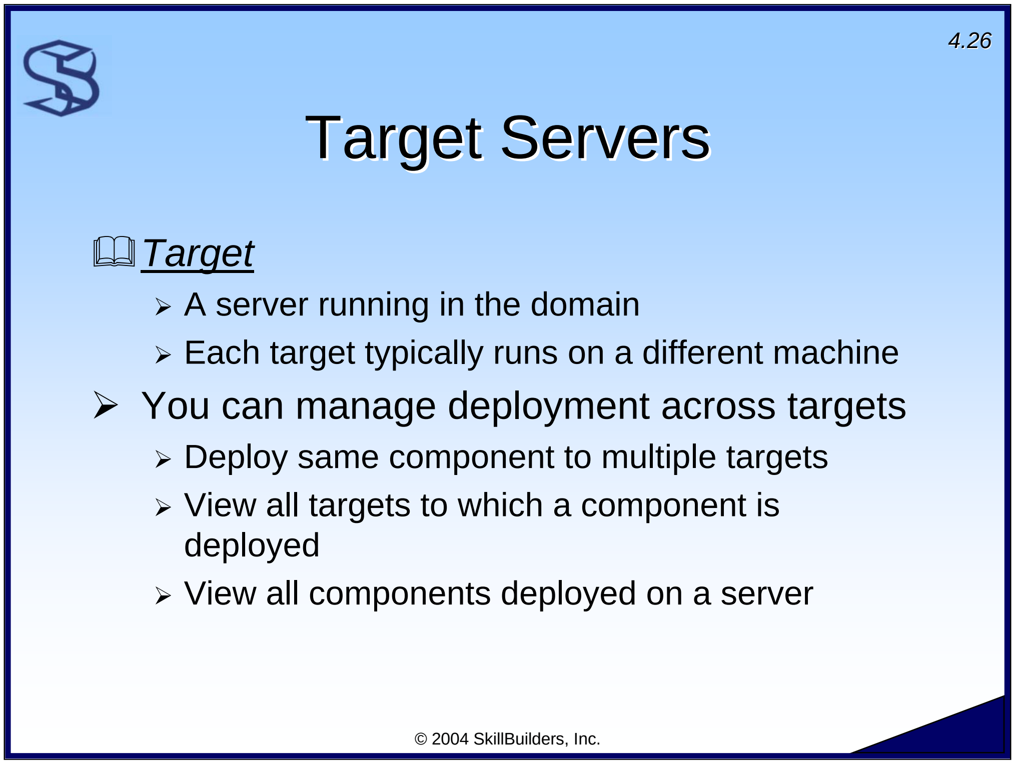
- Rather than undeploying an EJB module you can temporarily stop it or inactivate it.
  - Makes it unavailable for use, but keeps the name
  - You can use the Deployer tool as described earlier, or you can stop it from the console (see notes below)

To stop an EJB module from the console:

- Click on “`Deployments`” then “`EJB Modules`”
- Click on your EJB module
- On the main screen on the right, click the “`stop`” button

To reactivate it:

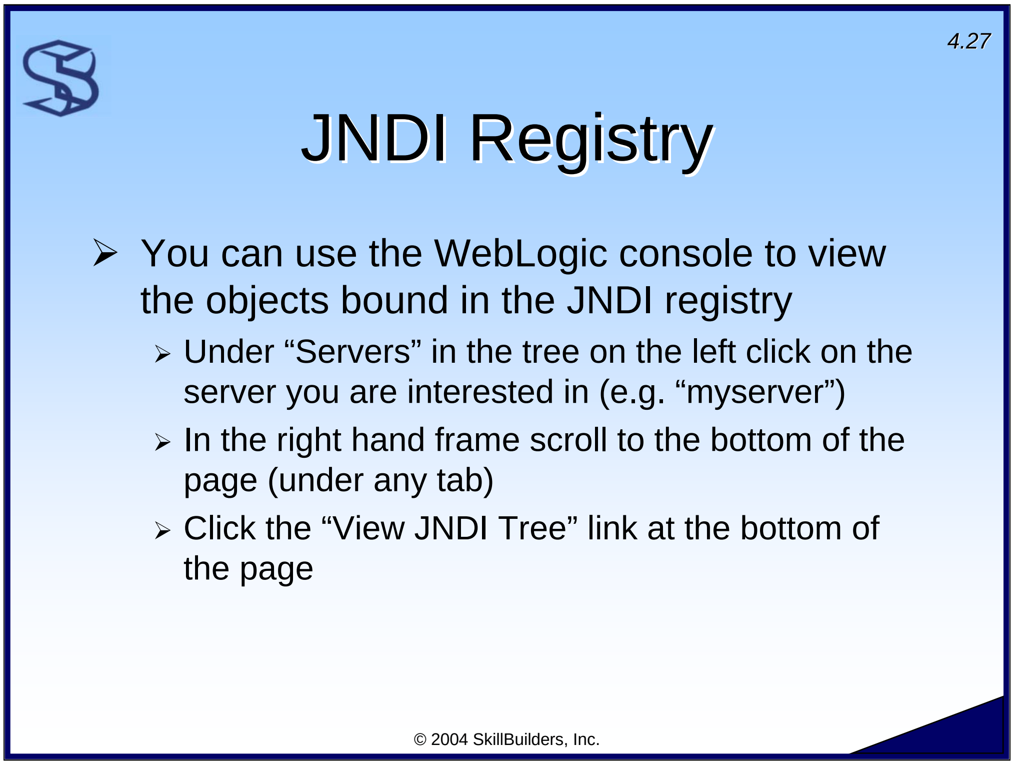
- Click on “`Deployments`”, then “`EJB Modules`”
- Click on your EJB module
- Click on the “`Deploy`” button



4.26

Target Servers

- 📖 *Target*
  - A server running in the domain
  - Each target typically runs on a different machine
- You can manage deployment across targets
  - Deploy same component to multiple targets
  - View all targets to which a component is deployed
  - View all components deployed on a server



4.27

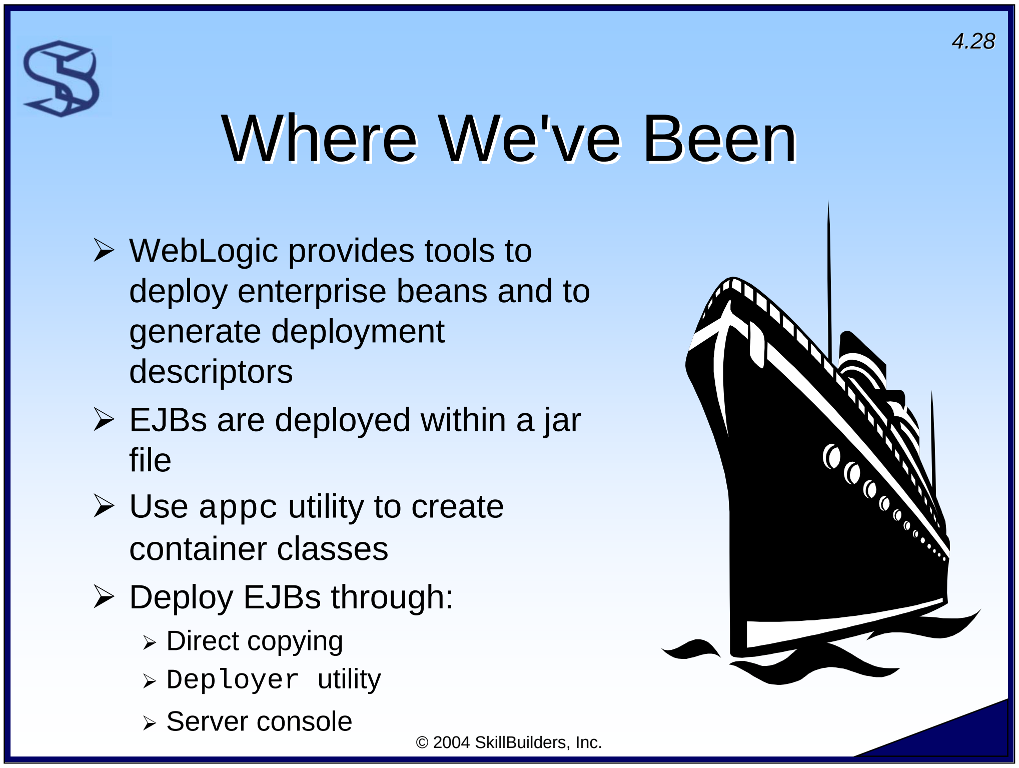
JNDI Registry

- You can use the WebLogic console to view the objects bound in the JNDI registry
  - Under “Servers” in the tree on the left click on the server you are interested in (e.g. “myserver”)
  - In the right hand frame scroll to the bottom of the page (under any tab)
  - Click the “View JNDI Tree” link at the bottom of the page

In order for a client to look up objects in the WebLogic JNDI registry, at least two pieces of information are required. They are:

Property Name	*Property Value*
java.naming.factory.initial	**weblogic.jndi.WLInitialContextFactory**
java.naming.provider.url	**t3://localhost:7001**

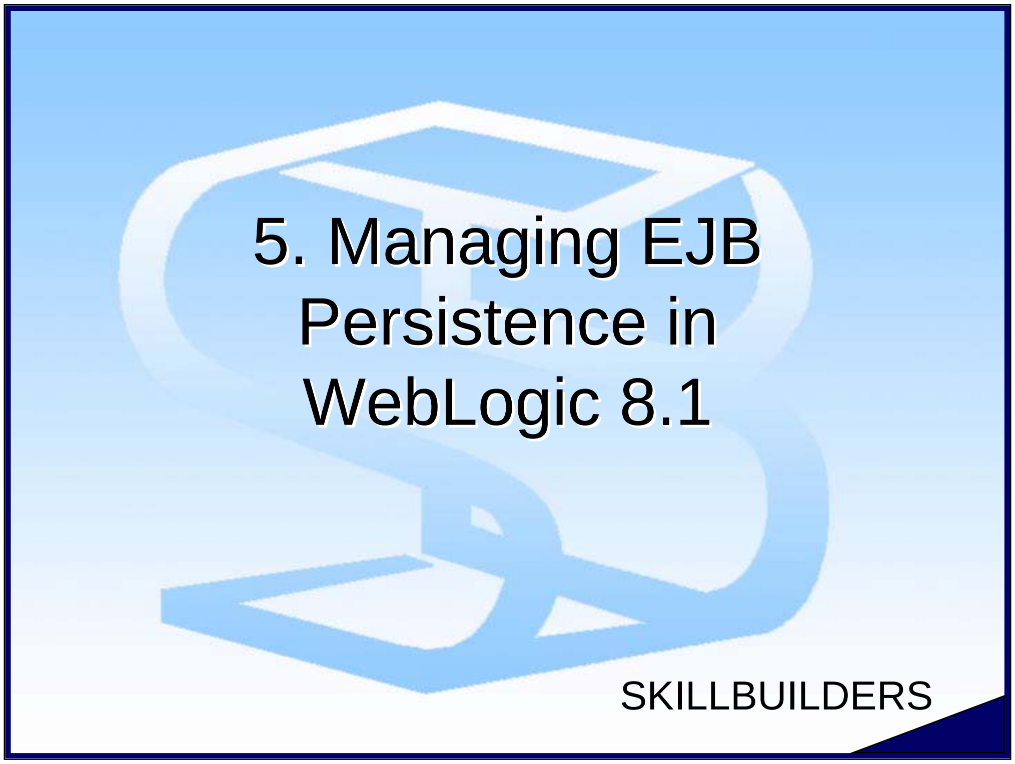
These values should be set as System properties. Or, at run time, you can create a `Hashtable` with these values and pass it as an argument when you get the JNDI `InitialContext`. You can also put these values in a file called “`jndi.properties`”.



4.28

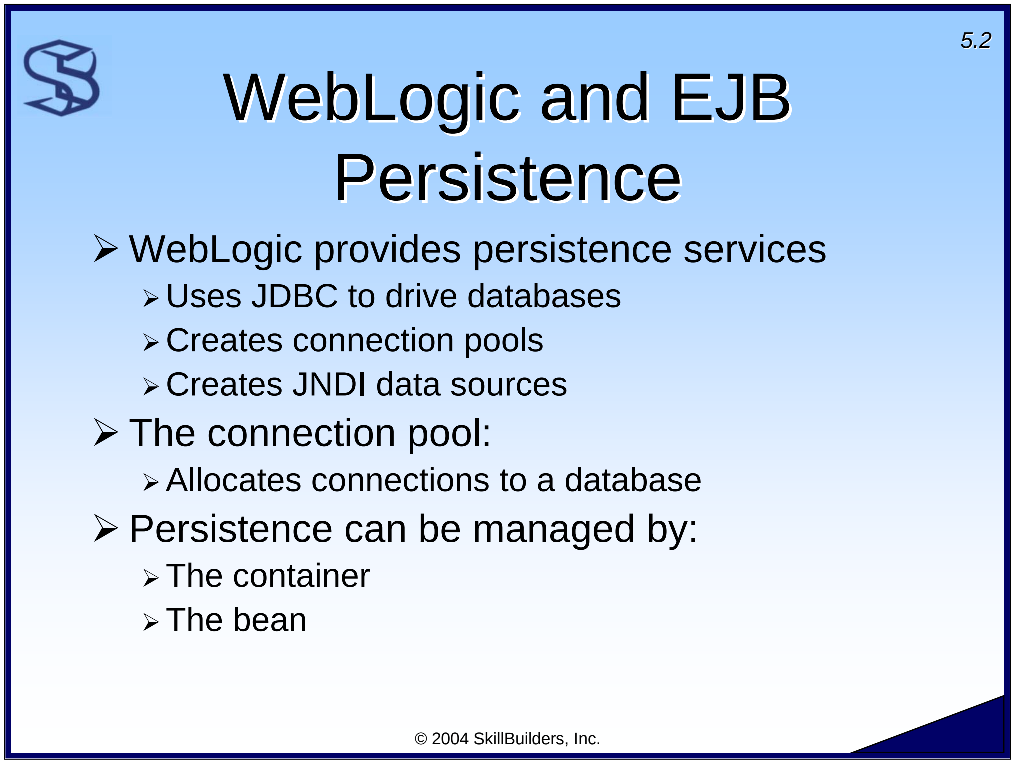
Where We've Been

- WebLogic provides tools to deploy enterprise beans and to generate deployment descriptors
- EJBs are deployed within a jar file
- Use `appc` utility to create container classes
- Deploy EJBs through:
  - Direct copying
  - `Deployer` utility
  - Server console



5. Managing EJB Persistence in WebLogic 8.1

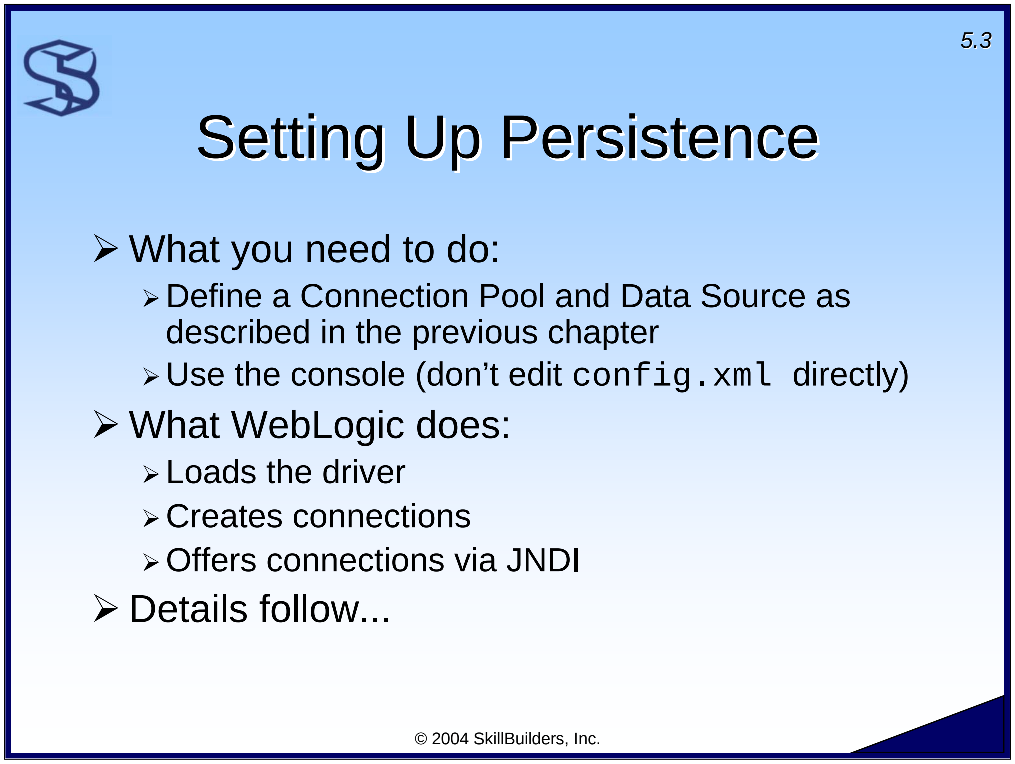
SKILLBUILDERS



5.2

WebLogic and EJB Persistence

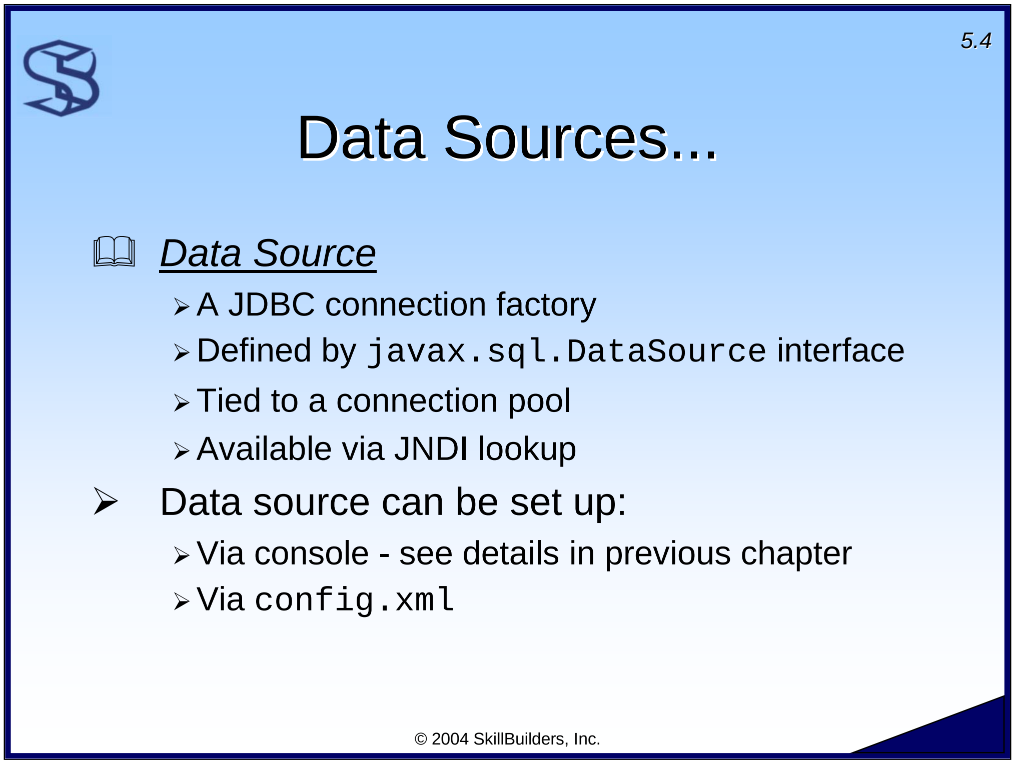
- WebLogic provides persistence services
  - Uses JDBC to drive databases
  - Creates connection pools
  - Creates JNDI data sources
- The connection pool:
  - Allocates connections to a database
- Persistence can be managed by:
  - The container
  - The bean



5.3

Setting Up Persistence

- What you need to do:
  - Define a Connection Pool and Data Source as described in the previous chapter
  - Use the console (don’t edit `config.xml` directly)
- What WebLogic does:
  - Loads the driver
  - Creates connections
  - Offers connections via JNDI
- Details follow...



5.4

Data Sources...

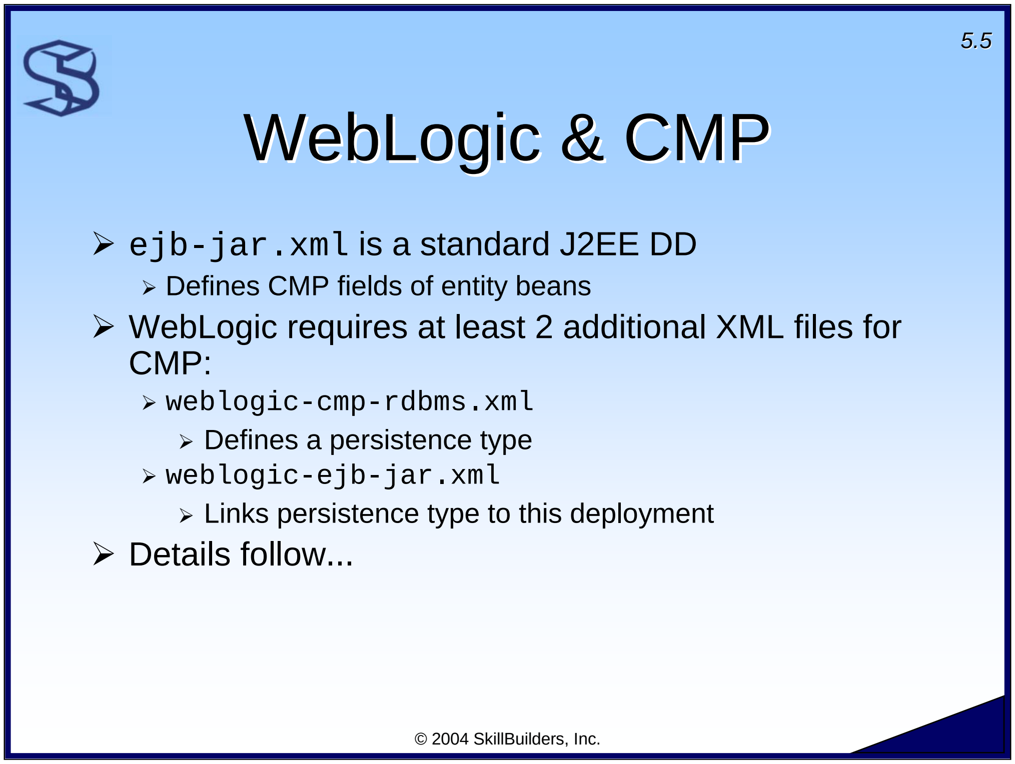
📖 *Data Source*

- A JDBC connection factory
- Defined by `javax.sql.DataSource` interface
- Tied to a connection pool
- Available via JNDI lookup

➢ Data source can be set up:

- Via console - see details in previous chapter
- Via `config.xml`

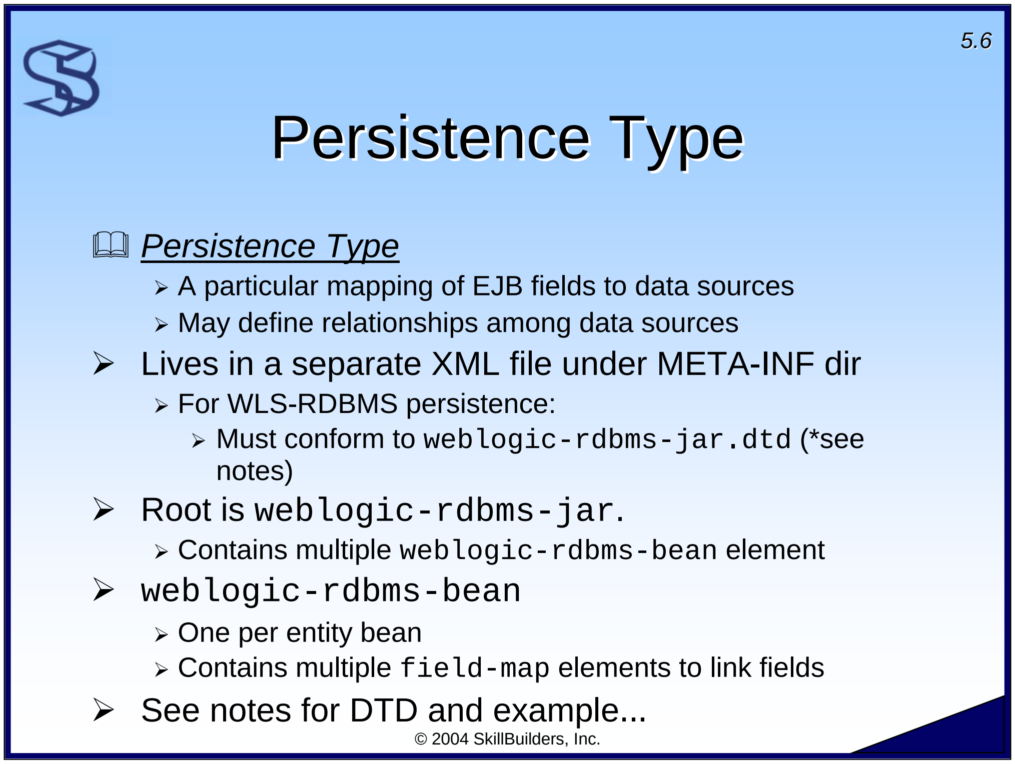
Previous versions of WebLogic used a “TxDataSource” instead of a DataSource for Container Managed Persistence. WebLogic version 8.1 no longer uses “TxDataSource” in the console GUI (it is still maintained internally in `config.xml`).



5.5

WebLogic & CMP

- `ejb-jar.xml` is a standard J2EE DD
  - Defines CMP fields of entity beans
- WebLogic requires at least 2 additional XML files for CMP:
  - `weblogic-cmp-rdbms.xml`
    - Defines a persistence type
  - `weblogic-ejb-jar.xml`
    - Links persistence type to this deployment
- Details follow...



5.6

Persistence Type

- *Persistence Type*
  - A particular mapping of EJB fields to data sources
  - May define relationships among data sources
- Lives in a separate XML file under META-INF dir
  - For WLS-RDBMS persistence:
    - Must conform to `weblogic-rdbms-jar.dtd` (*see notes)
- Root is `weblogic-rdbms-jar`.
  - Contains multiple `weblogic-rdbms-bean` element
- `weblogic-rdbms-bean`
  - One per entity bean
  - Contains multiple `field-map` elements to link fields
- See notes for DTD and example...

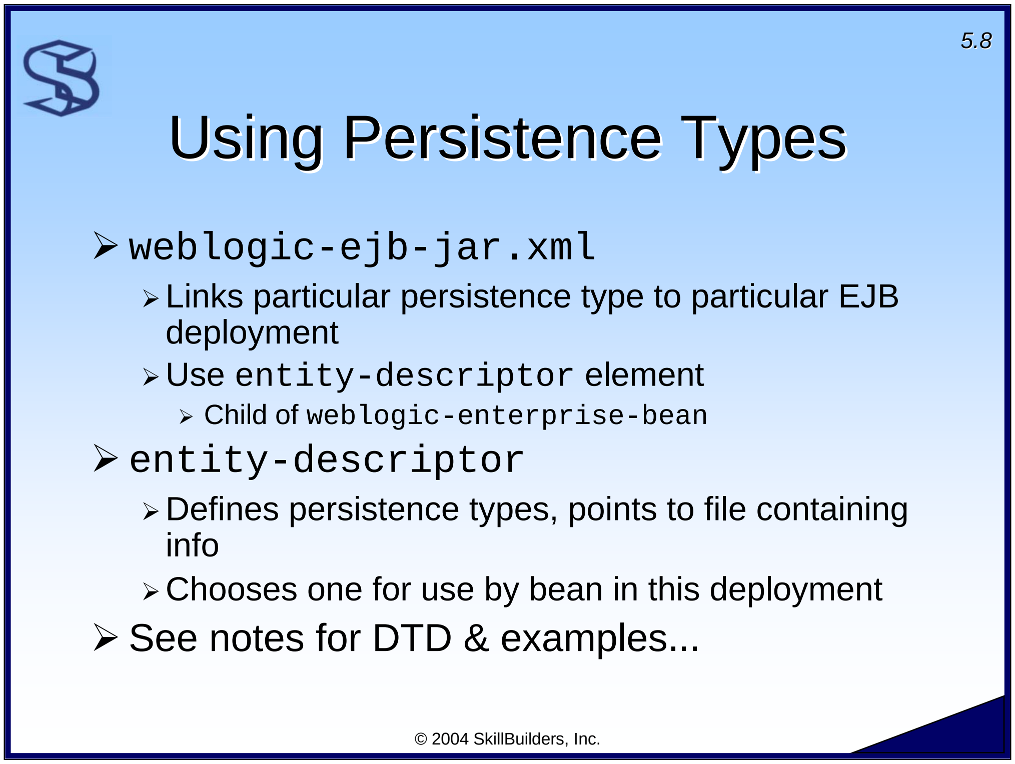
When deploying a CMP entity bean, you must include an additional deployment file for the bean's persistence type. For WebLogic Server RDBMS-based persistence services, the file is generally named `weblogic-cmp-rdbms-jar.xml`. If you use a third-party persistence vendor, the file type as well as its contents may be different from `weblogic-cmp-rdbms-jar.xml`; refer to the vendor's documentation for details. See `xxx.dtd` files in `weblogic.jar` for further details.

Here are the key element definitions for a WebLogic file:

```
<!ELEMENT weblogic-rdbms-jar (
  weblogic-rdbms-bean+,weblogic-rdbms-relation*, create-default-dbms-tables?,
    validate-db-schema-with?,database-type?)>
<!ELEMENT weblogic-rdbms-bean (
    ejb-name,data-source-name,table-map+,field-group*,relationship-caching*,
    weblogic-query*,delay-database-insert-until?,automatic-key-generation?,
    check-exists-on-method?)>
<!ELEMENT ejb-name (#PCDATA)>
<!ELEMENT data-source-name (#PCDATA)>
<!ELEMENT table-map (
    table-name,field-map*,verify-columns?,optimistic-column?)>
<!ELEMENT table-name (#PCDATA)>
<!ELEMENT field-map (
    cmp-field,dbms-column, dbms-column-type?,group-name?)>
```

Here is a sample `weblogic-rdbms-jar` file that maps CMP fields in the `Customer` bean to fields in the data source `oursystem.database`.

```
<!DOCTYPE weblogic-rdbms-jar PUBLIC
 '-// BEA Systems, Inc.//DTD WebLogic 7.0.0 EJB RDBMS Persistence//EN'
 'http://www.bea.com/servers/wls700/dtd/weblogic-rdbms20-persistence-
700.dtd'


<weblogic-rdbms-jar>
   <weblogic-rdbms-bean>
      <!-- ejb name to match ejb-jar.xml -->
      <ejb-name>CustomerBean</ejb-name>

      <!-- JNDI name of data-source -->
      <data-source-name>oursystem.database</data-source-name>

      <!-- Table Map -->
      <table-map>
         <!-- Table name -->
         <table-name>customers</table-name>

         <!-- Field mapping -->
         <field-map>
            <cmp-field>id</cmp-field>
            <dbms-column>custnum</dbms-column>
         </field-map>
         <field-map>
            <cmp-field>name</cmp-field>
            <dbms-column>name</dbms-column>
         </field-map>
         <field-map>
            <cmp-field>status</cmp-field>
            <dbms-column>status</dbms-column>
         </field-map>
      </table-map>
   </weblogic-rdbms-bean>

   <weblogic-rdbms-bean>
      <!-- Another mapping... -->
   </weblogic-rdbms-bean>
</weblogic-rdbms-jar>
```



5.8

Using Persistence Types

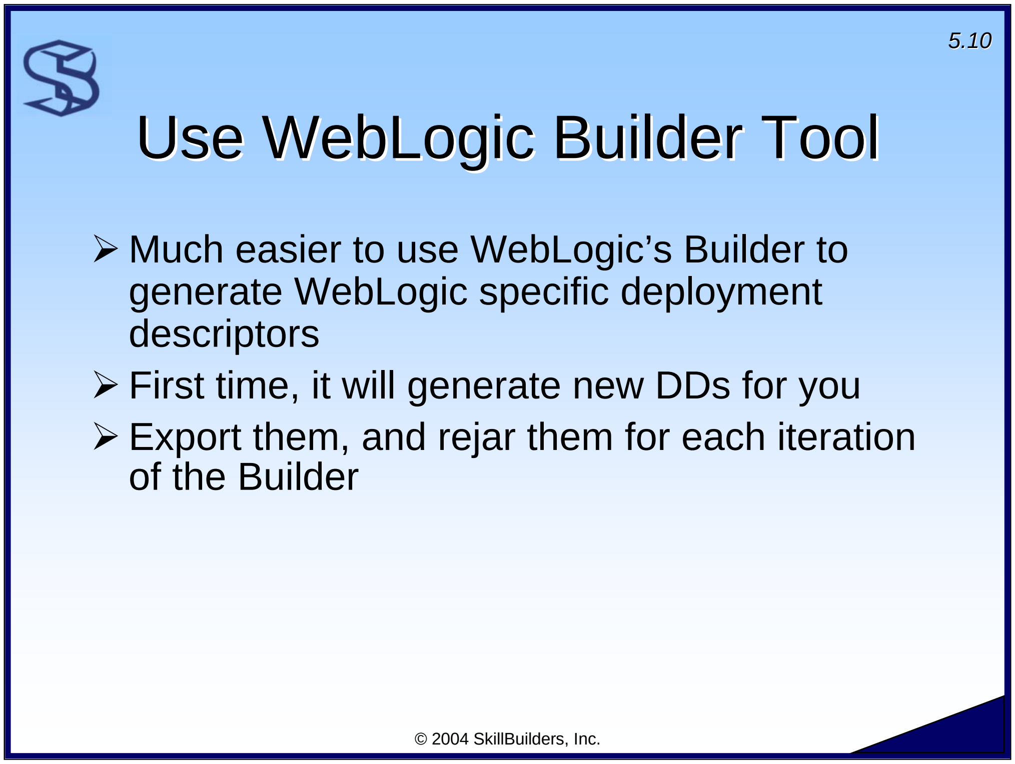
- `weblogic-ejb-jar.xml`
  - Links particular persistence type to particular EJB deployment
  - Use `entity-descriptor` element
    - Child of `weblogic-enterprise-bean`
- `entity-descriptor`
  - Defines persistence types, points to file containing info
  - Chooses one for use by bean in this deployment
- See notes for DTD & examples...

© 2004 SkillBuilders, Inc.

Here are the relevant definitions of `weblogic-ejb-jar.dtd` defining the parts of `weblogic-ejb-jar.xml` that pertain to CMP entity beans:

```
<!ELEMENT entity-descriptor (
  pool?, (entity-cache | entity-cache-ref)?, persistence?,entity-
clustering?,
  invalidation-target?,enable-dynamic-queries?)>
<!ELEMENT persistence (
    is-modified-method-name?,delay-updates-until-end-of-tx?,finders-load-
bean?,
    persistence-use?)>
<!ELEMENT persistence-use (type-identifier, type-version, type-storage)>
```

```
<?xml version="1.0"?>

<!DOCTYPE weblogic-ejb-jar PUBLIC
  '-//BEA Systems, Inc.//DTD WebLogic 7.0.0 EJB//EN'
'http://www.bea.com/servers/wls700/dtd/weblogic-ejb-jar.dtd'
>

<weblogic-ejb-jar>
   <weblogic-enterprise-bean>

      <!-- EJB name must match that in ejb-jar.xml -->
      <ejb-name>ShoppingCartBean</ejb-name>

      <!-- JNDI name is used to look up home object in WL6 -->
      <jndi-name>oursystem.ShoppingCart</jndi-name>

   </weblogic-enterprise-bean>
   <weblogic-enterprise-bean>

      <!-- EJB name must match that in ejb-jar.xml -->
      <ejb-name>CustomerBean</ejb-name>

      <entity-descriptor>
         <persistence>
            <persistence-use>
               <type-identifier>WebLogic_CMP_RDBMS
               </type-identifier>
               <type-version>6.0</type-version>
            </persistence-use>
         </persistence>
      </entity-descriptor>

      <!-- JNDI name is used to look up home object in WL6 -->
      <jndi-name>oursystem.Customer</jndi-name>
    </weblogic-enterprise-bean>
</weblogic-ejb-jar>
```



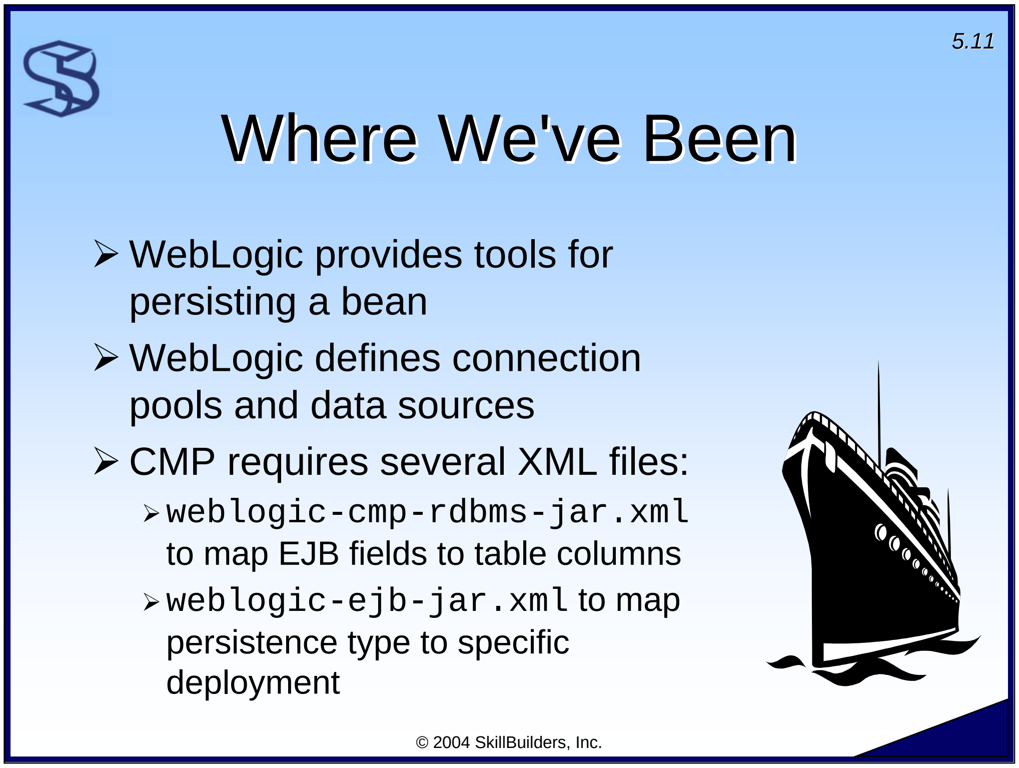
5.10

Use WebLogic Builder Tool

- Much easier to use WebLogic’s Builder to generate WebLogic specific deployment descriptors
- First time, it will generate new DDs for you
- Export them, and rejar them for each iteration of the Builder

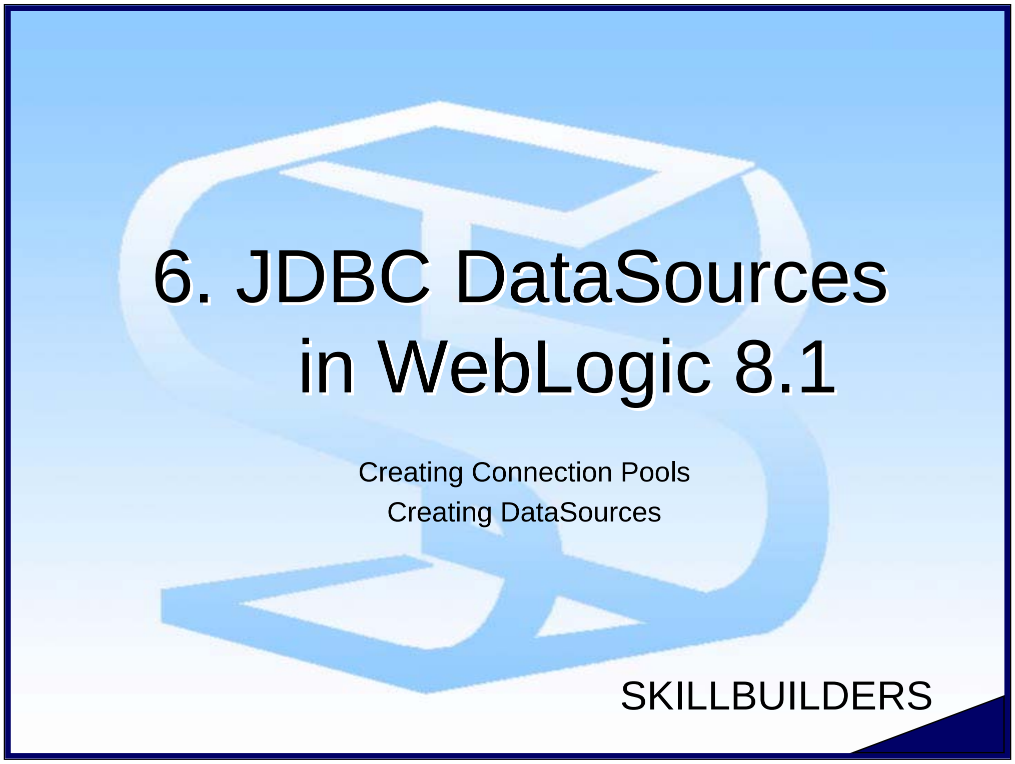
Here are the relevant definitions of `weblogic-ejb-jar.dtd` defining the parts of `weblogic-ejb-jar.xml` that pertain to CMP entity beans:

```
<!ELEMENT entity-descriptor (
  pool?, (entity-cache | entity-cache-ref)?, persistence?,entity-
clustering?,
  invalidation-target?,enable-dynamic-queries?)>
<!ELEMENT persistence (
    is-modified-method-name?,delay-updates-until-end-of-tx?,finders-load-
bean?,
    persistence-use?)>
<!ELEMENT persistence-use (type-identifier, type-version, type-storage)>
```



Where We've Been

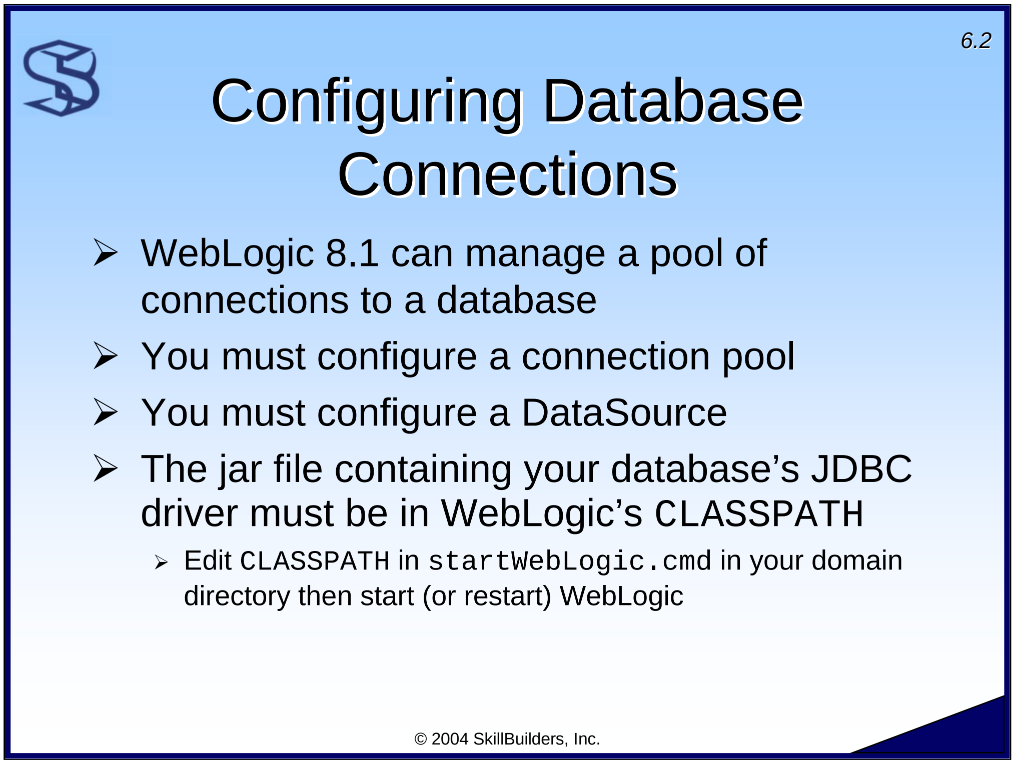
- WebLogic provides tools for persisting a bean
- WebLogic defines connection pools and data sources
- CMP requires several XML files:
  - `weblogic-cmp-rdbms-jar.xml` to map EJB fields to table columns
  - `weblogic-ejb-jar.xml` to map persistence type to specific deployment



6. JDBC DataSources in WebLogic 8.1

Creating Connection Pools
Creating DataSources

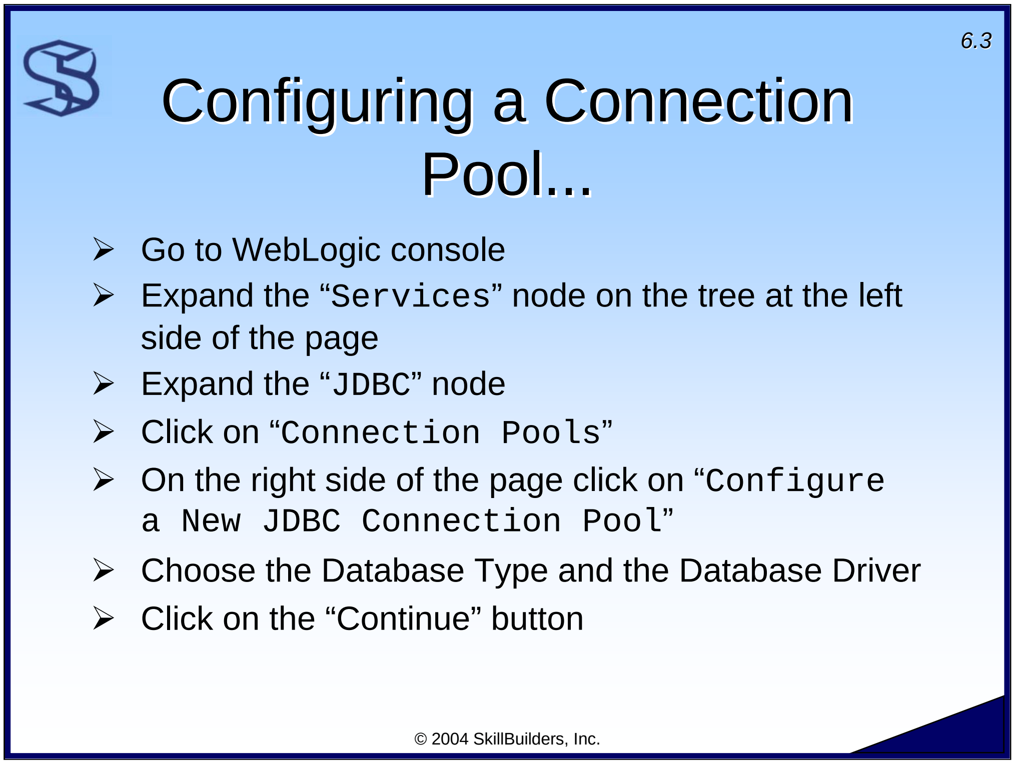
SKILLBUILDERS



6.2

Configuring Database Connections

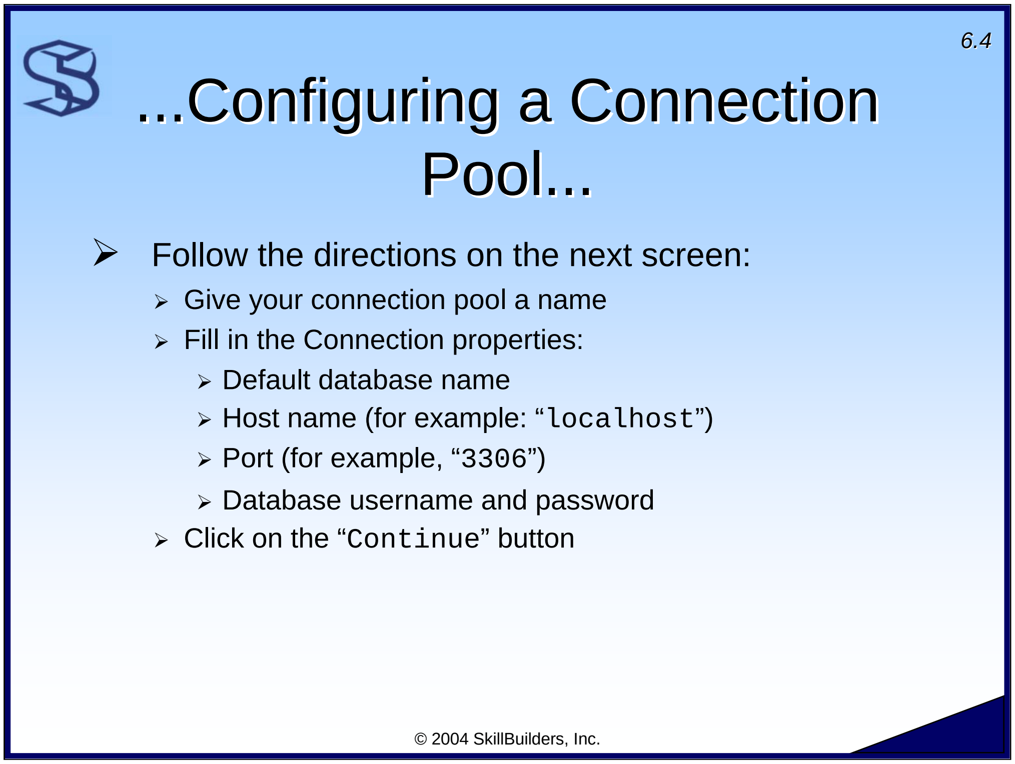
- WebLogic 8.1 can manage a pool of connections to a database
- You must configure a connection pool
- You must configure a DataSource
- The jar file containing your database's JDBC driver must be in WebLogic's `CLASSPATH`
  - Edit `CLASSPATH` in `startWebLogic.cmd` in your domain directory then start (or restart) WebLogic



6.3

Configuring a Connection Pool...

- Go to WebLogic console
- Expand the “`Services`” node on the tree at the left side of the page
- Expand the “`JDBC`” node
- Click on “`Connection Pools`”
- On the right side of the page click on “`Configure a New JDBC Connection Pool`”
- Choose the Database Type and the Database Driver
- Click on the “Continue” button

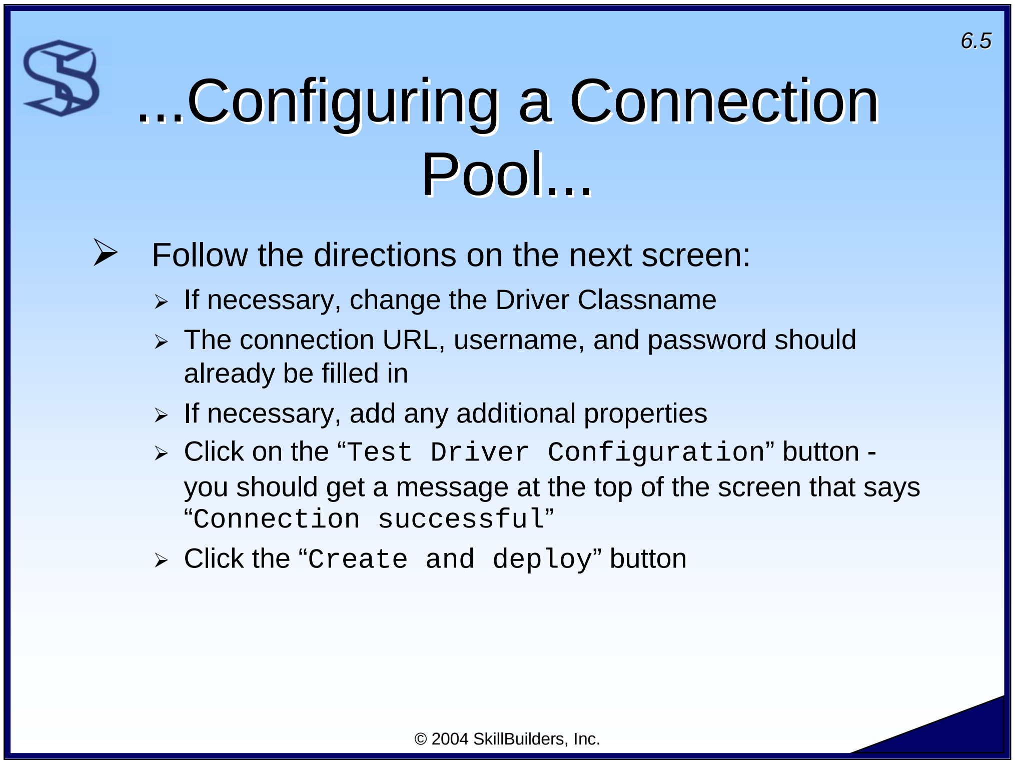


6.4

...Configuring a Connection Pool...

- Follow the directions on the next screen:
  - Give your connection pool a name
  - Fill in the Connection properties:
    - Default database name
    - Host name (for example: “`localhost`”)
    - Port (for example, “`3306`”)
    - Database username and password
  - Click on the “`Continue`” button

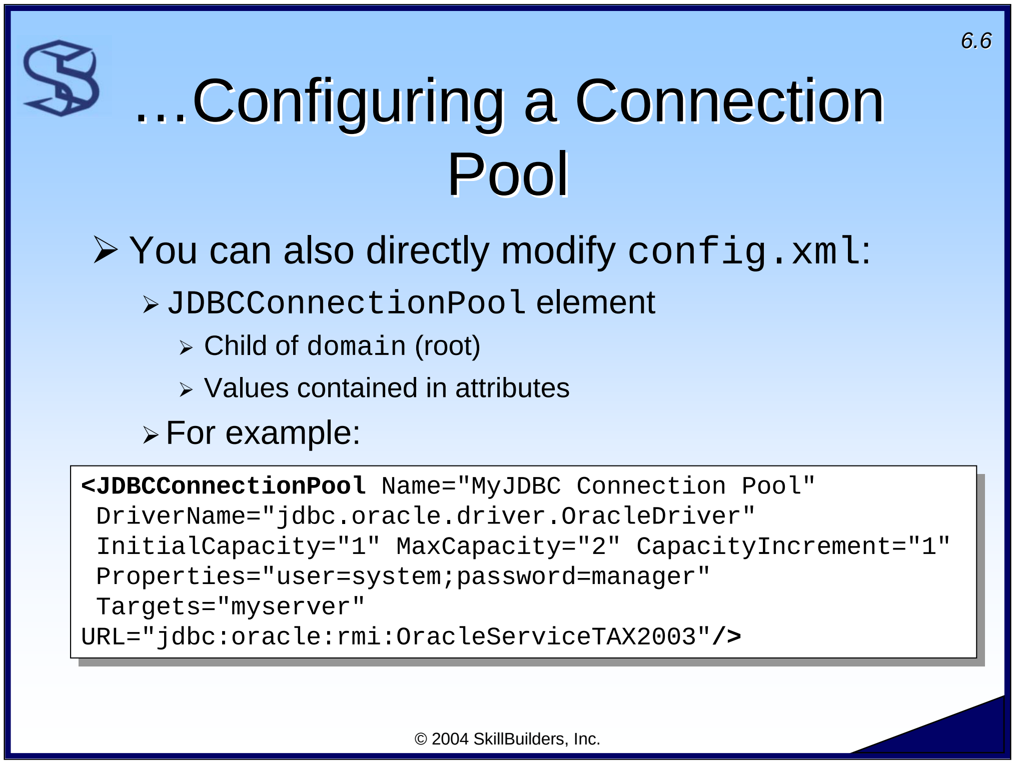
Some of this information can be obtained from documentation provided by the JDBC driver vendor. Some information you may have to obtain from your database administrator.



6.5

...Configuring a Connection Pool...

- Follow the directions on the next screen:
  - If necessary, change the Driver Classname
  - The connection URL, username, and password should already be filled in
  - If necessary, add any additional properties
  - Click on the “`Test Driver Configuration`” button - you should get a message at the top of the screen that says “`Connection successful`”
  - Click the “`Create and deploy`” button



6.6

…Configuring a Connection Pool

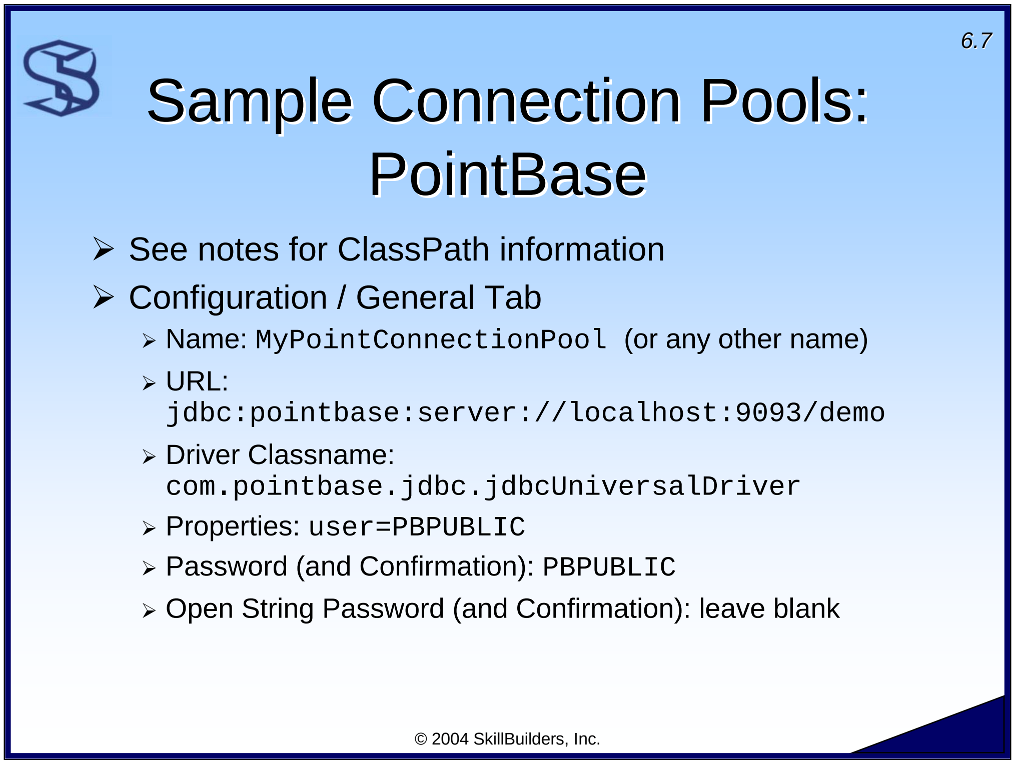
- You can also directly modify `config.xml`:
  - `JDBCConnectionPool` element
    - Child of domain (root)
    - Values contained in attributes
  - For example:

```
<JDBCConnectionPool Name="MyJDBC Connection Pool"
 DriverName="jdbc.oracle.driver.OracleDriver"
 InitialCapacity="1" MaxCapacity="2" CapacityIncrement="1"
 Properties="user=system;password=manager"
 Targets="myserver"
URL="jdbc:oracle:rmi:OracleServiceTAX2003"/>
```

```
<JDBCConnectionPool CapacityIncrement="1"
        DriverName="COM.cloudscape.core.RmiJdbcDriver"
        InitialCapacity="1" MaxCapacity="2" Name="LabPool"
        Properties="user=none;password=none;server=none"
        Targets="myserver" URL="jdbc:cloudscape:rmi:CloudscapeDB"/>
```

It is much easier, and you are less likely to make errors, if you use the WebLogic console to add a Connection Pool rather than editing `config.xml` directly.

Errors in `config.xml` can prevent WebLogic from starting.



6.7

Sample Connection Pools: PointBase

- See notes for ClassPath information
- Configuration / General Tab
  - Name: `MyPointConnectionPool` (or any other name)
  - URL: `jdbc:pointbase:server://localhost:9093/demo`
  - Driver Classname: `com.pointbase.jdbc.jdbcUniversalDriver`
  - Properties: `user=PBPUBLIC`
  - Password (and Confirmation): `PBPUBLIC`
  - Open String Password (and Confirmation): leave blank

© 2004 SkillBuilders, Inc.

Here are steps to create a **PointBase** ConnectionPool

- WebLogic 8.1 comes with PointBase drivers already in the classpath
- If for some reason you need to add them, they are located at:

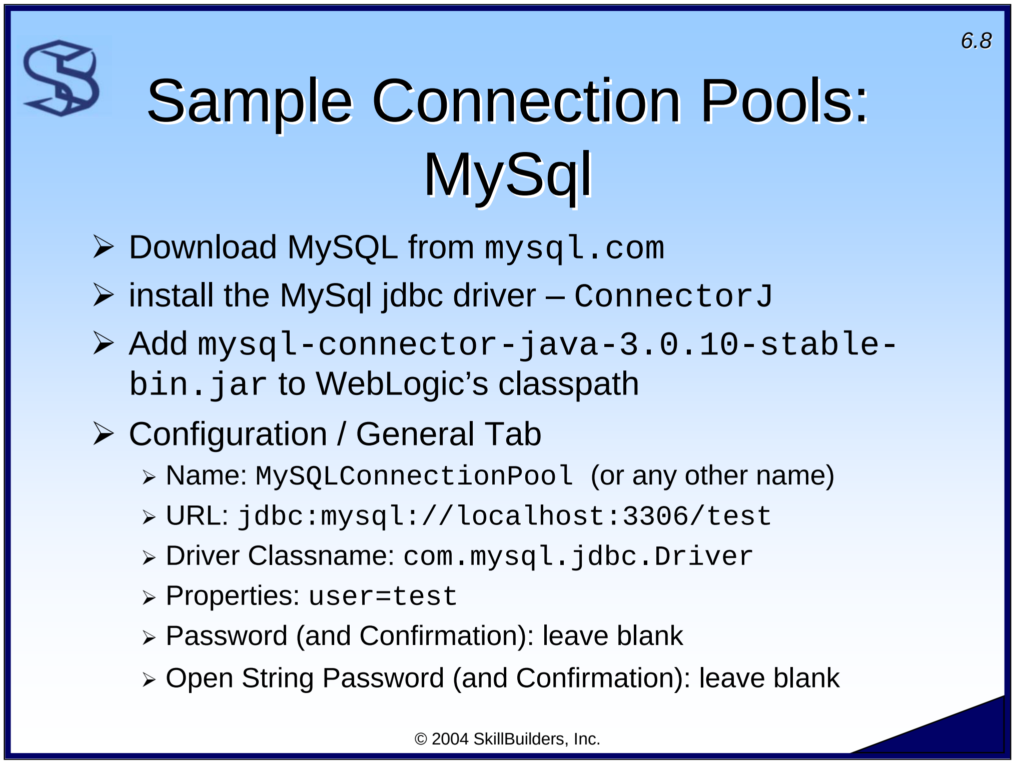
```
c:\bea\weblogic81\common\eval\pointbase\pbserver44.jar
```

and

```
c:\bea\weblogic81\common\eval\pointbase\pbclient44.jar
```

Configuration / General Tab

- Name: `MyPointConnectionPool` (or any other name)
- URL: `jdbc:pointbase:server://localhost:9093/demo`
- Driver Classname: `com.pointbase.jdbc.jdbcUniversalDriver`
- Properties: `user=PBPUBLIC`
- Password (and Confirmation): `PBPUBLIC`
- Open String Password (and Confirmation): leave blank



6.8

Sample Connection Pools: MySql

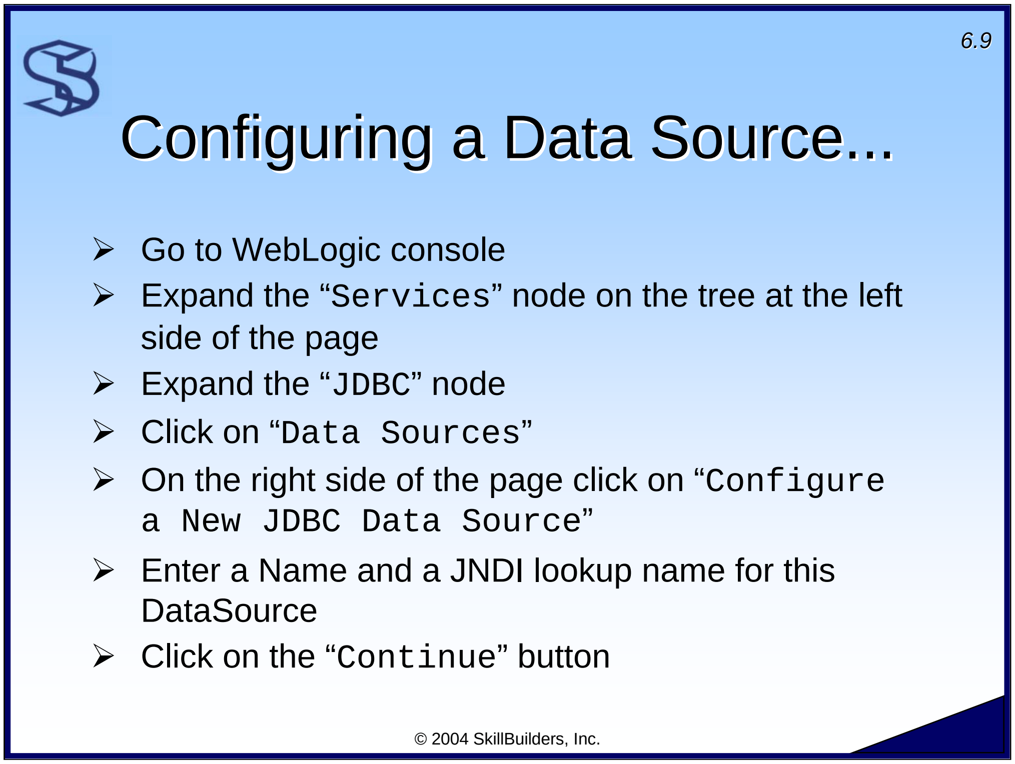
- Download MySQL from `mysql.com`
- install the MySql jdbc driver – `ConnectorJ`
- Add `mysql-connector-java-3.0.10-stable-bin.jar` to WebLogic's classpath
- Configuration / General Tab
  - Name: `MySQLConnectionPool` (or any other name)
  - URL: `jdbc:mysql://localhost:3306/test`
  - Driver Classname: `com.mysql.jdbc.Driver`
  - Properties: `user=test`
  - Password (and Confirmation): leave blank
  - Open String Password (and Confirmation): leave blank

Here are steps to create a **MySQL** ConnectionPool:

- If you do not already have MySql installed, you will need to download it from `mysql.com.` See the MySql documentation for information about creating userids and passwords.
- You will need to install the MySql jdbc driver, `ConnectorJ`. Download it from `mysql.com.`
  - Be sure to compare checksum of downloaded files with value given on site. If there is a discrepancy, try a mirror site.
- Add the file `mysql-connector-java-3.0.10-stable-bin.jar` to WebLogic's classpath. One way to do this is to edit the file `startWebLogic.cmd` located in your domain directory, and add it to the "`set CLASSPATH=`…" statement.

Configuration / General Tab

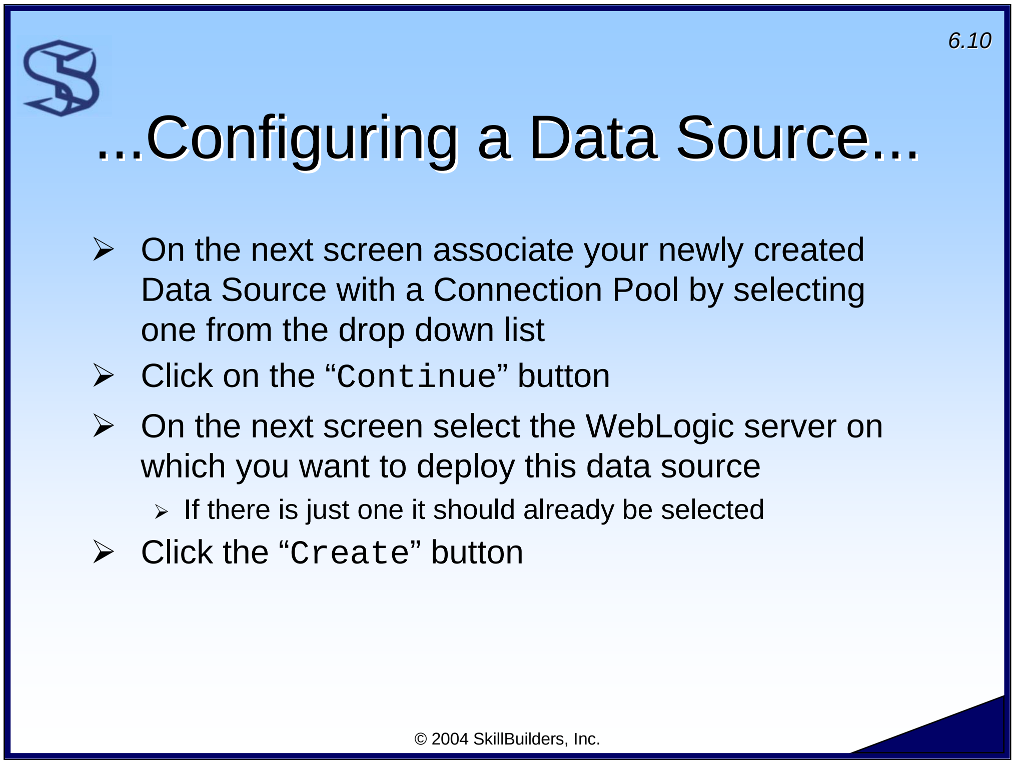
- Name: `MySQLConnectionPool` (or any other name)
- URL: `jdbc:mysql://localhost:3306/test`
- Driver Classname: `com.mysql.jdbc.Driver`
- Properties: `user=test`
- Password (and Confirmation): leave blank
- Open String Password (and Confirmation): leave blank



6.9

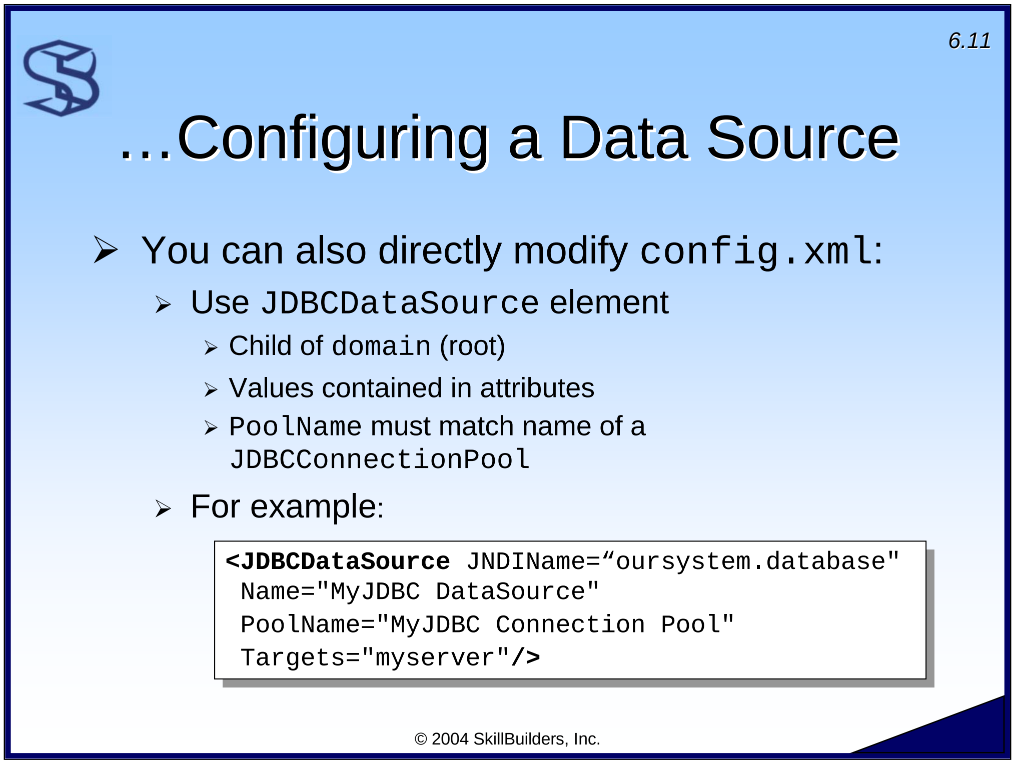
Configuring a Data Source...

- Go to WebLogic console
- Expand the “`Services`” node on the tree at the left side of the page
- Expand the “`JDBC`” node
- Click on “`Data Sources`”
- On the right side of the page click on “`Configure a New JDBC Data Source`”
- Enter a Name and a JNDI lookup name for this DataSource
- Click on the “`Continue`” button



...Configuring a Data Source...

- On the next screen associate your newly created Data Source with a Connection Pool by selecting one from the drop down list
- Click on the “`Continue`” button
- On the next screen select the WebLogic server on which you want to deploy this data source
  - If there is just one it should already be selected
- Click the “`Create`” button



6.11

…Configuring a Data Source

- You can also directly modify `config.xml`:
  - Use `JDBCDataSource` element
    - Child of `domain` (root)
    - Values contained in attributes
    - `PoolName` must match name of a `JDBCConnectionPool`
  - For example:

```
<JDBCDataSource JNDIName=“oursystem.database"
 Name="MyJDBC DataSource"
 PoolName="MyJDBC Connection Pool"
 Targets="myserver"/>
```

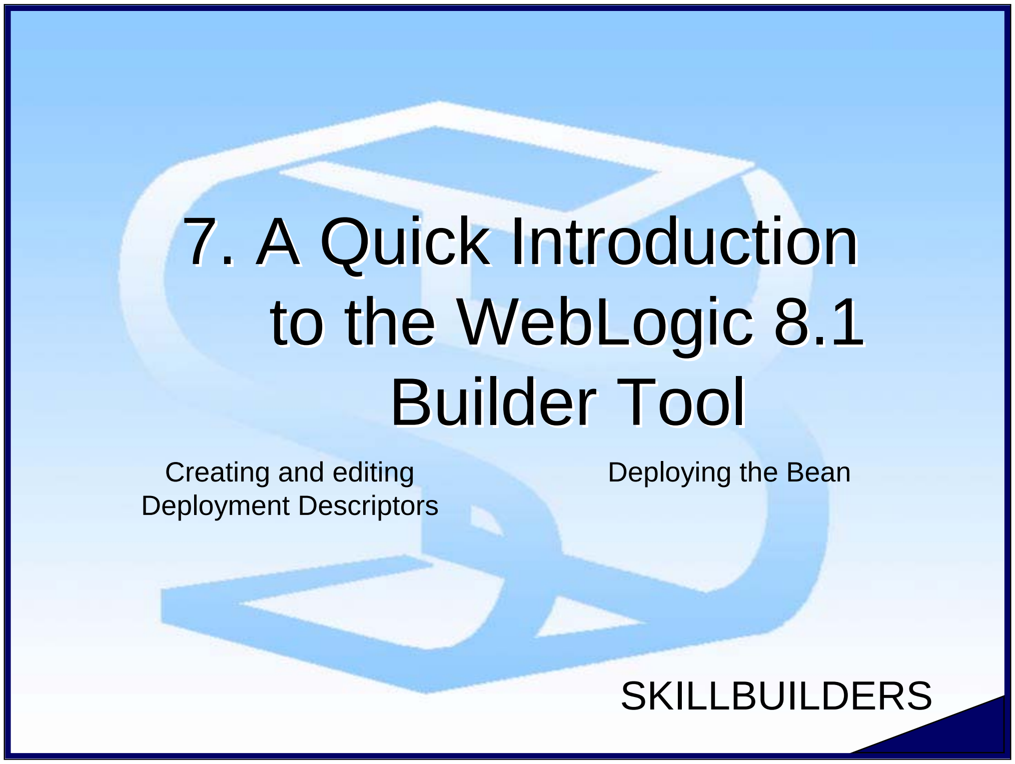
It is much easier and you are less likely to make errors if you use the WebLogic console to add a Data Source rather than editing `config.xml` directly.

Errors in `config.xml` can prevent WebLogic from starting.

6.12

Where We've Been

- WebLogic provides Connection Pools and Data Sources
- Use the console to create these objects

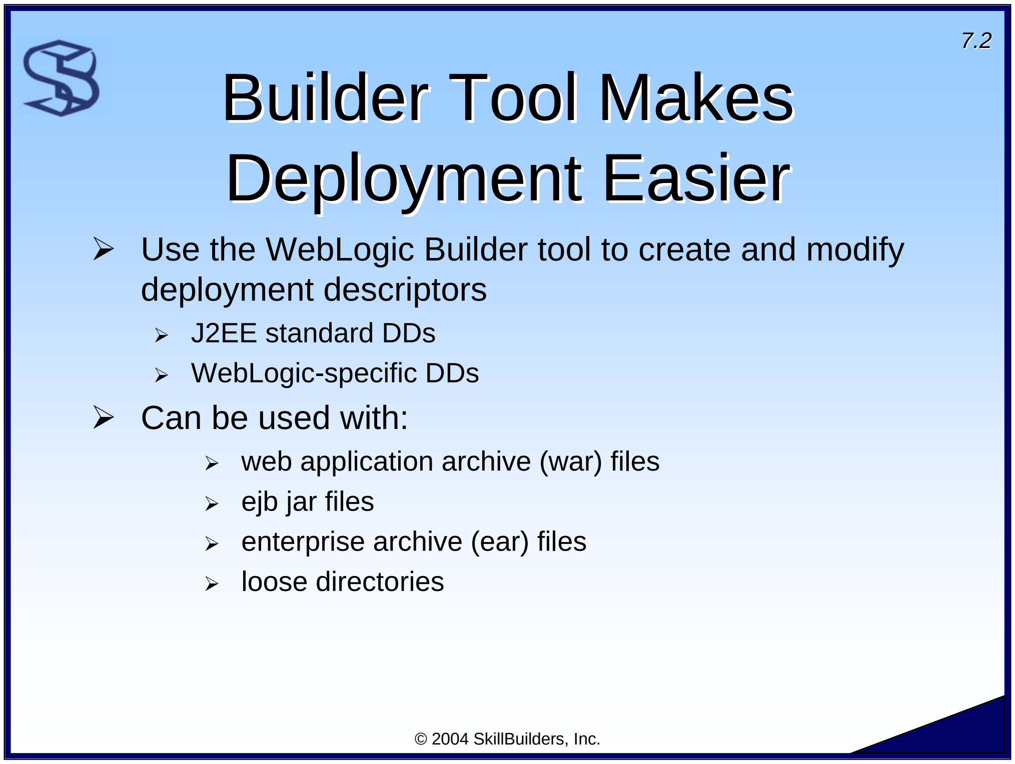


7. A Quick Introduction to the WebLogic 8.1 Builder Tool

Creating and editing Deployment Descriptors

Deploying the Bean

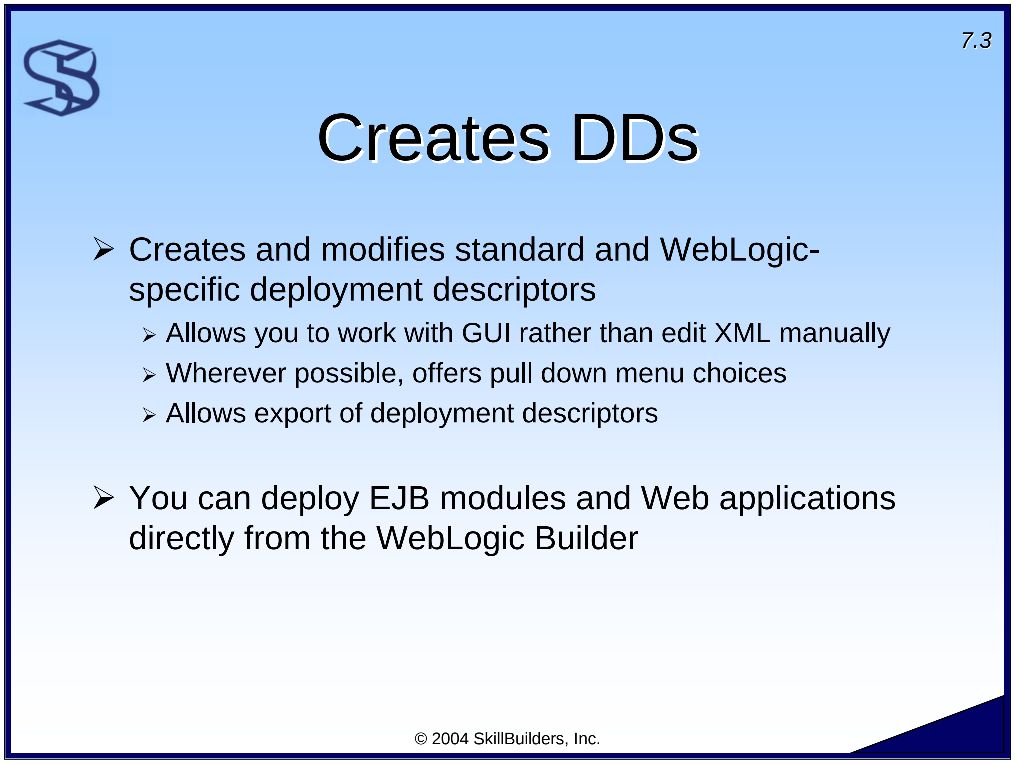
SKILLBUILDERS



7.2

Builder Tool Makes Deployment Easier

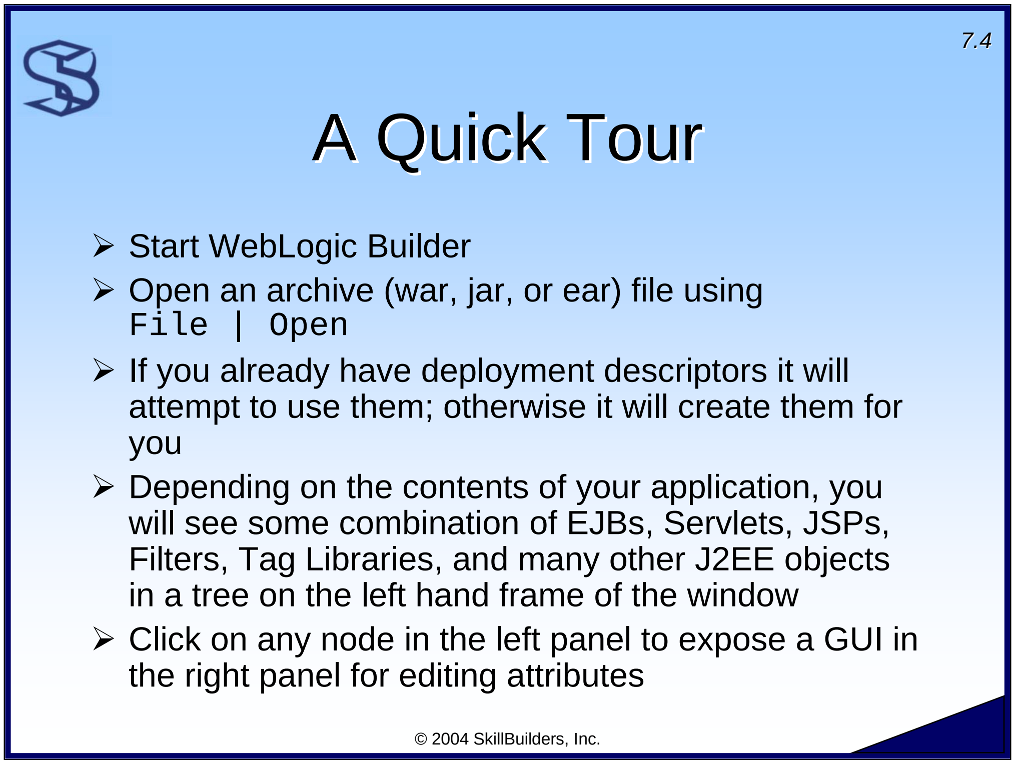
- Use the WebLogic Builder tool to create and modify deployment descriptors
  - J2EE standard DDs
  - WebLogic-specific DDs
- Can be used with:
  - web application archive (war) files
  - ejb jar files
  - enterprise archive (ear) files
  - loose directories



7.3

Creates DDs

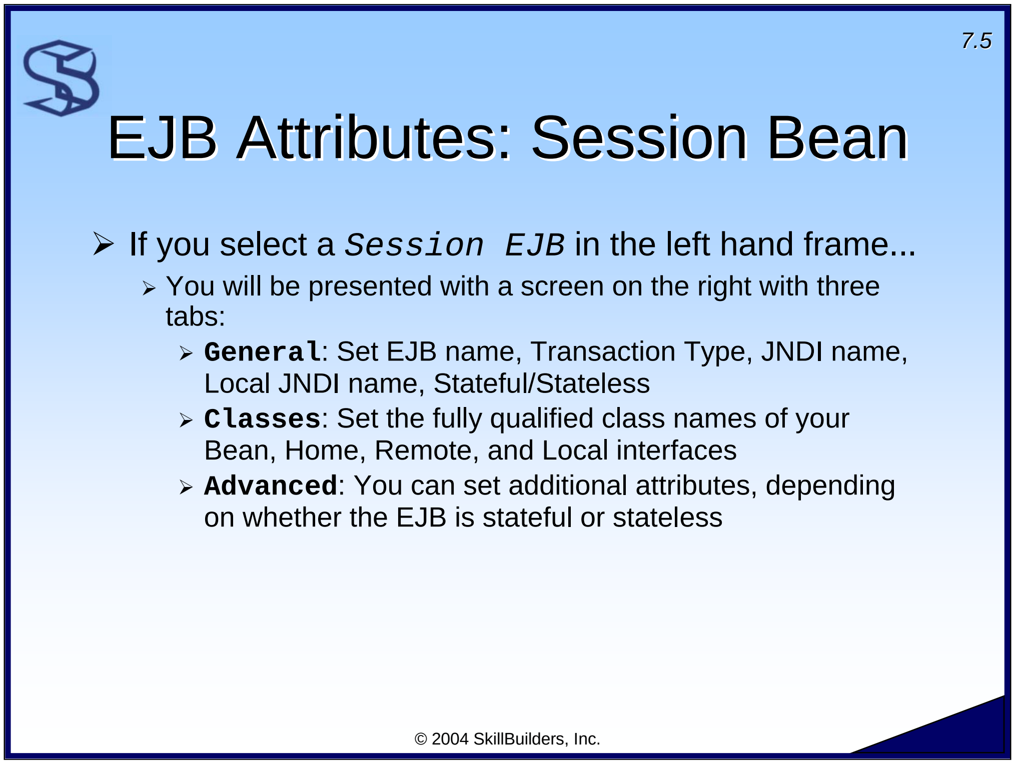
- Creates and modifies standard and WebLogic-specific deployment descriptors
  - Allows you to work with GUI rather than edit XML manually
  - Wherever possible, offers pull down menu choices
  - Allows export of deployment descriptors
- You can deploy EJB modules and Web applications directly from the WebLogic Builder



7.4

A Quick Tour

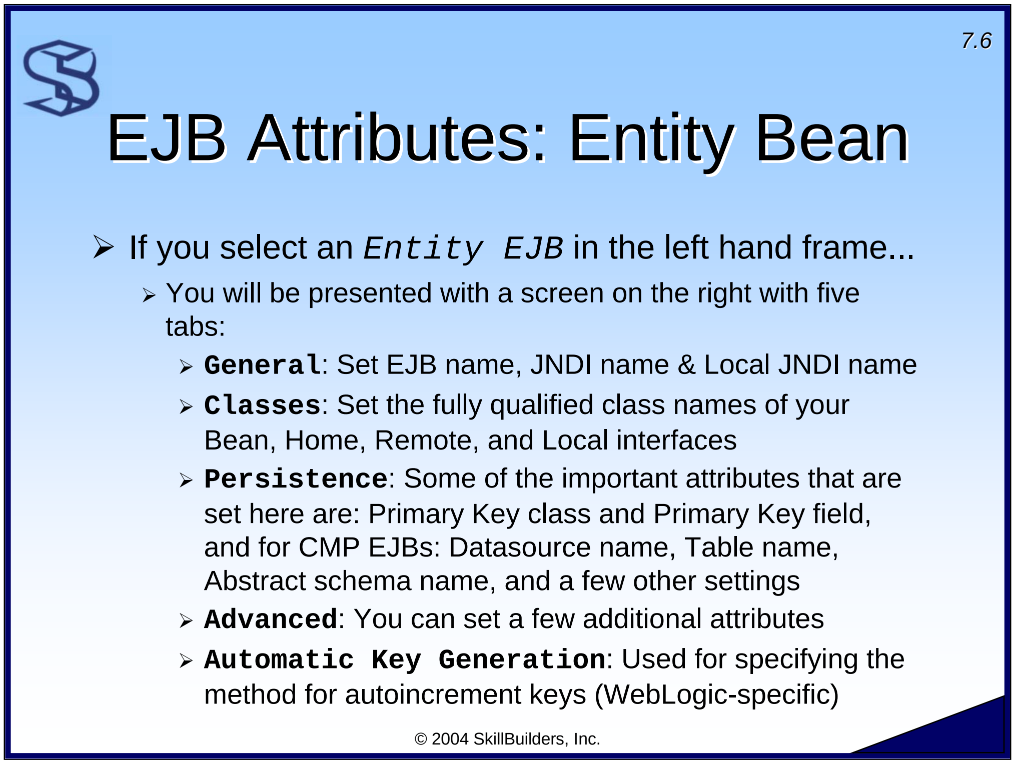
- Start WebLogic Builder
- Open an archive (war, jar, or ear) file using `File | Open`
- If you already have deployment descriptors it will attempt to use them; otherwise it will create them for you
- Depending on the contents of your application, you will see some combination of EJBs, Servlets, JSPs, Filters, Tag Libraries, and many other J2EE objects in a tree on the left hand frame of the window
- Click on any node in the left panel to expose a GUI in the right panel for editing attributes



7.5

EJB Attributes: Session Bean

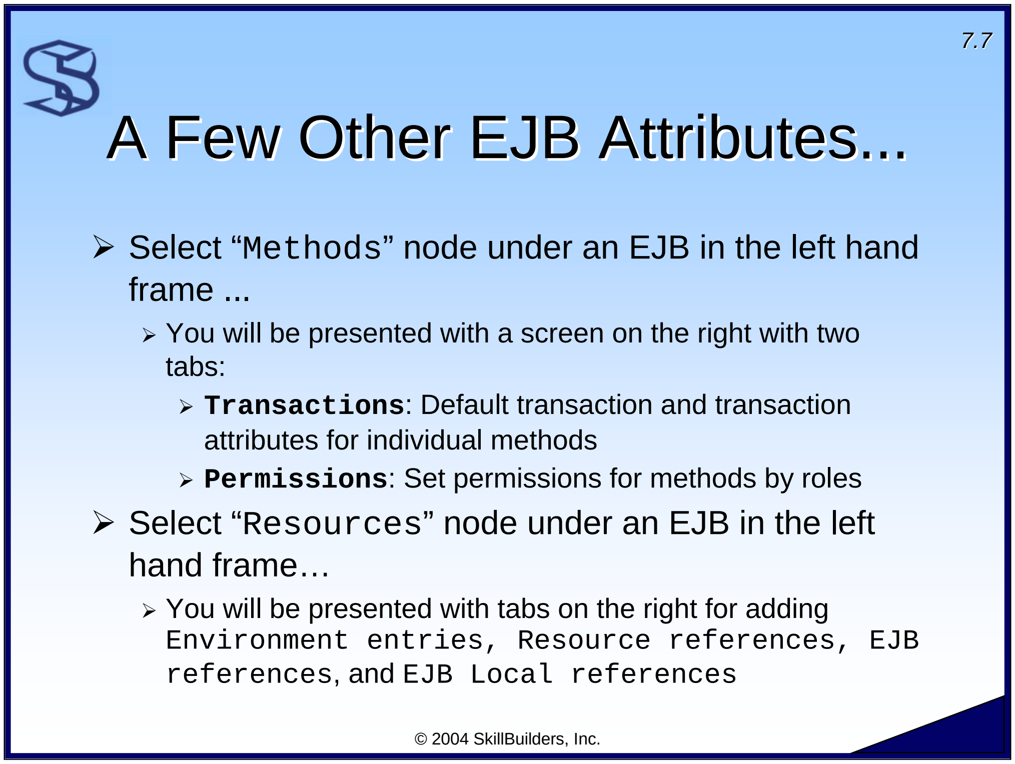
- If you select a `Session EJB` in the left hand frame...
  - You will be presented with a screen on the right with three tabs:
    - **`General`**: Set EJB name, Transaction Type, JNDI name, Local JNDI name, Stateful/Stateless
    - **`Classes`**: Set the fully qualified class names of your Bean, Home, Remote, and Local interfaces
    - **`Advanced`**: You can set additional attributes, depending on whether the EJB is stateful or stateless



7.6

EJB Attributes: Entity Bean

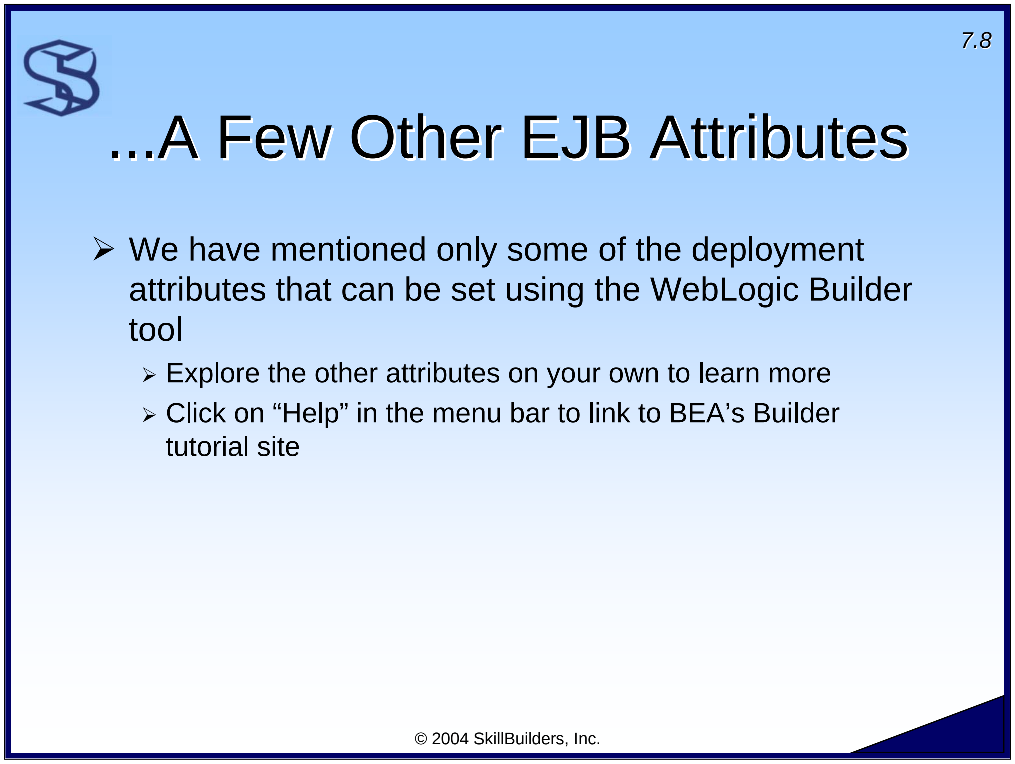
- If you select an *`Entity EJB`* in the left hand frame...
  - You will be presented with a screen on the right with five tabs:
    - **`General`**: Set EJB name, JNDI name & Local JNDI name
    - **`Classes`**: Set the fully qualified class names of your Bean, Home, Remote, and Local interfaces
    - **`Persistence`**: Some of the important attributes that are set here are: Primary Key class and Primary Key field, and for CMP EJBs: Datasource name, Table name, Abstract schema name, and a few other settings
    - **`Advanced`**: You can set a few additional attributes
    - **`Automatic Key Generation`**: Used for specifying the method for autoincrement keys (WebLogic-specific)



7.7

A Few Other EJB Attributes...

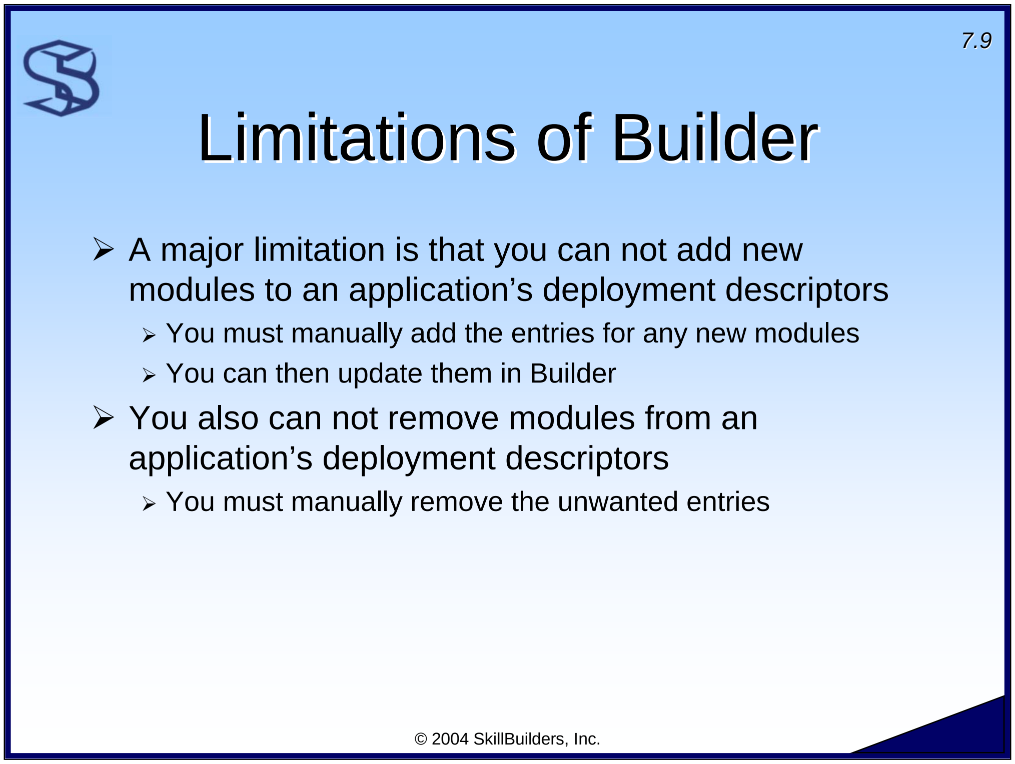
- Select “`Methods`” node under an EJB in the left hand frame ...
  - You will be presented with a screen on the right with two tabs:
    - **`Transactions`**: Default transaction and transaction attributes for individual methods
    - **`Permissions`**: Set permissions for methods by roles
- Select “`Resources`” node under an EJB in the left hand frame…
  - You will be presented with tabs on the right for adding `Environment entries, Resource references, EJB references`, and `EJB Local references`



7.8

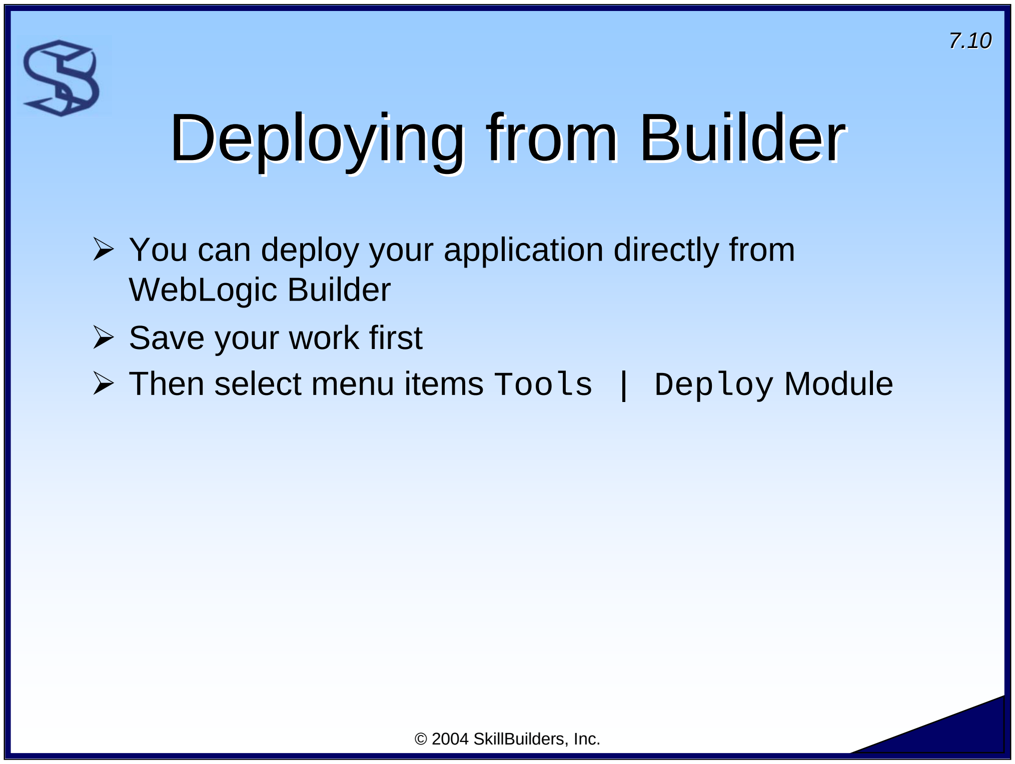
...A Few Other EJB Attributes

- We have mentioned only some of the deployment attributes that can be set using the WebLogic Builder tool
  - Explore the other attributes on your own to learn more
  - Click on “Help” in the menu bar to link to BEA’s Builder tutorial site



Limitations of Builder

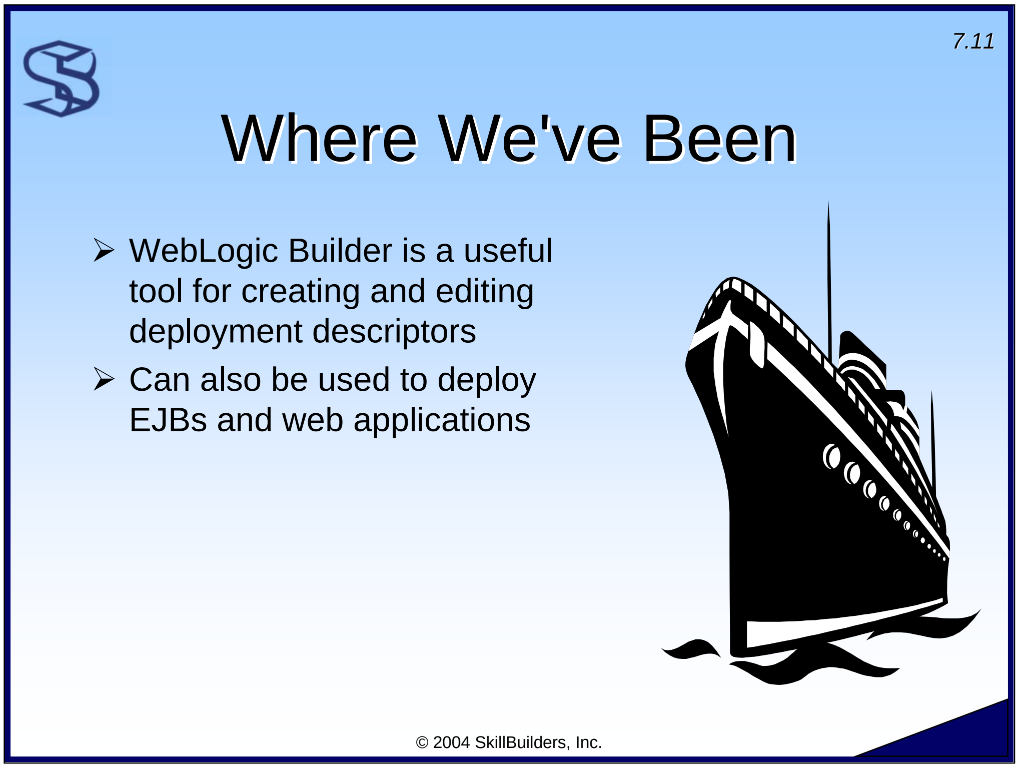
- A major limitation is that you can not add new modules to an application’s deployment descriptors
  - You must manually add the entries for any new modules
  - You can then update them in Builder
- You also can not remove modules from an application’s deployment descriptors
  - You must manually remove the unwanted entries



7.10

Deploying from Builder

- You can deploy your application directly from WebLogic Builder
- Save your work first
- Then select menu items `Tools | Deploy` Module



7.11

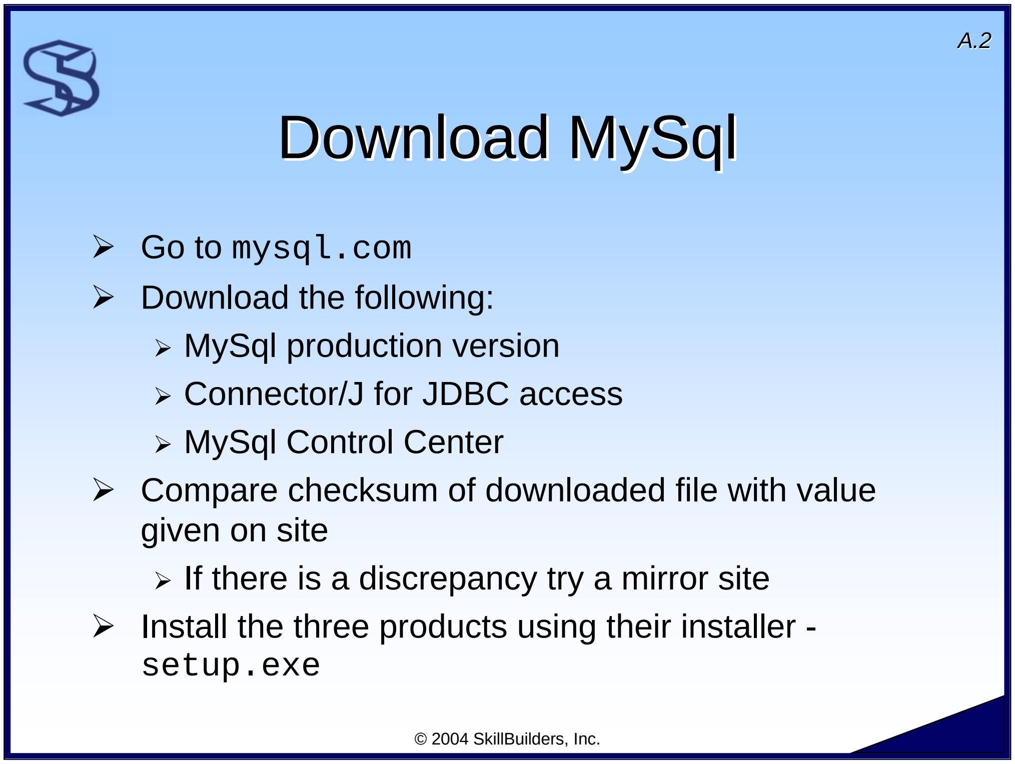
Where We've Been

- WebLogic Builder is a useful tool for creating and editing deployment descriptors
- Can also be used to deploy EJBs and web applications

A. Using MySql with WebLogic 8.1

Installing under Windows

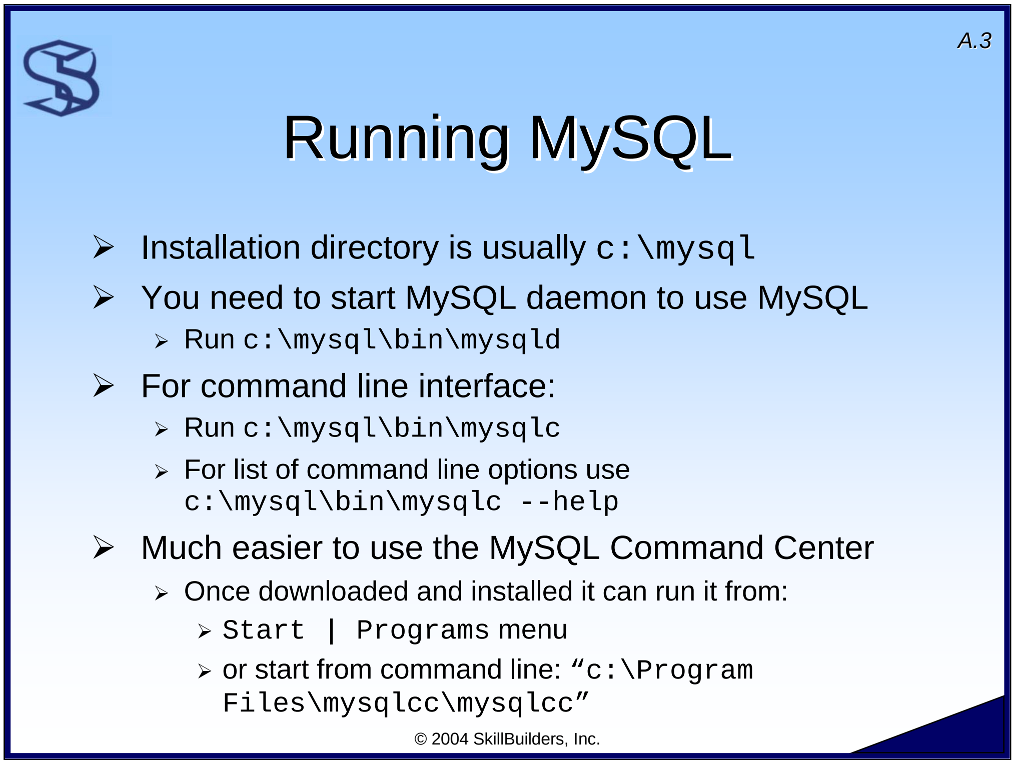
SKILLBUILDERS



A.2

Download MySql

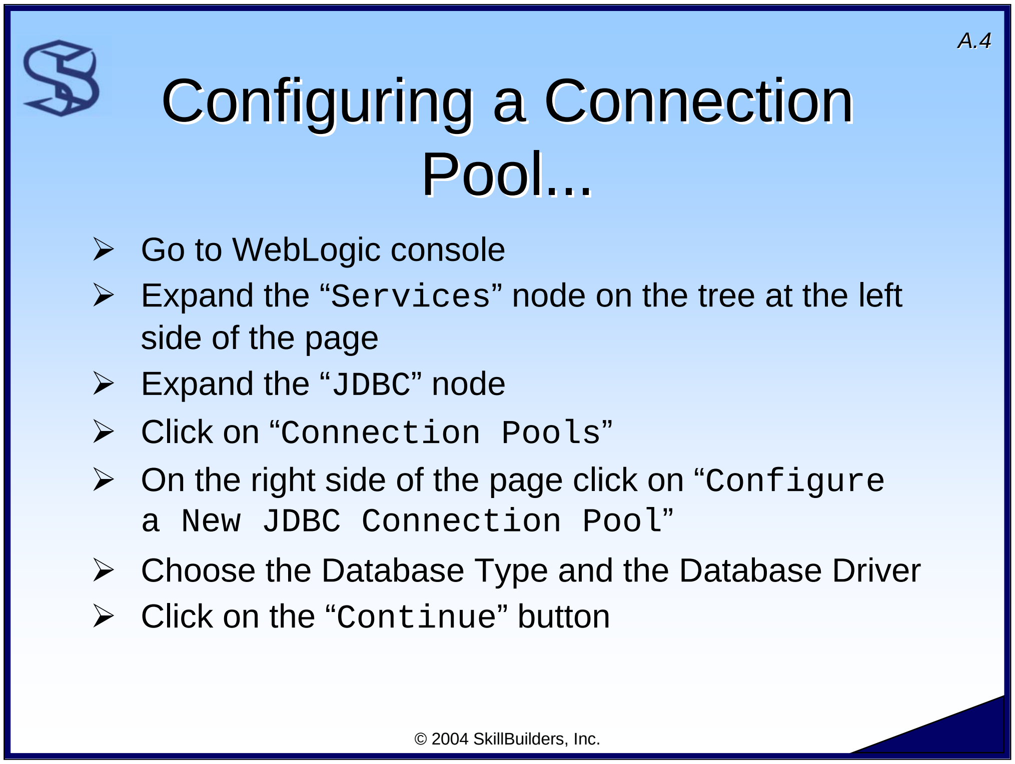
- Go to `mysql.com`
- Download the following:
  - MySql production version
  - Connector/J for JDBC access
  - MySql Control Center
- Compare checksum of downloaded file with value given on site
  - If there is a discrepancy try a mirror site
- Install the three products using their installer - `setup.exe`



A.3

Running MySQL

- Installation directory is usually `c:\mysql`
- You need to start MySQL daemon to use MySQL
  - Run `c:\mysql\bin\mysqld`
- For command line interface:
  - Run `c:\mysql\bin\mysqlc`
  - For list of command line options use `c:\mysql\bin\mysqlc --help`
- Much easier to use the MySQL Command Center
  - Once downloaded and installed it can run it from:
    - `Start | Programs` menu
    - or start from command line: “`c:\Program Files\mysqlcc\mysqlcc`”



A.4

Configuring a Connection Pool...

- Go to WebLogic console
- Expand the “`Services`” node on the tree at the left side of the page
- Expand the “`JDBC`” node
- Click on “`Connection Pools`”
- On the right side of the page click on “`Configure a New JDBC Connection Pool`”
- Choose the Database Type and the Database Driver
- Click on the “`Continue`” button

© 2004 SkillBuilders, Inc.

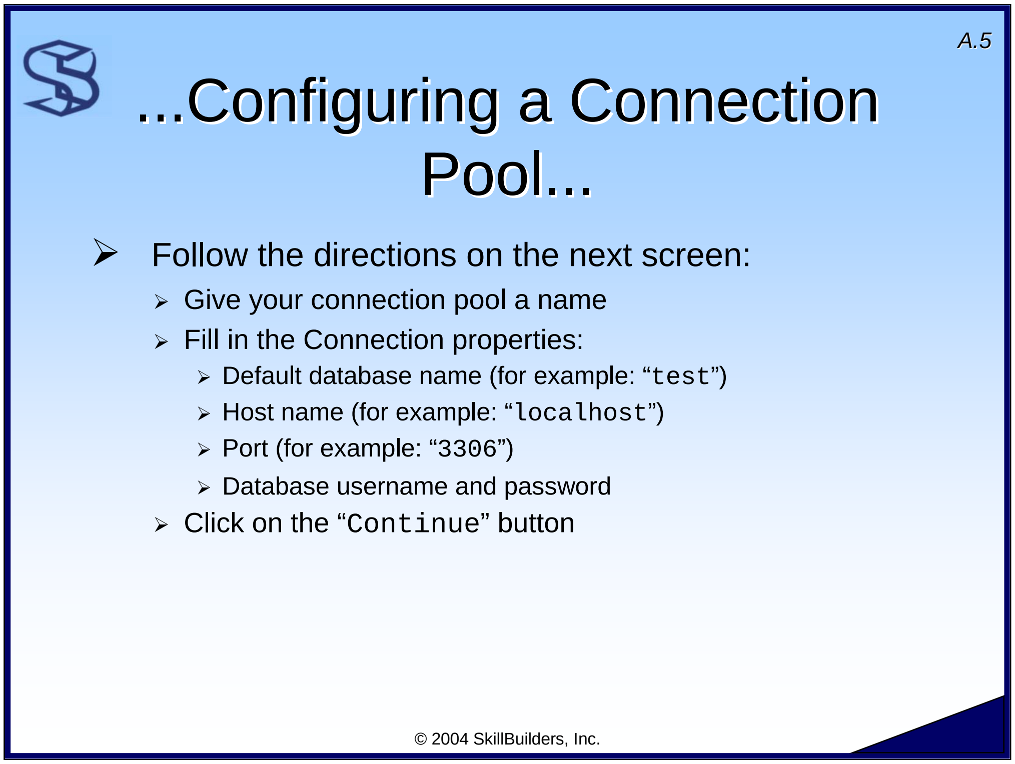
Make sure that the Connector/J jar file that contains the MySql JDBC Driver is in WebLogic’s CLASSPATH for your domain. The latest one (as of this writing) will be:

```
mysql-connector-java-3.0.10-stable-bin.jar
```

located in the following directory:

```
c:\mysql-connector-java-3.0.10-stable
```

You can edit `startWebLogic.cmd` in your WebLogic domain directory, and add it there. Then, before attempting to create a connection pool, shut down and restart WebLogic.

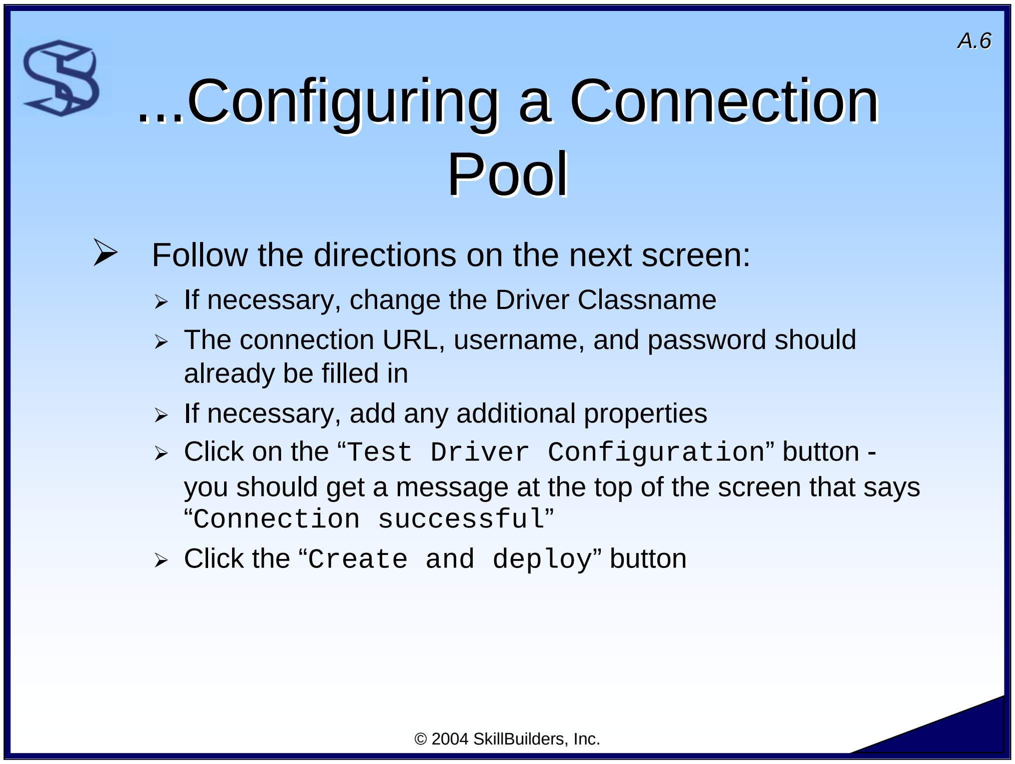


A.5

...Configuring a Connection Pool...

- Follow the directions on the next screen:
  - Give your connection pool a name
  - Fill in the Connection properties:
    - Default database name (for example: "`test`")
    - Host name (for example: "`localhost`")
    - Port (for example: "`3306`")
    - Database username and password
  - Click on the "`Continue`" button

© 2004 SkillBuilders, Inc.

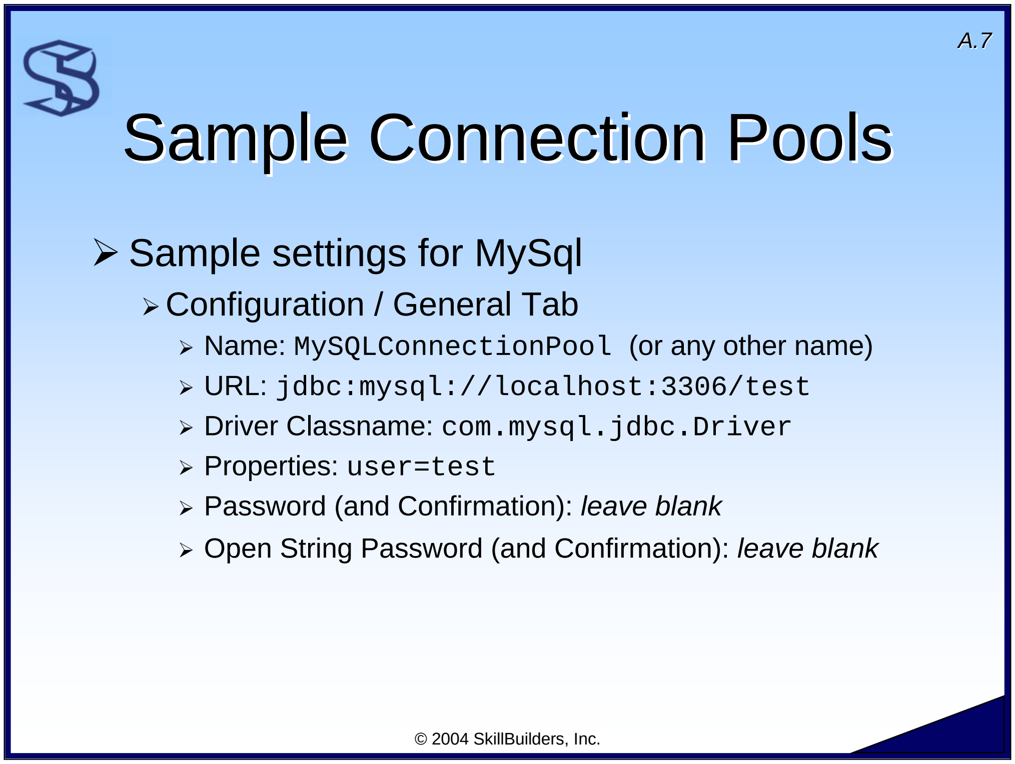


A.6

...Configuring a Connection Pool

- Follow the directions on the next screen:
  - If necessary, change the Driver Classname
  - The connection URL, username, and password should already be filled in
  - If necessary, add any additional properties
  - Click on the "`Test Driver Configuration`" button - you should get a message at the top of the screen that says "`Connection successful`"
  - Click the "`Create and deploy`" button

© 2004 SkillBuilders, Inc.



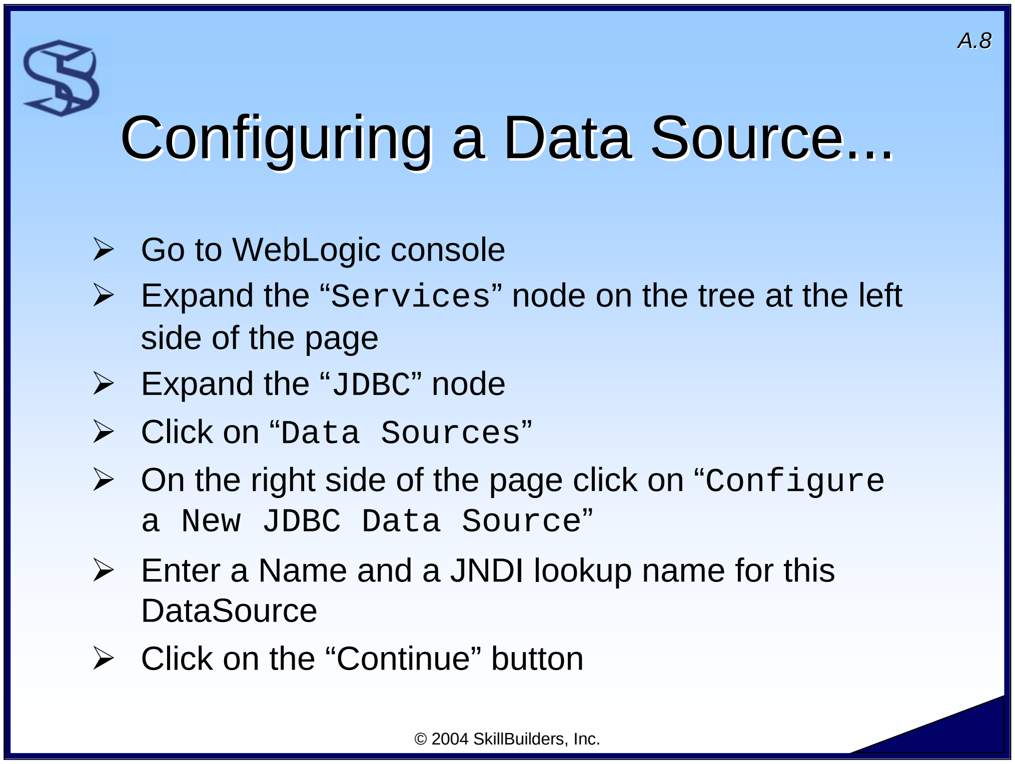
A.7

Sample Connection Pools

- Sample settings for MySql
  - Configuration / General Tab
    - Name: `MySQLConnectionPool` (or any other name)
    - URL: `jdbc:mysql://localhost:3306/test`
    - Driver Classname: `com.mysql.jdbc.Driver`
    - Properties: `user=test`
    - Password (and Confirmation): *leave blank*
    - Open String Password (and Confirmation): *leave blank*

Configuration / General Tab

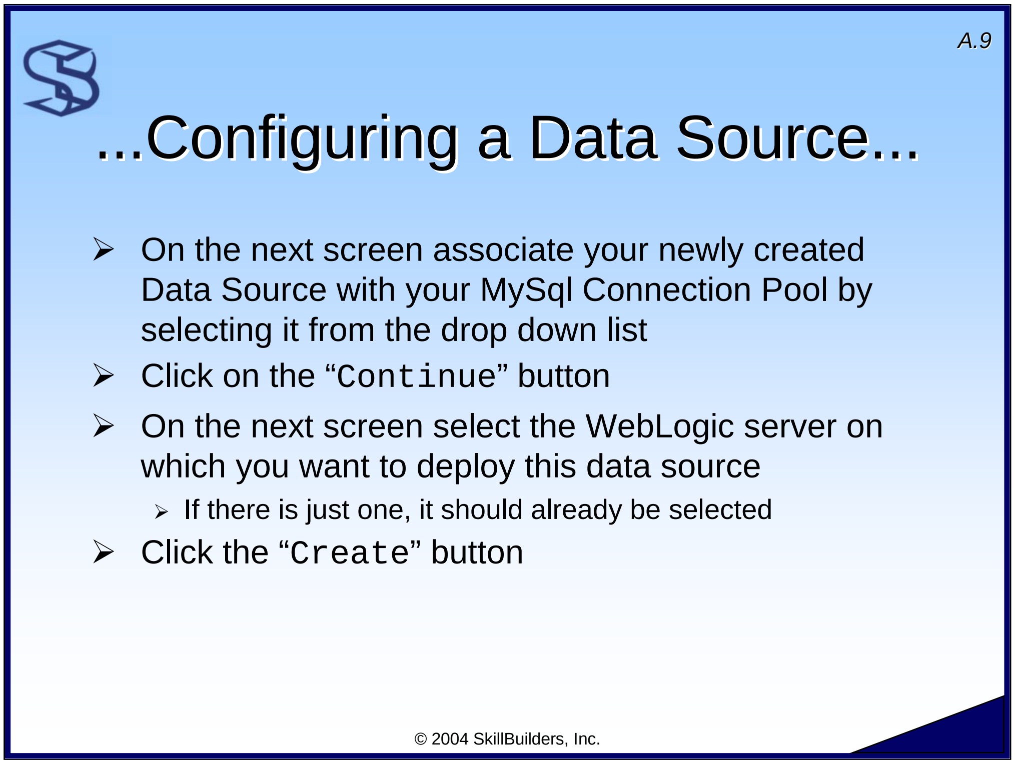
- Name: `MySQLConnectionPool` (or any other name)
- URL: `jdbc:mysql://localhost:3306/test`
- Driver Classname: `com.mysql.jdbc.Driver`
- Properties: `user=test`
- Password (and Confirmation): *leave blank*
- Open String Password (and Confirmation): *leave blank*



A.8

Configuring a Data Source...

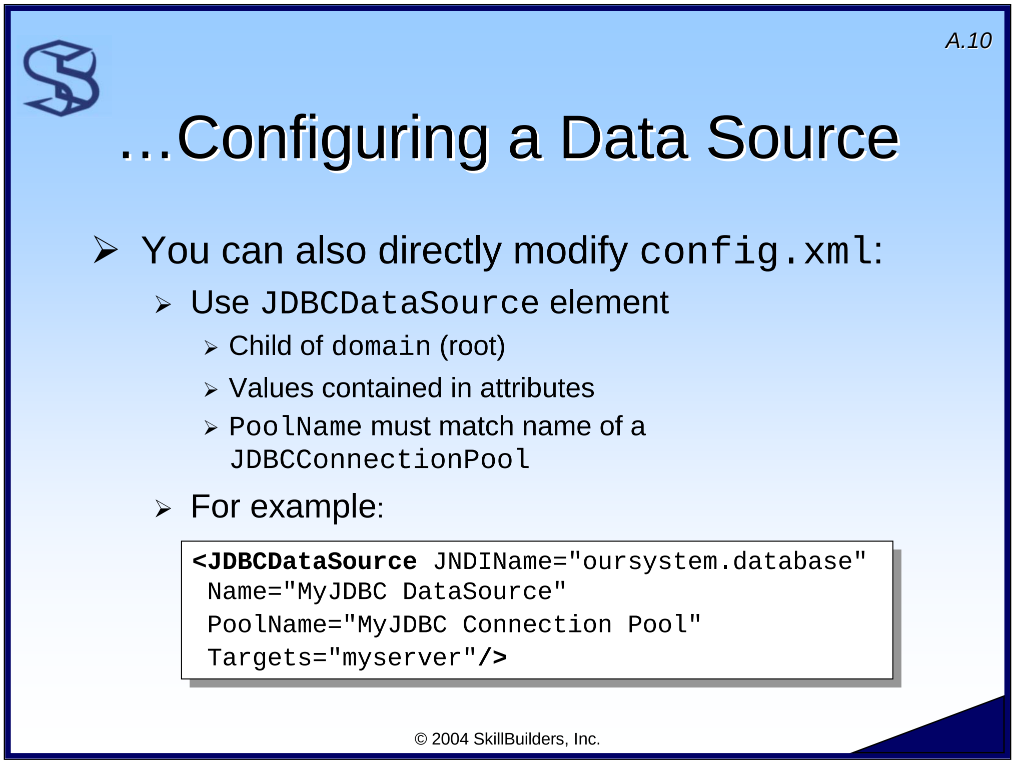
- Go to WebLogic console
- Expand the “`Services`” node on the tree at the left side of the page
- Expand the “`JDBC`” node
- Click on “`Data Sources`”
- On the right side of the page click on “`Configure a New JDBC Data Source`”
- Enter a Name and a JNDI lookup name for this DataSource
- Click on the “Continue” button



A.9

...Configuring a Data Source...

- On the next screen associate your newly created Data Source with your MySql Connection Pool by selecting it from the drop down list
- Click on the “`Continue`” button
- On the next screen select the WebLogic server on which you want to deploy this data source
  - If there is just one, it should already be selected
- Click the “`Create`” button



A.10

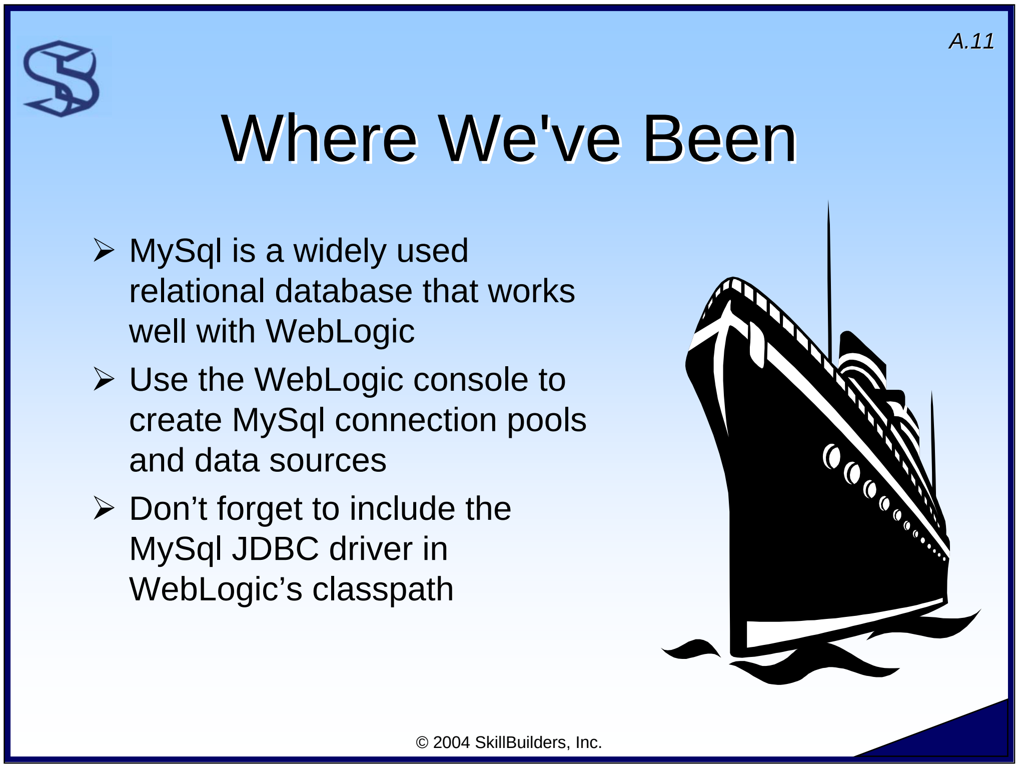
…Configuring a Data Source

- You can also directly modify `config.xml`:
  - Use `JDBCDataSource` element
    - Child of `domain` (root)
    - Values contained in attributes
    - `PoolName` must match name of a `JDBCConnectionPool`
  - For example:

```
<JDBCDataSource JNDIName="oursystem.database"
 Name="MyJDBC DataSource"
 PoolName="MyJDBC Connection Pool"
 Targets="myserver"/>
```

It is much easier, and you are less likely to make errors, if you use the WebLogic console to add a Data Source rather than editing `config.xml` directly.

Errors in `config.xml` can prevent WebLogic from starting.



A.11

Where We've Been

- ➢ MySql is a widely used relational database that works well with WebLogic
- ➢ Use the WebLogic console to create MySql connection pools and data sources
- ➢ Don’t forget to include the MySql JDBC driver in WebLogic’s classpath

© 2004 SkillBuilders, Inc.