Community Experience Distilled

Java EE Development with Eclipse

Develop Java EE applications with Eclipse and commonly used technologies and frameworks

Deepak Vohra

[PACKT] PUBLISHING open source* community experience distilled

Java EE Development with Eclipse

Develop Java EE applications with Eclipse and commonly used technologies and frameworks

Deepak Vohra

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

Java EE Development with Eclipse

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: December 2012

Production Reference: 1131212

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78216-096-0

www.packtpub.com

Cover Image by Asher Wishkerman (wishkerman@hotmail.com)

Credits

Author
Deepak Vohra

Reviewers
Filippo Bosi
Frank Nimphius
Federico Tomassetti
Phil Wilkins

Acquisition Editors
Mary Nadar
Dhwani Devater

Lead Technical Editors
Sweny M. Sukumaran
Neeshma Ramakrishnan

Technical Editors
Sharvari Baet
Devdutt Kulkarni

Project Coordinator
Amey Sawant

Proofreader
Maria Gould

Indexer
Tejal R. Soni

Production Coordinator
Conidon Miranda

Cover Work
Conidon Miranda

About the Author

Deepak Vohra is a consultant and a principal member of the `NuBean.com` software company. Deepak is a Sun Certified Java Programmer and Web Component Developer, and has worked in the fields of XML and Java programming and J2EE for over five years. Deepak is the co-author of the Apress book, *Pro XML Development with Java Technology,* and was the technical reviewer for the O'Reilly book, *WebLogic: The Definitive Guide*. Deepak was also the technical reviewer for the Course Technology PTR book, *Ruby Programming for the Absolute Beginner*, and the technical editor for the Manning Publications book, *Prototype and Scriptaculous in Action*. Deepak is also the author of the Packt Publishing books, *JDBC 4.0 and Oracle JDeveloper for J2EE Development*, *Processing XML documents with Oracle JDeveloper 11g*, *EJB 3.0 Database Persistence with Oracle Fusion Middleware 11g*, and *Java 7 JAX-WS Web Services*.

About the Reviewers

Filippo Bosi (Twitter `@filippobosi`) is currently employed at Imola Informatica (`www.imolinfo.it`), an Italian consulting company where he works as Senior Advisor, managing important projects for banking and insurance companies.

He has been working for more than 25 years in the computer programming field. He started his career as a freelance consultant and writer for some Italian computer magazines, while at the same time offering freelance consultancies in the first years that banking and insurance companies were starting to move away from mainframes in order to implement their business.

In the last four years, he's been involved in redesigning from scratch the entire information system of a banking institution in an SOA fashion, in studying ways to apply Semantic Web technologies to address Enterprise Architecture and Knowledge Management problems for some Italian large banking and insurance companies, and an European project (Cloud4SOA – `www.cloud4soa.eu`) that attempts, through the use of semantics, to address the portability of applications and data between different PaaS providers.

He is currently interested in Agile and Lean Management (applied), design of SOA Architectures, Enterprise Architecture, Cloud Computing, and Semantic Web.

He can be contacted at `fbosi@imolinfo.it`.

Frank Nimphius is a Senior Principal Product Manager in the Oracle Application Development Tools group at Oracle Corporation, specializing in Oracle JDeveloper and the Oracle Application Development Framework (ADF).

In his current position, Frank represents and evangelizes the Oracle JDeveloper and Oracle ADF product worldwide as a speaker at user group and technology conferences, as well as in various publications. Frank runs the ADF Code Corner website, the *OTN Forum Harvest* blog, and is the co-author of the *Oracle Fusion Developer Guide* book published in 2009 by McGraw Hill.

Federico Tomassetti is a software engineer and a PhD student in computer engineering. He is interested mainly in model-driven development and domain specific languages. He has experience as a technical writer, teacher, and consultant about these technologies.

He is studying at the Politecnico di Torino. He spent a semester in the Universität Karlsruhe and one at Fortiss, an Institut of the Technische Universität München.

Phil Wilkins has spent nearly 25 years in the software industry working with both multinationals and software startups. He started out as a developer and has worked his way up through technical and development management roles. The last 12 years have been primarily in Java based environments. He now works as an architect with an enterprise wide technical remit within the IT group for a global optical healthcare manufacturer and retailer.

Outside of his work commitments, he has contributed his technical capabilities to supporting others in a wide range of activities from the development of community websites to providing input and support to people authoring books, and developing software ideas and businesses.

When not immersed in work and technology, he spends his down time pursing his passion for music and time with his wife and two boys.

I'd like to take this opportunity to thank my wife Catherine and our two sons Christopher and Aaron for their tolerance for the innumerable hours that I spent in front of a computer contributing to both my employer and the many other IT related activities that I've supported over the years.

Table of Contents

Preface

Java Platform, Enterprise Edition (Java EE) 6 is the industry standard for enterprise Java computing. Eclipse IDE for Java EE developers is the most commonly used Java IDE for Java EE development. Eclipse IDE for Java EE developers supports Java EE 5 completely and also supports several features from Java EE 6.

The Oracle WebLogic Server product line is the industry's most comprehensive platform for developing, deploying, and integrating enterprise applications. Oracle Enterprise Pack for Eclipse provides a set of plugins (project facets) for Eclipse development with WebLogic Server.

While a number of books are available on Eclipse IDE for Java Developers, none or very few are available on Eclipse IDE for Java EE Developers. In this book, we shall discuss Java EE development in Eclipse IDE for Java EE developers. While it is not feasible to cover all of the more than 30 technologies in the Java EE stack (`http://www.oracle.com/technetwork/java/javaee/tech/index.html`), we shall discuss the most commonly used Java EE technologies, especially the ones Eclipse IDE for Java EE developers (or Oracle Enterprise Pack for Eclipse) provides Project for Facets. Oracle Enterprise Pack for Eclipse is just an enhancement of Eclipse IDE for Java EE developers with integrated support for Oracle WebLogic Server.

The objective of the book is to discuss how a developer would develop Java EE applications using commonly used Java EE technologies and frameworks in Eclipse IDE for Java EE developers. The book covers all aspects of application development including:

- Setting the environment for an application
- Using the Eclipse IDE wizards and the Component Palette
- Running a sample application

What this book covers

Chapter 1, EJB 3.0 Database Persistence discusses creating an EJB project using the EJB 3.0 Module project facet. To create an entity bean, we add the JPA project facet. Subsequently, we generate entity beans from Oracle database tables. We create a session bean facade for the entity beans; wrapping an entity bean in a session bean facade is a best practice. We create a JSP client for the EJB application. We package and deploy the EJB application to Oracle WebLogic Server using an Ant build script and run the test client on the WebLogic Server.

Chapter 2, O/X Mapping with JAXB 2.x discusses the Object/XML (O/X) bi-directional mapping provided by the JAXB framework. We discuss the advantages of JAXB 2.x over JAXB 1.0. We create a JAXB web project using the JAXB project facet. We use the EclipseLink 2.4 persistence provider. We create an XML Schema and generate JAXB classes from the XML Schema using JAXB schema compilation. Subsequently, we marshall an XML document from a Java **Document Object Model** (**DOM**) document object, and also unmarshall an XML document using the compiled Java classes. We map an annotated Java class to an XML document using the annotations API. We also demonstrate the support for mapping Java classes to an XML Schema.

Chapter 3, Developing a Web Project for JasperReports demonstrates the use of the Oracle Enterprise Pack for Eclipse's integrated support for Oracle WebLogic Server to deploy and run any web application that requires an application server. First, we configure an Oracle database data source in WebLogic Server. We create and deploy a web application for JasperReports to the WebLogic Server, and subsequently run the web application to create PDF and Excel reports.

Chapter 4, Creating a JSF Data Table discusses how to use the JavaServer Faces project facet to create a JSF data table. First, we create a web project. Subsequently, we create a managed bean, create a JSF page, add a JSF data table to the JSF page, and run the JSF web application on the integrated WebLogic Server to create a JSF data table.

Chapter 5, Templating with Facelets discusses templating with Facelets. **Templating** is the use of a common "template", which is just an XHTML page, in Facelets' composition pages. Templating makes use of Facelets' header and footer pages for describing the common sections of Facelets' composition pages. WebLogic Server includes a shared library for JSF 2.0, which we configure first. We create a web project for Facelets, and create a managed bean to create a JSF data table. We add the 2.0 version of the JavaServer Faces project facet to the web project. For templating, we add a Facelets Template in which we configure the default sections of a Facelets composition page, a header, a content section, and a footer. We add Facelets composition pages for an SQL query input and a JSF Data Table output. We add the implicit navigation, a new feature in JSF 2.0. We run the Facelets application to demonstrate templating by including the same header and footer images in the input and output pages.

Chapter 6, Creating Apache Trinidad User Interfaces discusses the Trinidad project facet. Trinidad was formerly Oracle ADF Faces and provides a set of user interface components. First, we create a web project and add the Trinidad project facet to it. Subsequently, we create JSPs to create and find a catalog entry in Oracle database. We add Trinidad components to the JSP pages. We run the Trinidad application in the integrated WebLogic Server.

Chapter 7, Creating an AJAX Application discusses how to develop an AJAX application to send an asynchronous request to the server and receive a response from the server. The JavaScript project facet is enabled by default in a web project. The AJAX application is used to create a catalog entry in Oracle database by first validating the catalog ID using AJAX. The application is packaged, deployed, and run on the WebLogic Server.

Chapter 8, Creating a JAX-WS Web Service discusses how to use the Java API for XML web services (JAX-WS) to create a web service. First, we create a web service project, which has the Oracle WebLogic web service project facet associated with it. We test the web service on the server and generate a WSDL, which we test in the web explorer. We create a client class for the web service and package, then deploy and test the web service on the WebLogic Server.

Chapter 9, RESTful Web Services Using the JAX-RS API discusses RESTful web services using **Java API for RESTful web services** (**JAX-RS**), which are specified in the JSR 311 specification. We use the JAX-RS project facet for the RESTful web service. We create a Resource class, which is exposed as a URI path using the `@PATH` annotation. Subsequently, we create a Jersey Client API to test the web service.

Chapter 10, Spring discusses how to create a Spring framework application using the Spring project facet. We discuss method interception with a method interceptor and a Spring client. We also discuss Aspect Oriented Programming (AOP) in combination with JSF. We discuss creating a Spring bean, a bean definition file, and an AOP JavaBean.

What you need for this book

The book is based on Eclipse IDE for Java EE Developers version 3.7. We use the Oracle Enterprise Pack for Eclipse packaged Eclipse IDE with integrated support for Oracle WebLogic Server 12*c*, which may be downloaded from `http://www.oracle.com/technetwork/middleware/ias/downloads/wls-main-097127.html`. We have used the Oracle Database Express Edition 11g Release 2, which can be downloaded from `http://www.oracle.com/technetwork/products/express-edition/overview/index.html`.

Some other chapter specific software such as JasperReports is also required. We have used the Windows version, but if you have Linux installed the book may still be used (though the source code and samples have not been tested with Linux). Slight modifications may be required with the Linux Install; for example, the directory paths on Linux would be different than the Windows directory paths used in the book. You need to install J2SE 5.0 or later.

Who this book is for

The target audience of the book is Java EE application developers who want to learn about the practical use of Eclipse IDE for application development. This book is suitable for professional Java EE developers. The book is also suitable for an intermediate/advanced level course in Java EE development. The target audience is expected to have prior, albeit beginner's, knowledge about Java EE, Enterprise JavaBeans (EJB) 3.0, entity and session EJBs, JavaServer Faces (JSF), ADF Faces, AJAX, web services, and Spring framework. The book also requires some familiarity with WebLogic Server and Eclipse IDE.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "The `catalog.xsd` Schema gets parsed and compiled."

A block of code is set as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.example.org/catalog"
xmlns:catalog="http://www.example.org/catalog"
  elementFormDefault="qualified">
  <xsd:element name="catalog" type="catalog:catalogType" />
  <xsd:element name="catalogid" type="xsd:int" />
  <xsd:complexType name="catalogType"> [default]
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.example.org/catalog"
xmlns:catalog="http://www.example.org/catalog"
  elementFormDefault="qualified">
  <xsd:element name="catalog" type="catalog:catalogType" />
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "clicking the **Next** button moves you to the next screen".

[Warnings or important notes appear in a box like this.]

[Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title through the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website, or added to any list of existing errata, under the Errata section of that title.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

Copyright Credits

Some of the contents of this book were originally published by Oracle Technology Network and `http://home.java.net/`. They are republished with the permission of Oracle Corporation.

1

EJB 3.0 Database Persistence

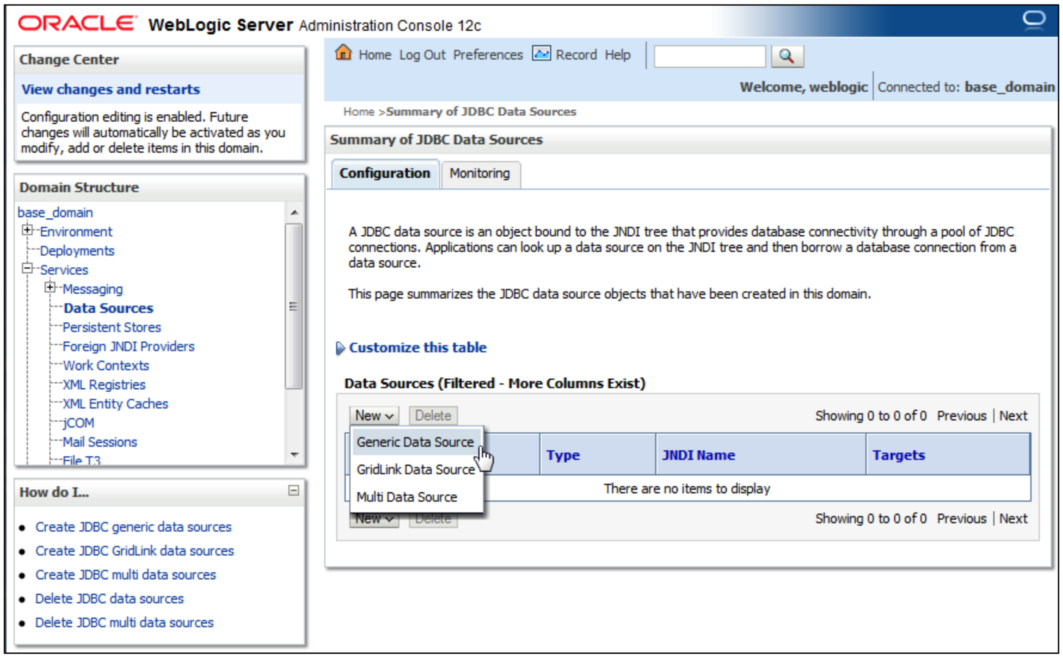
EJB's **entity beans** are the most common technology for database persistence. Developing entity EJBs requires a Java IDE, an application server, and a relational database. Eclipse 3.7 provides wizards for developing entity beans and session facades. In this chapter, we shall develop EJB 3.0 entity beans including session facades. We shall deploy the EJB application to WebLogic Server 12*c* (12.1.1) and test database persistence with the Oracle database 11g XE.

In this chapter, we shall learn the following:

- Configuring a data source in **WebLogic Server** (**WLS**) with the Oracle database
- Creating tables in the Oracle database
- Creating an **Enterprise JavaBeans** (**EJB**) project
- Adding the **Java Persistence API** (**JPA**) project facet
- Generating entity beans from database tables
- Creating a session bean facade
- Creating the `application.xml` file
- Creating a test client
- Packaging and deploying the entity bean application
- Testing the **JavaServer Pages** (JSP) client

Configuring a data source

In this section we shall configure a data source in Oracle WebLogic Server 12*c*. First, download and install the Oracle WebLogic Server from `http://www.oracle.com/technetwork/middleware/ias/downloads/wls-main-097127.html`. Configure the **base_domain** structure in the **WebLogic Server** console. We need to create a data source so that when we deploy and run the application in the server, the application has access to the database. Log in to the **WebLogic Server Administration Console** server for the **base_domain** domain using the URL `http://localhost:7001/Console`. In the **base_domain** domain structure, expand the **Services** tab and select the **Data Sources** node. In the **Data Sources** table, click on **New** and select **Generic Data Source** as shown in the following screenshot:



Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

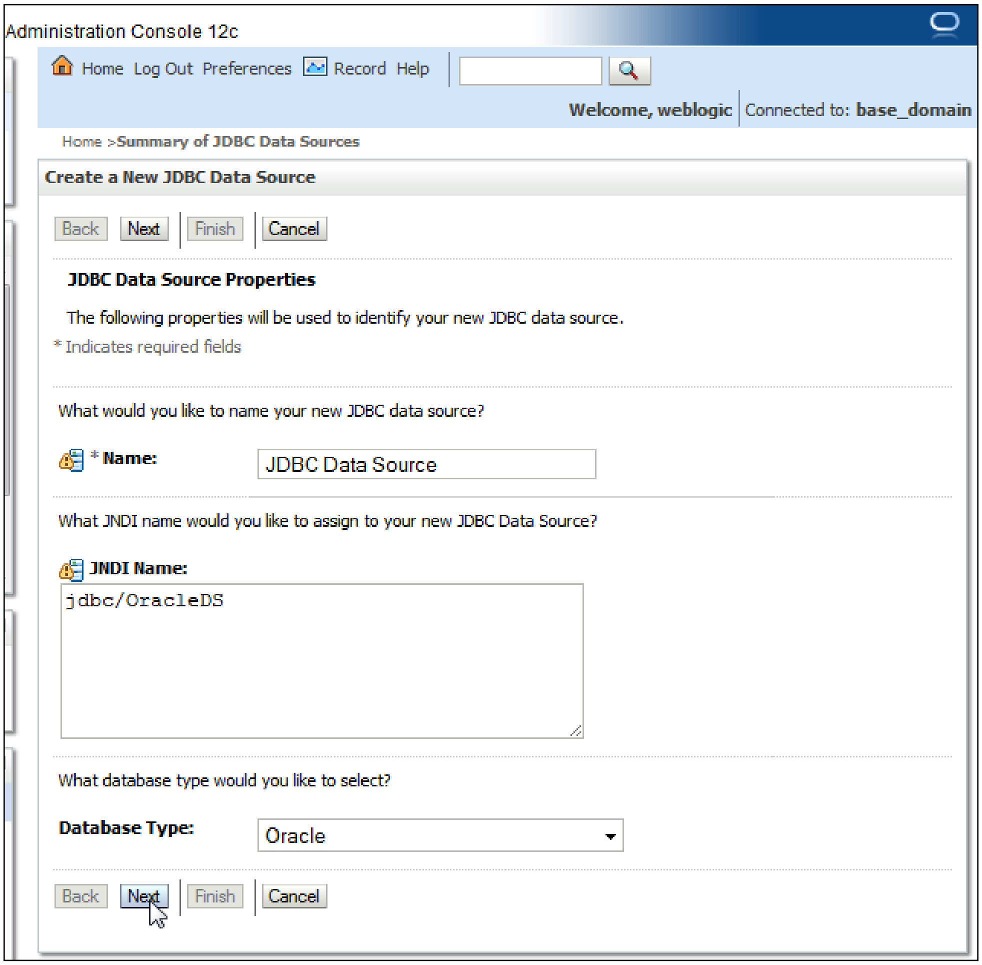
A JDBC data source is an object bound to the JNDI tree that provides database connectivity through a pool of JDBC connections. Applications can look up a data source on the JNDI tree and then borrow a database connection from a data source.

This page summarizes the JDBC data source objects that have been created in this domain.

Data Sources (Filtered - More Columns Exist)

Type	JNDI Name	Targets
There are no items to display		

In **Create a New JDBC Data Source**, specify a data source name and **JNDI Name** (for example, **jdbc/OracleDS**) for the data source. The database shall be accessed using **JNDI Name** lookup in the *Creating a session bean facade* section. Select **Database Type** as **Oracle** and click on **Next** as shown in the following screenshot:


Administration Console 12c
Home
Log Out
Preferences
Record
Help
Welcome, weblogic
Connected to: base_domain
Home >Summary of JDBC Data Sources
Back
Next
Finish
Cancel

Create a New JDBC Data Source

JDBC Data Source Properties

The following properties will be used to identify your new JDBC data source.

* Indicates required fields

What would you like to name your new JDBC data source?

* Name: JDBC Data Source

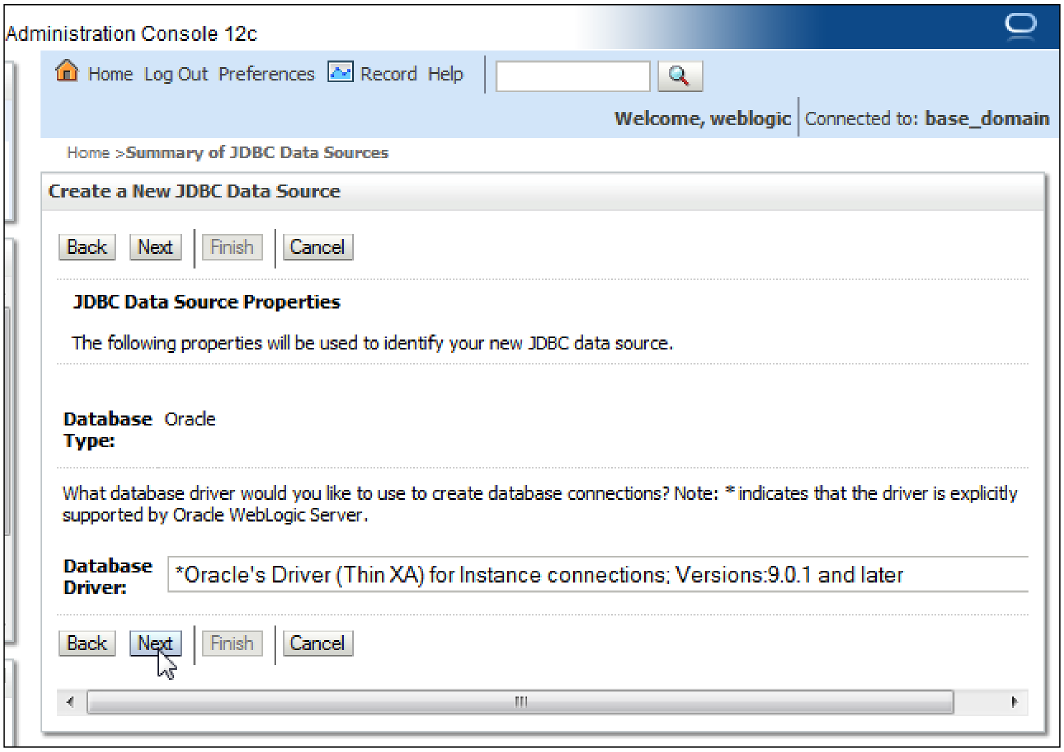
What JNDI name would you like to assign to your new JDBC Data Source?

JNDI Name: jdbc/OracleDS

What database type would you like to select?

Database Type: Oracle

In **JDBC Data Source Properties**, select **Database Driver** as **Oracle's Driver (Thin XA)**. Another JDBC driver may also be selected based on requirements. Refer to the *Selection of the JDBC Driver* document available at `http://docs.oracle.com/cd/E14072_01/java.112/e10590/keyprog.htm#i1005587` for selecting a suitable JDBC driver. Click on **Next** as shown in the following screenshot:


Administration Console 12c
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Home >Summary of JDBC Data Sources
Create a New JDBC Data Source
Back Next Finish Cancel

JDBC Data Source Properties

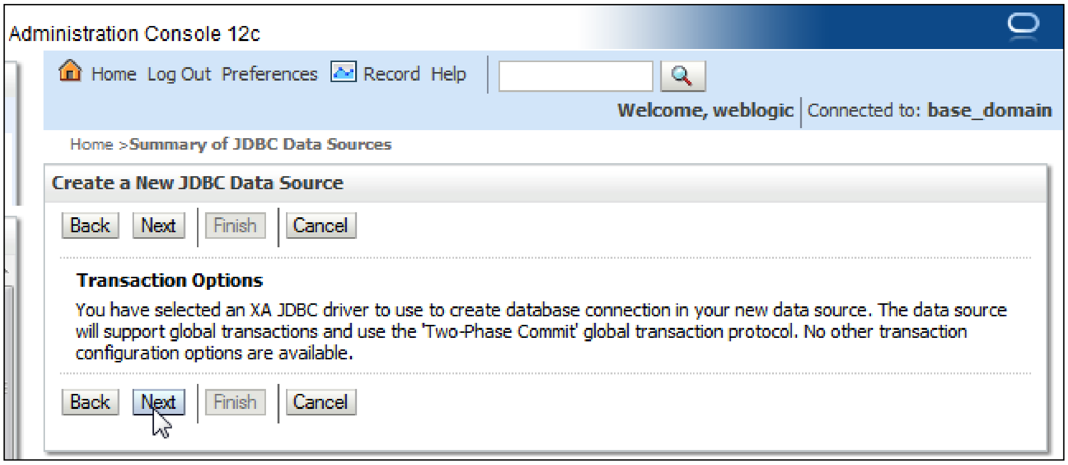
The following properties will be used to identify your new JDBC data source.

Database Type: Oracle

What database driver would you like to use to create database connections? Note: * indicates that the driver is explicitly supported by Oracle WebLogic Server.

Database Driver: *Oracle's Driver (Thin XA) for Instance connections; Versions:9.0.1 and later

By default, an XA JDBC driver supports global transactions and uses the *Two-Phase Commit* global transaction protocol. **Global transactions** are recommended for EJBs using container managed transactions for relation between the JDBC driver (XA or non-XA) transactionality and EJB container managed transactions. Click on **Next** as shown in the following screenshot. (for more information on global transactions, refer `http://docs.oracle.com/cd/E23943_01/web.1111/e13737/transactions.htm`):

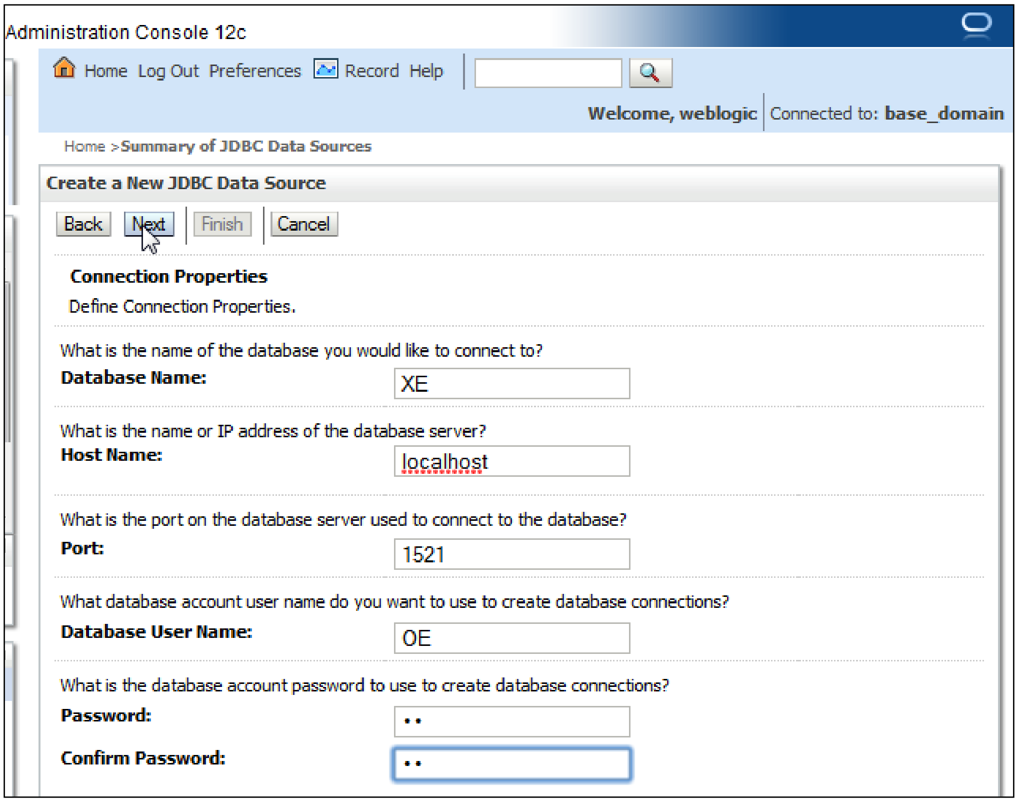

Administration Console 12c
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Home >Summary of JDBC Data Sources
Back Next Finish Cancel

Create a New JDBC Data Source

Transaction Options

You have selected an XA JDBC driver to use to create database connection in your new data source. The data source will support global transactions and use the 'Two-Phase Commit' global transaction protocol. No other transaction configuration options are available.

Specify **Database Name** as `XE`, **Host Name** as `localhost`, **Port** as `1521`, **Database User Name** and **Password** as `OE`, and click on **Next** as shown in the following screenshot:


Administration Console 12c
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Home >Summary of JDBC Data Sources
Back Next Finish Cancel

Create a New JDBC Data Source

Connection Properties

Define Connection Properties.

What is the name of the database you would like to connect to?

Database Name: XE

What is the name or IP address of the database server?

Host Name: localhost

What is the port on the database server used to connect to the database?

Port: 1521

What database account user name do you want to use to create database connections?

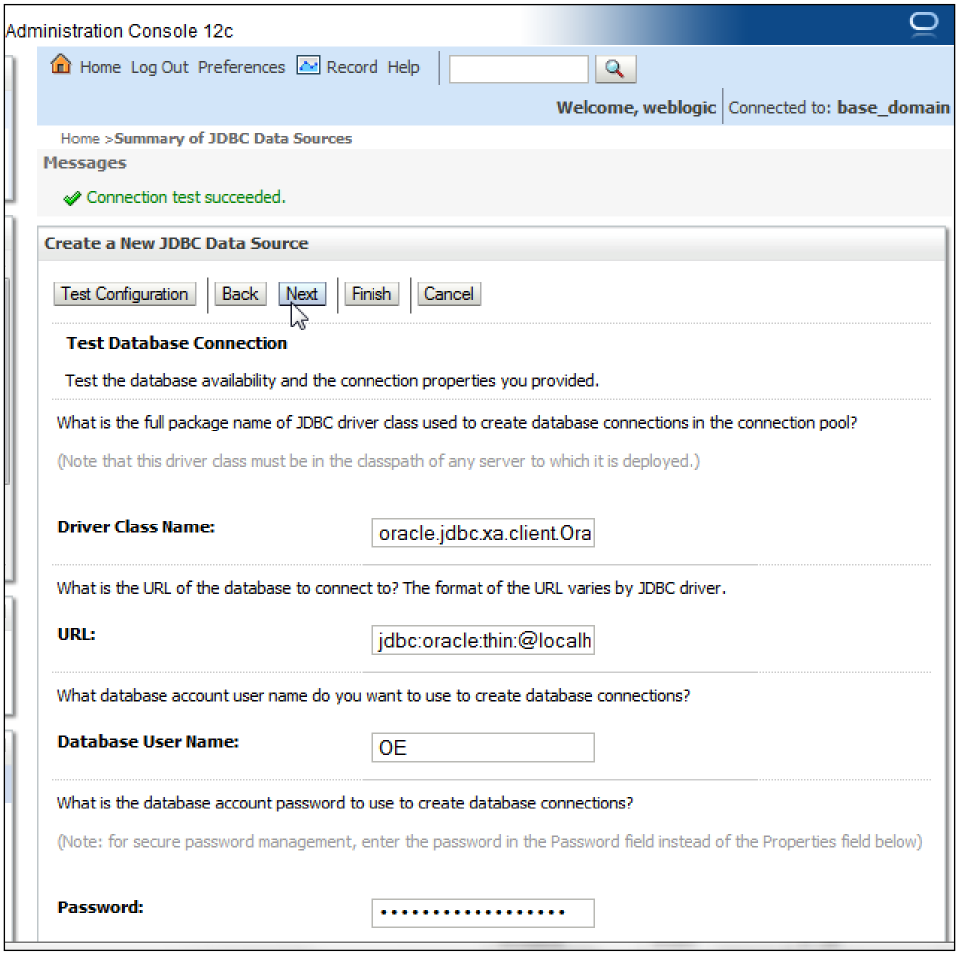
Database User Name: OE

What is the database account password to use to create database connections?

Password: ••

Confirm Password: ••

The **Driver Class Name** textbox and connection **URL** textbox get configured. Click on the **Test Configuration** button to test the database connection. If a connection gets established the message **Connection test succeeded.** gets displayed. Click on **Next** as shown in the following screenshot:


Administration Console 12c
Home Log Out Preferences Record Help
Welcome, weblogic | Connected to: base_domain
Home >Summary of JDBC Data Sources
Test Configuration
Back
Next
Finish
Cancel

Messages

✓ Connection test succeeded.

Create a New JDBC Data Source

Test Database Connection

Test the database availability and the connection properties you provided.

What is the full package name of JDBC driver class used to create database connections in the connection pool?

(Note that this driver class must be in the classpath of any server to which it is deployed.)

Driver Class Name: oracle.jdbc.xa.client.Ora

What is the URL of the database to connect to? The format of the URL varies by JDBC driver.

URL: jdbc:oracle:thin:@localh

What database account user name do you want to use to create database connections?

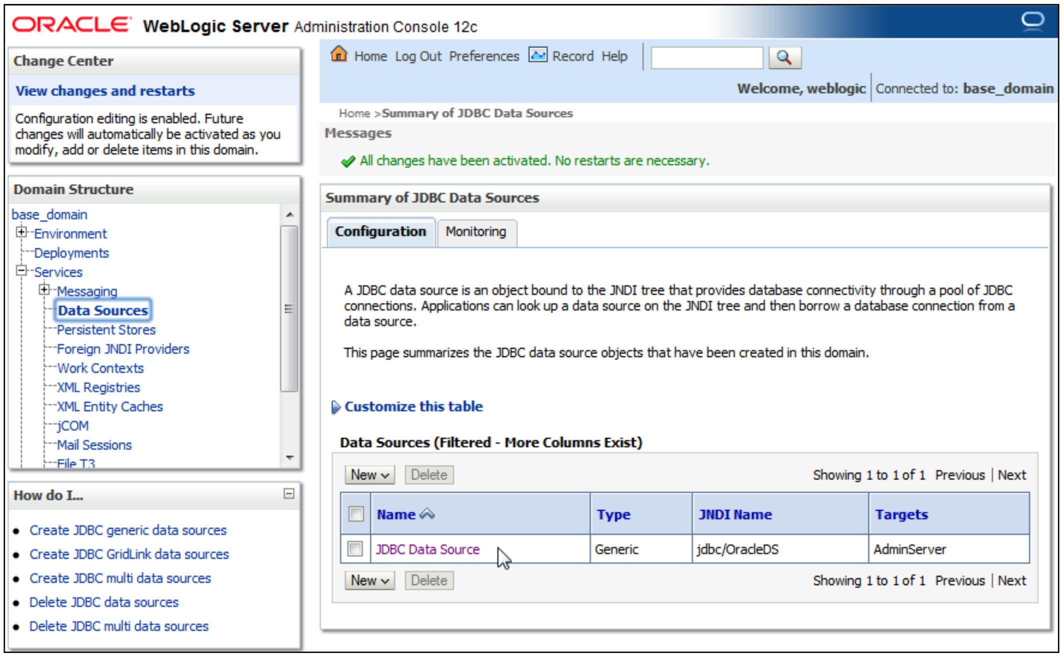
Database User Name: OE

What is the database account password to use to create database connections?

(Note: for secure password management, enter the password in the Password field instead of the Properties field below)

Password: ••••••••••••••••••

In **Select targets**, select the **AdminServer** option and click on **Finish**. A data source gets added to the data sources table. The data source configuration may be modified by clicking on the data source link as shown in the following screenshot:


ORACLE WebLogic Server Administration Console 12c
Home Log Out Preferences Record Help
Welcome, weblogic
Connected to: base_domain
Change Center
View changes and restarts
Domain Structure
base_domain
Environment
Deployments
Services
Messaging
Data Sources
Persistent Stores
Foreign JNDI Providers
Work Contexts
XML Registries
XML Entity Caches
jCOM
Mail Sessions
File T3
How do I...
Create JDBC generic data sources
Create JDBC GridLink data sources
Create JDBC multi data sources
Delete JDBC data sources
Delete JDBC multi data sources
Home >Summary of JDBC Data Sources
Messages
Summary of JDBC Data Sources
Configuration
Monitoring
Customize this table
New
Delete
Showing 1 to 1 of 1 Previous | Next

Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

✓ All changes have been activated. No restarts are necessary.

A JDBC data source is an object bound to the JNDI tree that provides database connectivity through a pool of JDBC connections. Applications can look up a data source on the JNDI tree and then borrow a database connection from a data source.

This page summarizes the JDBC data source objects that have been created in this domain.

Data Sources (Filtered - More Columns Exist)

□	Name	Type	JNDI Name	Targets
□	JDBC Data Source	Generic	jdbc/OracleDS	AdminServer

Creating tables in the Oracle database

We need to create database tables for database persistence. Create database tables `CATALOG`, `EDITION`, `SECTION`, and `ARTICLE` with the following SQL script; the script can be run from the SQL command line:

```
CREATE TABLE CATALOG (id INTEGER PRIMARY KEY NOT NULL,
journal VARCHAR(100));
CREATE TABLE EDITION (id INTEGER PRIMARY KEY NOT NULL,
edition VARCHAR(100));
CREATE TABLE SECTION (id VARCHAR(100) PRIMARY KEY NOT NULL,
sectionName VARCHAR(100));
CREATE TABLE ARTICLE(id INTEGER PRIMARY KEY NOT NULL,
title VARCHAR(100));
```

As Oracle database does not support the autoincrement of primary keys, we need to create sequences for autoincrementing, one for each table. Create sequences `CATALOG_SEQ`, `EDITION_SEQ`, `SECTION_SEQ`, and `ARTICLE_SEQ` with the following SQL script.

```
CREATE SEQUENCE CATALOG_SEQ MINVALUE 1 START WITH 1 INCREMENT BY 1
NOCACHE;
CREATE SEQUENCE EDITION_SEQ MINVALUE 1 START WITH 1 INCREMENT BY 1
NOCACHE;
CREATE SEQUENCE SECTION_SEQ MINVALUE 1 START WITH 1 INCREMENT BY 1
NOCACHE;
CREATE SEQUENCE ARTICLE_SEQ MINVALUE 1 START WITH 1 INCREMENT BY 1
NOCACHE;
```

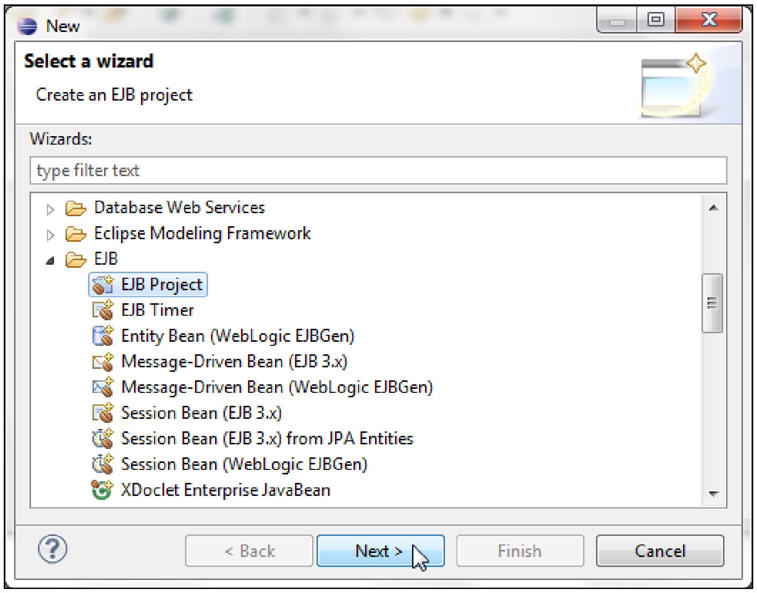
We also need to create join tables between tables. Create join tables using the following SQL script:

```
CREATE TABLE  CATALOGEDITIONS(catalogId INTEGER, editionId INTEGER);
CREATE TABLE EditionCatalog(editionId INTEGER, catalogId INTEGER);
CREATE TABLE EditionSections (editionId INTEGER, sectionId INTEGER);
CREATE TABLE SectionEdition (sectionId INTEGER, editionId INTEGER);
CREATE TABLE SectionArticles(sectionId INTEGER, articleId INTEGER);
CREATE TABLE ArticleSection(articleId INTEGER, sectionId INTEGER);
```

Creating an EJB project

Now, we shall create an EJB project to create entity beans.

In Eclipse, go to **File** | **New** | **Other** to create an EJB project. In the **New** wizard, select **EJB Project** from the **EJB** folder and click on **Next** as shown in the following screenshot:

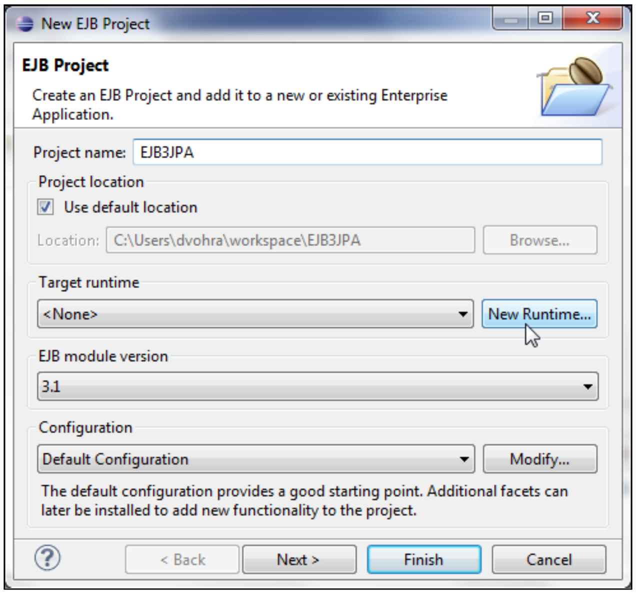


Select a wizard

Create an EJB project

New
Wizards:
type filter text
Database Web Services
Eclipse Modeling Framework
EJB
EJB Project
EJB Timer
Entity Bean (WebLogic EJBGen)
Message-Driven Bean (EJB 3.x)
Message-Driven Bean (WebLogic EJBGen)
Session Bean (EJB 3.x)
Session Bean (EJB 3.x) from JPA Entities
Session Bean (WebLogic EJBGen)
XDoclet Enterprise JavaBean
< Back
Next >
Finish
Cancel

Specify a **Project name** and click on **New Runtime** to configure a target runtime for **Oracle WebLogic Server 12c** if not already configured, as shown in the following screenshot:



EJB Project

Create an EJB Project and add it to a new or existing Enterprise Application.

Project name: EJB3JPA

Project location

☑ Use default location

Location: C:\Users\dvohra\workspace\EJB3JPA

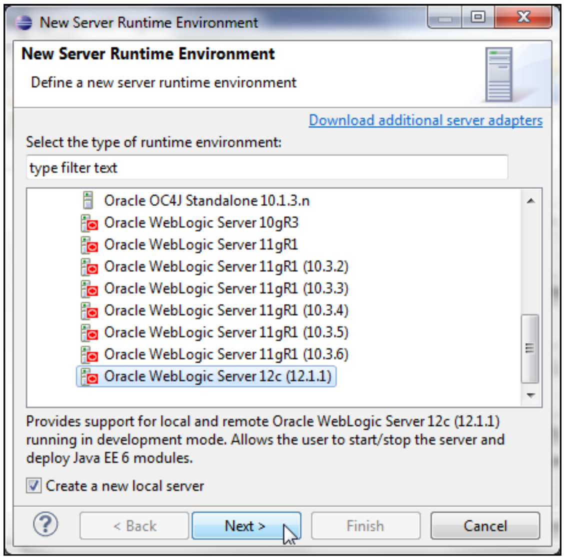
Target runtime: <None>

EJB module version: 3.1

Configuration: Default Configuration

The default configuration provides a good starting point. Additional facets can later be installed to add new functionality to the project.

In **New Server Runtime Environment**, select the **Oracle WebLogic Server 12c (12.1.1)** server, tick **Create a new local server** checkbox, and then click on **Next** as shown in the following screenshot:


New Server Runtime Environment
New Server Runtime Environment

Define a new server runtime environment

Download additional server adapters

Select the type of runtime environment:

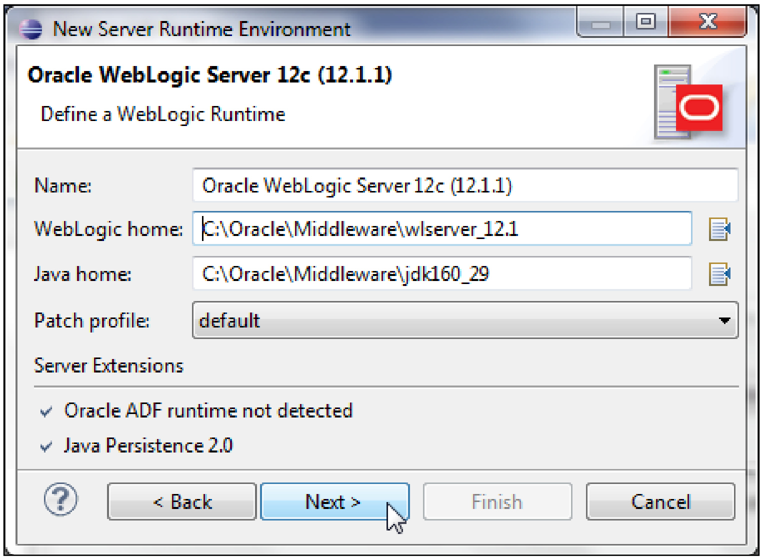
type filter text

- Oracle OC4J Standalone 10.1.3.n
- Oracle WebLogic Server 10gR3
- Oracle WebLogic Server 11gR1
- Oracle WebLogic Server 11gR1 (10.3.2)
- Oracle WebLogic Server 11gR1 (10.3.3)
- Oracle WebLogic Server 11gR1 (10.3.4)
- Oracle WebLogic Server 11gR1 (10.3.5)
- Oracle WebLogic Server 11gR1 (10.3.6)
- Oracle WebLogic Server 12c (12.1.1)

Provides support for local and remote Oracle WebLogic Server 12c (12.1.1) running in development mode. Allows the user to start/stop the server and deploy Java EE 6 modules.

☑ Create a new local server

Select the **WebLogic home** directory, and the **Java home** directory also gets specified. Click on **Next** as shown in the following screenshot:


New Server Runtime Environment

Oracle WebLogic Server 12c (12.1.1)

Define a WebLogic Runtime

Name: Oracle WebLogic Server 12c (12.1.1)

WebLogic home: C:\Oracle\Middleware\wlserver_12.1

Java home: C:\Oracle\Middleware\jdk160_29

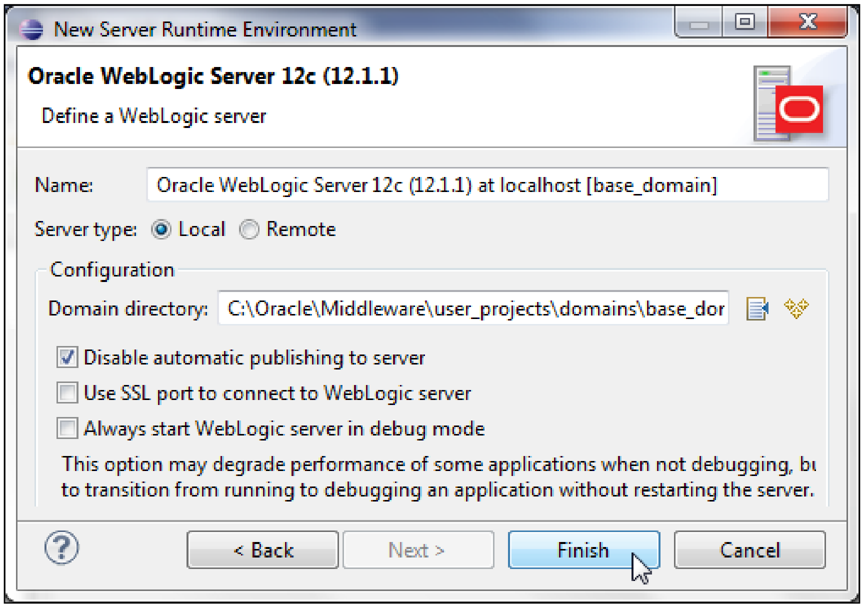
Patch profile: default

Server Extensions

✓ Oracle ADF runtime not detected

✓ Java Persistence 2.0

Select **Server Type** as **Local** and then select **Domain Directory** as `C:\Oracle\Middleware\user_project\domains\base_domain`. Click on **Finish** as shown in the screenshot:


New Server Runtime Environment

Oracle WebLogic Server 12c (12.1.1)

Define a WebLogic server

Name: Oracle WebLogic Server 12c (12.1.1) at localhost [base_domain]

Server type: ● Local ○ Remote

Configuration

Domain directory: C:\Oracle\Middleware\user_projects\domains\base_dor

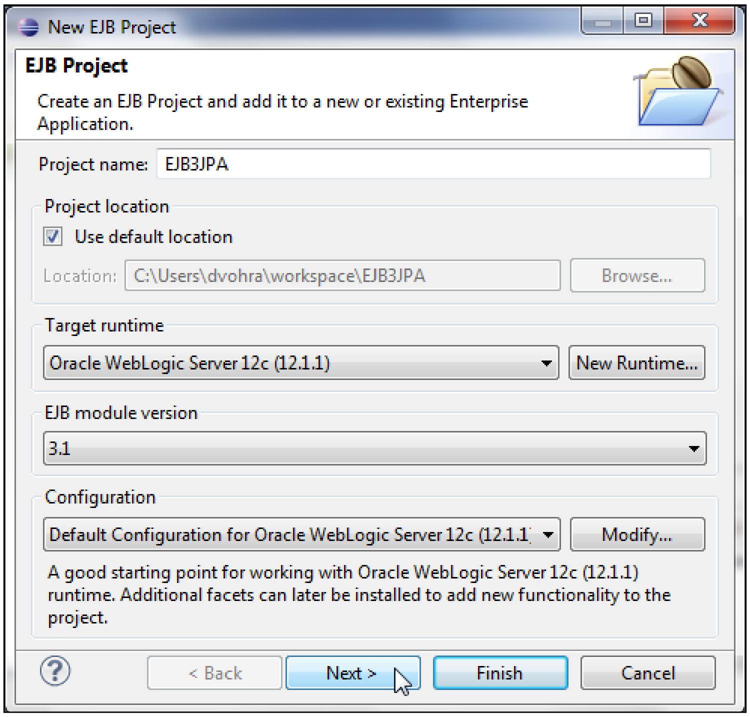
☑ Disable automatic publishing to server

□ Use SSL port to connect to WebLogic server

□ Always start WebLogic server in debug mode

This option may degrade performance of some applications when not debugging, bu to transition from running to debugging an application without restarting the server.

The **Target runtime** server gets configured. Select **EJB module version** as **3.1**. Select the default **Configuration** and click on **Next** as shown in the following screenshot:



New EJB Project

EJB Project

Create an EJB Project and add it to a new or existing Enterprise Application.

Project name: EJB3JPA

Project location

☑ Use default location

Location: C:\Users\dvohra\workspace\EJB3JPA

Target runtime

Oracle WebLogic Server 12c (12.1.1)

EJB module version

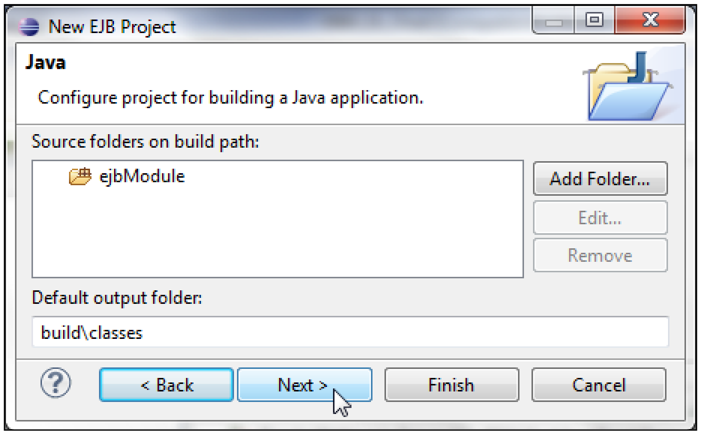
3.1

Configuration

Default Configuration for Oracle WebLogic Server 12c (12.1.1

A good starting point for working with Oracle WebLogic Server 12c (12.1.1) runtime. Additional facets can later be installed to add new functionality to the project.

Select the default Java configuration for **Source folders on build path** as **ejbModule** and **Default output folder** as `build/classes`, and click on **Next** as shown in the following screenshot:



New EJB Project

Java

Configure project for building a Java application.

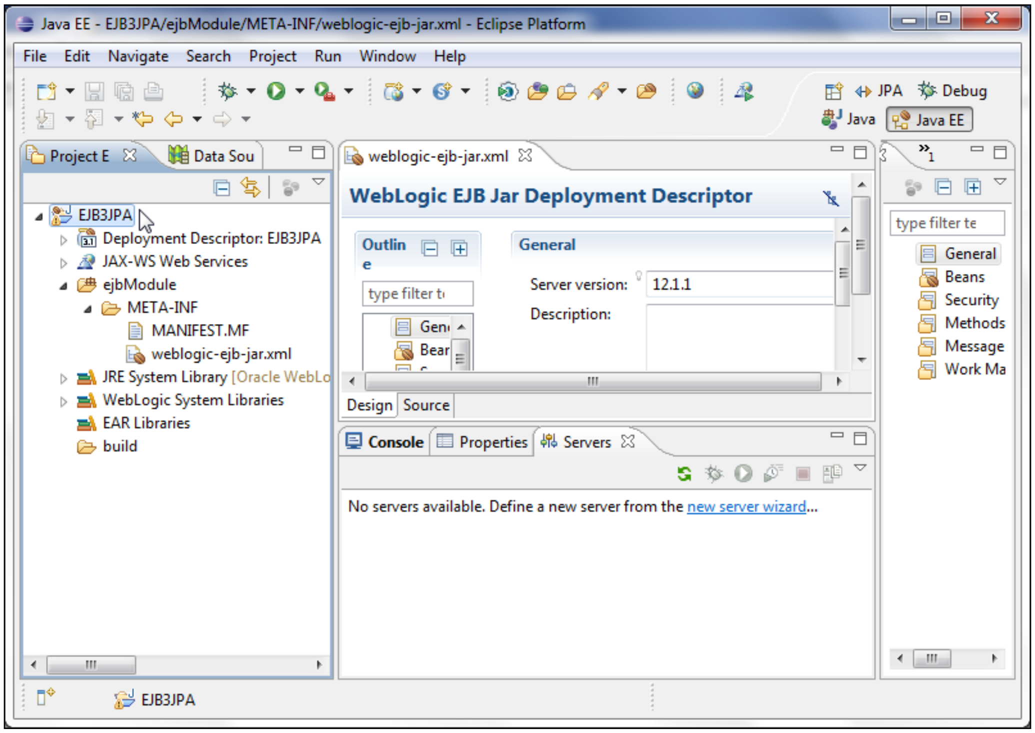
Source folders on build path:

ejbModule

Default output folder:

build\classes

Select the default EJB module configuration and click on **Finish**. An EJB project gets created. The EJB project does not contain any EJBs, which we shall add in subsequent sections.



Java EE - EJB3JPA/ejbModule/META-INF/weblogic-ejb-jar.xml - Eclipse Platform

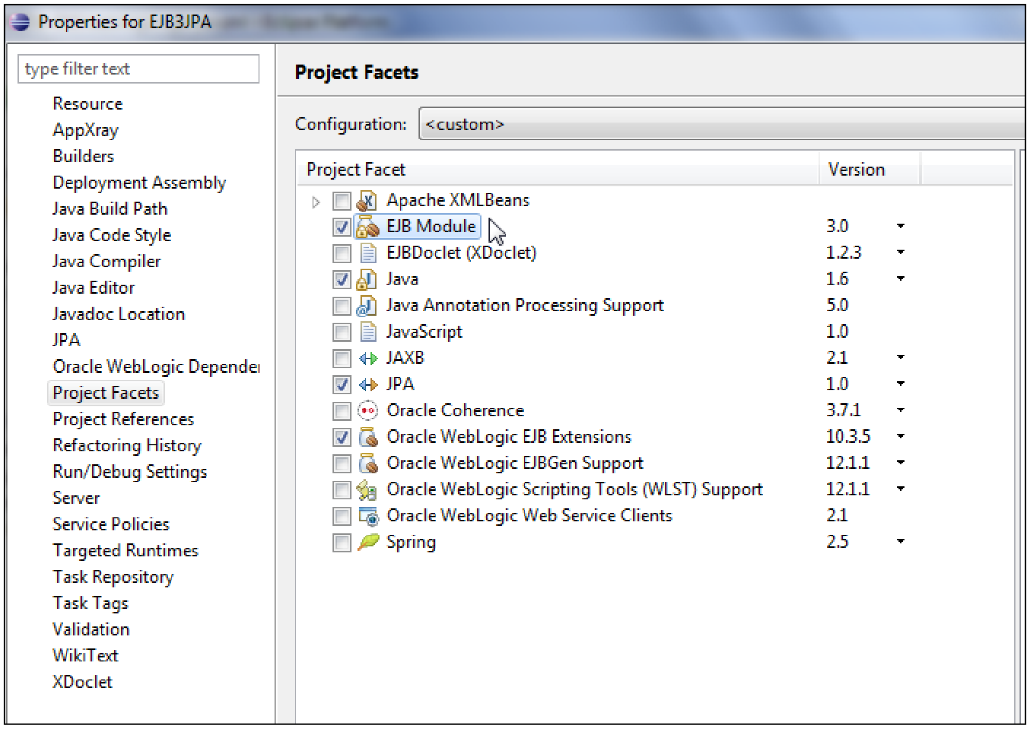
WebLogic EJB Jar Deployment Descriptor

Server version: 12.1.1

Description:

No servers available. Define a new server from the new server wizard...

Right-click on the project node in the **Project Explorer** tab and select **Project Properties**. Select **Project Facets** in the **Properties** window. The EJB project should have the **EJB Module** project facet enabled as shown in the following screenshot:



Project Facets

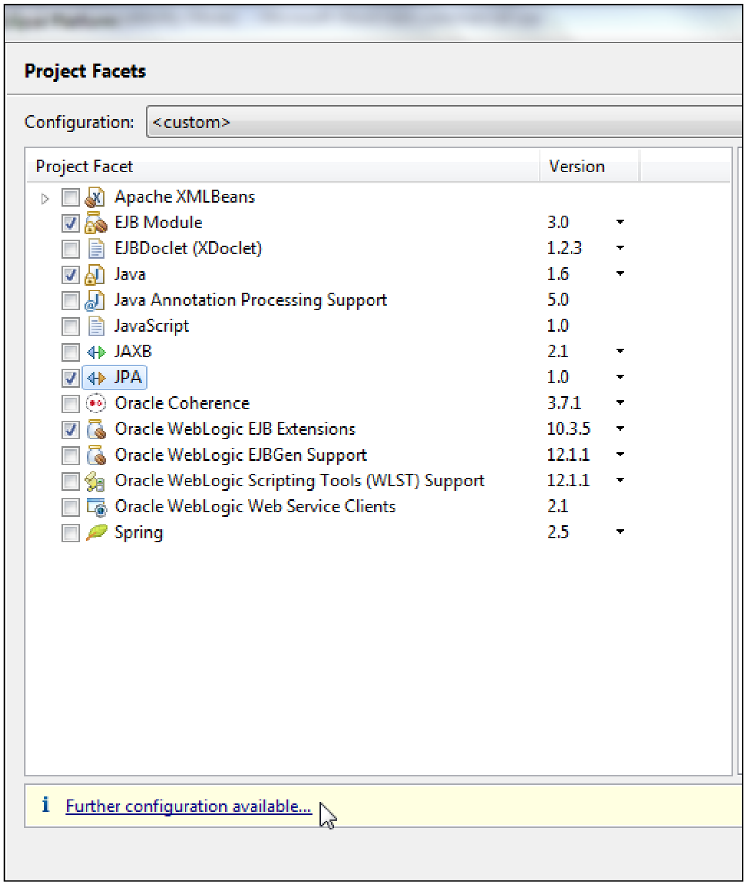
Configuration: <custom>

Project Facet	Version
□ Apache XMLBeans	
☑ EJB Module	3.0
□ EJBDoclet (XDoclet)	1.2.3
☑ Java	1.6
□ Java Annotation Processing Support	5.0
□ JavaScript	1.0
□ JAXB	2.1
☑ JPA	1.0
□ Oracle Coherence	3.7.1
☑ Oracle WebLogic EJB Extensions	10.3.5
□ Oracle WebLogic EJBGen Support	12.1.1
□ Oracle WebLogic Scripting Tools (WLST) Support	12.1.1
□ Oracle WebLogic Web Service Clients	2.1
□ Spring	2.5

Session beans require an EJB project and entity beans require the JPA project facet for database persistence. We have created an EJB project but this EJB project does not have the **JPA** project facet enabled by default. In the next section, we shall add the **JPA** facet to the EJB project.

Adding the JPA facet

We require the JPA project facet to create entity beans. We could have created a JPA project to start with, but to create a session bean facade we first created an EJB project; session beans require an EJB project by default. To add the JPA project facet, right-click on the project in **Project Explorer** and select **Properties**. Select the **Project Facets** node and select the **JPA 1.0** project facet. Click on the **Further configuration available** link as shown in the screenshot:



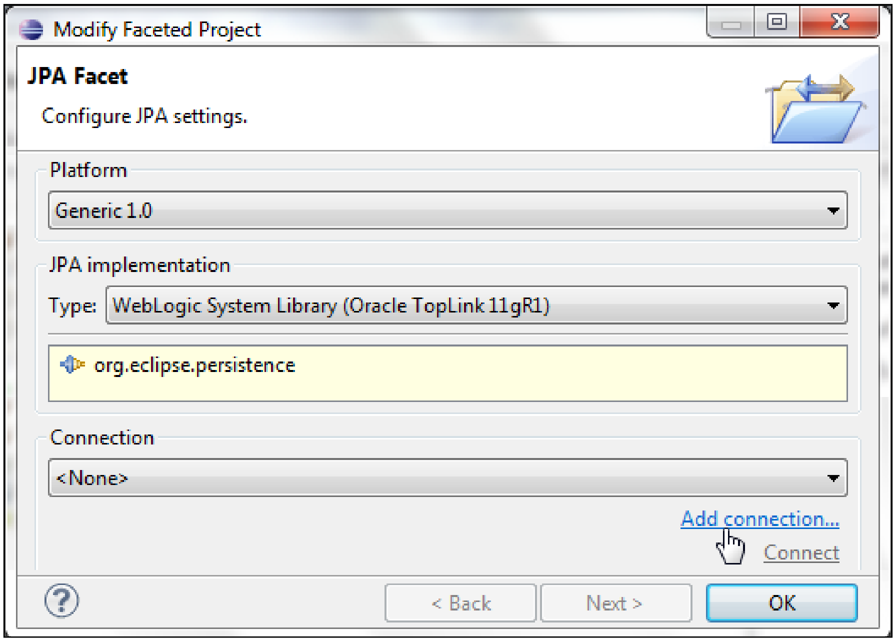
Project Facets

Configuration: <custom>

Project Facet	Version
☐ Apache XMLBeans	
☑ EJB Module	3.0
☐ EJBDoclet (XDoclet)	1.2.3
☑ Java	1.6
☐ Java Annotation Processing Support	5.0
☐ JavaScript	1.0
☐ JAXB	2.1
☑ JPA	1.0
☐ Oracle Coherence	3.7.1
☑ Oracle WebLogic EJB Extensions	10.3.5
☐ Oracle WebLogic EJBGen Support	12.1.1
☐ Oracle WebLogic Scripting Tools (WLST) Support	12.1.1
☐ Oracle WebLogic Web Service Clients	2.1
☐ Spring	2.5

Further configuration available...

In **JPA Facet**, select **Platform** as **Generic 1.0**. Select **JPA implementation** as **Oracle TopLink 11g R1**. We also need a database connection for JPA. To configure a new **Connection**, click on the **Add connection** link as shown in the following screenshot:



Modify Faceted Project

JPA Facet

Configure JPA settings.

Platform

Generic 1.0

JPA implementation

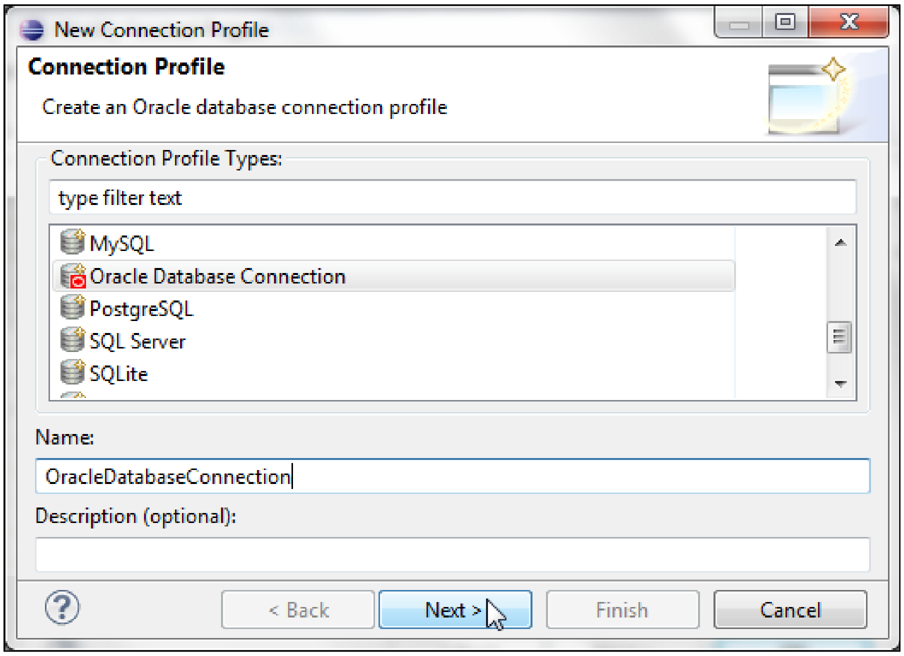
Type: WebLogic System Library (Oracle TopLink 11gR1)

org.eclipse.persistence

Connection

<None>

In **Connection Profile**, select the **Oracle Database Connection** profile, specify a connection **Name** and click on **Next** as shown in the following screenshot:



New Connection Profile

Connection Profile

Create an Oracle database connection profile

Connection Profile Types:

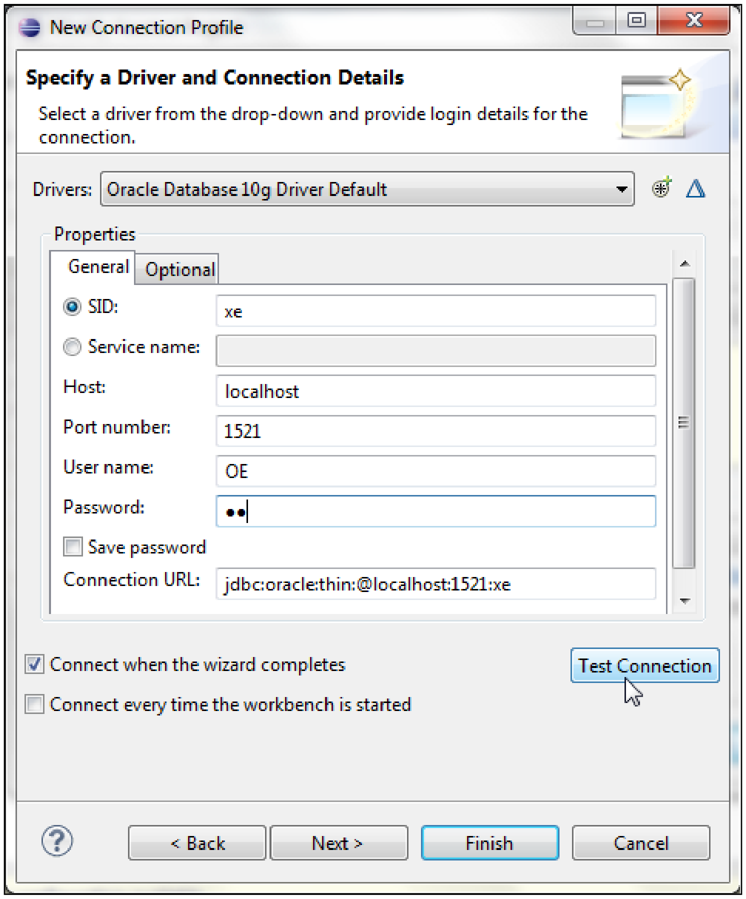
type filter text

- MySQL
- Oracle Database Connection
- PostgreSQL
- SQL Server
- SQLite

Name: OracleDatabaseConnection

Description (optional):

In the **Specify a Driver and Connection Details** window, select the driver as **Oracle Database 10g Driver**. Specify **SID** as `XE`, **Host** as `localhost`, **Port number** as `1521`, **User name** as `OE`, and **Password** as `OE`. The **Connection URL** gets specified. Now, click on **Test Connection** as shown in the following screenshot:



New Connection Profile

Specify a Driver and Connection Details

Select a driver from the drop-down and provide login details for the connection.

Drivers: Oracle Database 10g Driver Default

Properties

General | Optional

● SID: xe

○ Service name:

Host: localhost

Port number: 1521

User name: OE

Password: ••

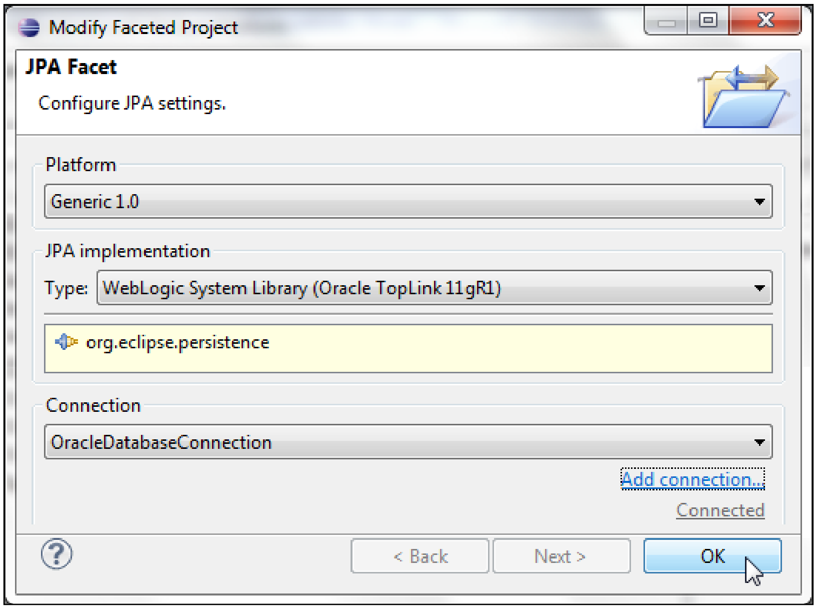
□ Save password

Connection URL: jdbc:oracle:thin:@localhost:1521:xe

☑ Connect when the wizard completes

□ Connect every time the workbench is started

A **Ping succeeded** message indicates that the connection got established. Click on **Next** and then click on **Finish** in **Summary**. A **Connection** for the **JPA Facet** gets configured. Click on **OK** as shown in the following screenshot:



Modify Faceted Project

JPA Facet

Configure JPA settings.

Platform: Generic 1.0

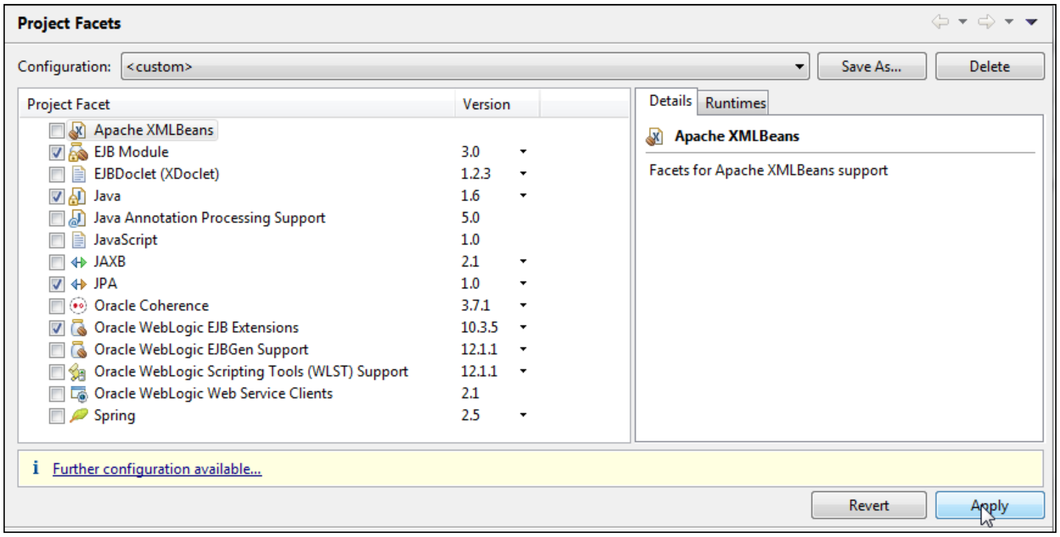
JPA implementation

Type: WebLogic System Library (Oracle TopLink 11gR1)

org.eclipse.persistence

Connection: OracleDatabaseConnection

The connection profile we have configured is for the JPA project facet, not to run client applications to entity beans. The data source we configured in the WebLogic server with JNDI **jdbc/OracleDS** is for running client applications to entity beans. Click on **Apply** in **Properties** to install the **JPA** facet as shown in the following screenshot:



Project Facets

Configuration: <custom>

Project Facet	Version
□ Apache XMLBeans	
☑ EJB Module	3.0
□ EJBDoclet (XDoclet)	1.2.3
☑ Java	1.6
□ Java Annotation Processing Support	5.0
□ JavaScript	1.0
□ JAXB	2.1
☑ JPA	1.0
□ Oracle Coherence	3.7.1
☑ Oracle WebLogic EJB Extensions	10.3.5
□ Oracle WebLogic EJBGen Support	12.1.1
□ Oracle WebLogic Scripting Tools (WLST) Support	12.1.1
□ Oracle WebLogic Web Service Clients	2.1
□ Spring	2.5

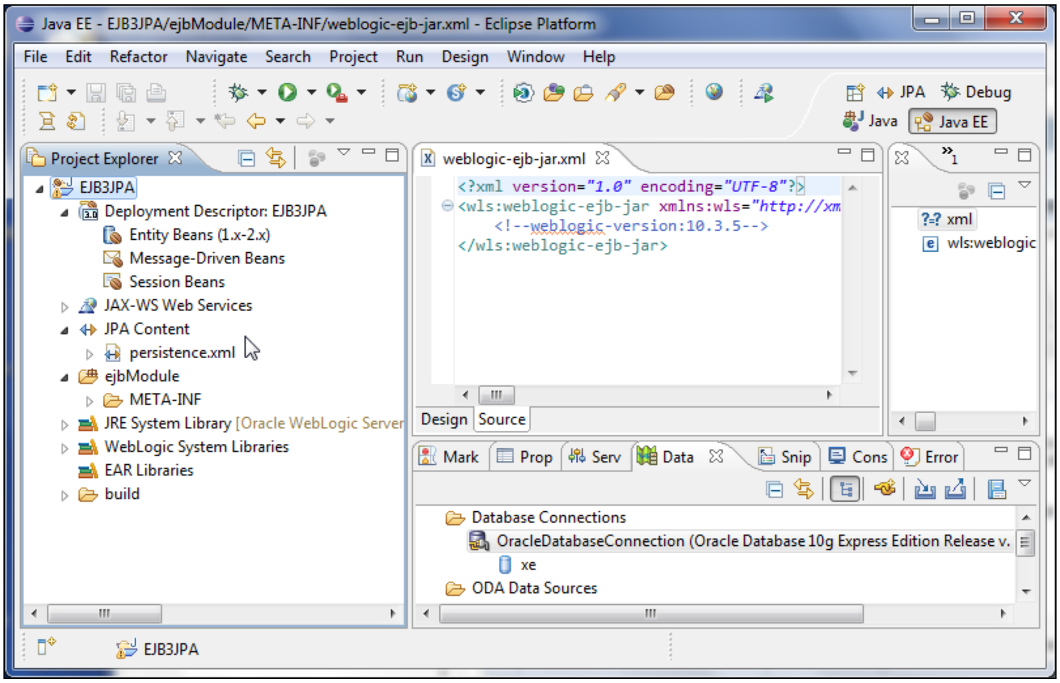
Details | Runtimes

Apache XMLBeans

Facets for Apache XMLBeans support

Further configuration available...

A node for **JPA Content** gets added to the EJB project. A **persistence.xml** configuration file gets added.

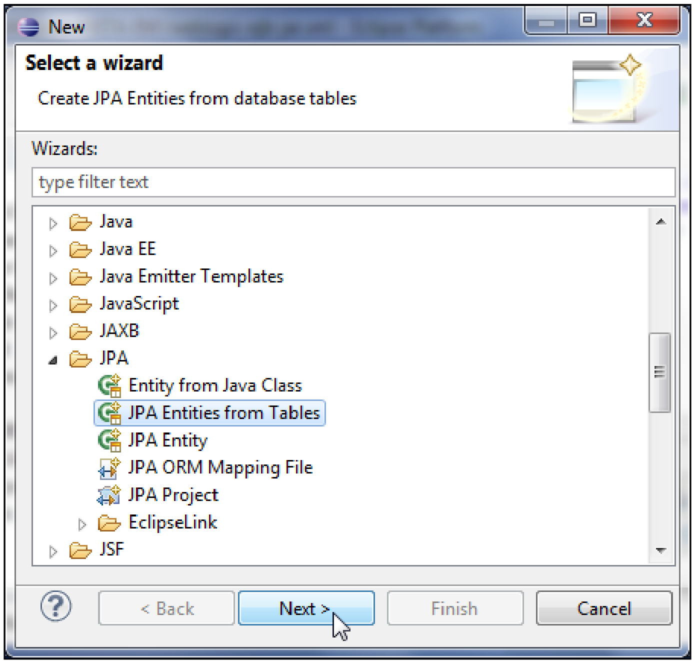


Java EE - EJB3JPA/ejbModule/META-INF/weblogic-ejb-jar.xml - Eclipse Platform
File Edit Refactor Navigate Search Project Run Design Window Help
JPA
Debug
Java
Java EE
Project Explorer
EJB3JPA
Deployment Descriptor: EJB3JPA
Entity Beans (1.x-2.x)
Message-Driven Beans
Session Beans
JAX-WS Web Services
JPA Content
persistence.xml
ejbModule
META-INF
JRE System Library [Oracle WebLogic Server
WebLogic System Libraries
EAR Libraries
build
weblogic-ejb-jar.xml
Design
Source
?-? xml
wls:weblogic
Mark
Prop
Serv
Data
Snip
Cons
Error
Database Connections
OracleDatabaseConnection (Oracle Database 10g Express Edition Release v.
xe
ODA Data Sources
EJB3JPA

```
<?xml version="1.0" encoding="UTF-8"?>
<wls:weblogic-ejb-jar xmlns:wls="http://xm
    <!--weblogic-version:10.3.5-->
</wls:weblogic-ejb-jar>
```

Creating entity beans from tables

In this section, we shall create entities from database tables we created earlier. Select the project node in **Project Explorer** and go to **File** | **New** | **Other**. In the **New** wizard window, select **JPA Entites from Tables** from the **JPA** folder as shown in the following screenshot. Click on **Next**. Alternatively, you can right-click on the project node in **Project Explorer** and select **Generate Entities from Tables** from **JPA Tools**.



New

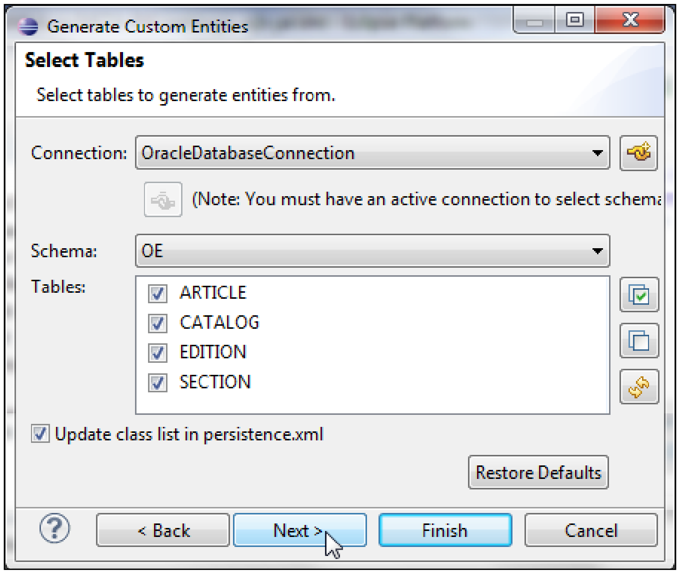
Select a wizard

Create JPA Entities from database tables

Wizards:

type filter text

In **Select Tables**, select the database connection configured when adding the JPA project facet. Select the **OE** Schema. Select the **CATALOG**, **EDITION**, **SECTION**, and **ARTICLE** tables. Select the checkbox **Update class list in persistence.xml** and click on **Next** as shown in the following screenshot:



Generate Custom Entities

Select Tables

Select tables to generate entities from.

Connection: OracleDatabaseConnection

(Note: You must have an active connection to select schema

Schema: OE

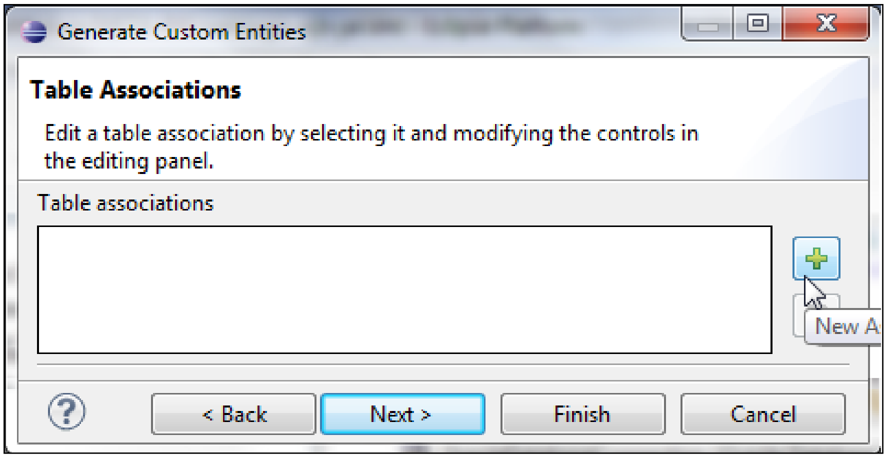
Tables:

☑ ARTICLE
☑ CATALOG
☑ EDITION
☑ SECTION

☑ Update class list in persistence.xml

Defining entity relationships

The entities to be generated have relationships between them. The `Catalog` entity has a one-to-many relationship with the `Edition` entity. The `Edition` entity has a one-to-many relationship with the `Section` entity and the `Section` entity has a further one-to-many relationship with the `Article` entity. In **Table Associations**, we shall define the associations between the tables. Click on the **+** button to create an association as shown in the following screenshot:



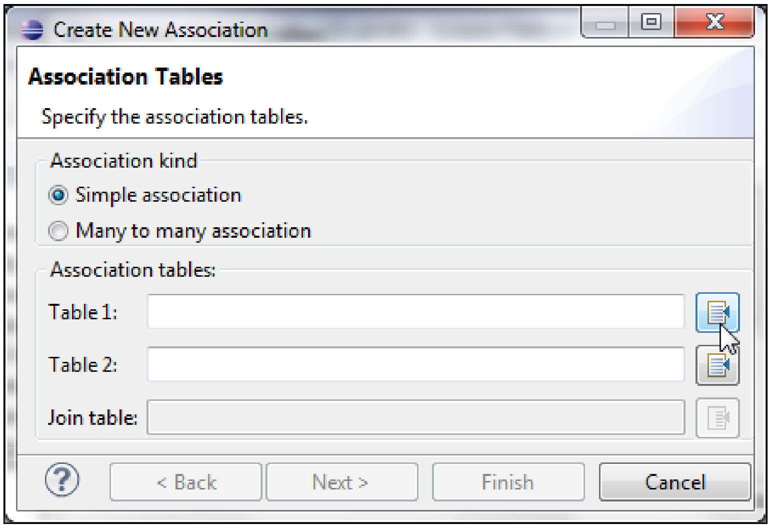
Generate Custom Entities

Table Associations

Edit a table association by selecting it and modifying the controls in the editing panel.

Table associations

In **Association Tables**, select the tables to create an association between them. We will need to create an association for each of the relationships. Click on the button for the **Table 1** field as shown in the following screenshot:



Create New Association

Association Tables

Specify the association tables.

Association kind

● Simple association

○ Many to many association

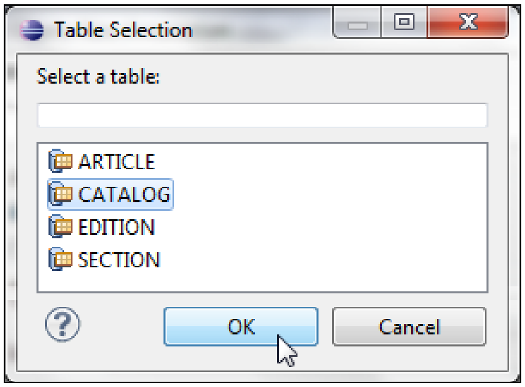
Association tables:

Table 1:

Table 2:

Join table:

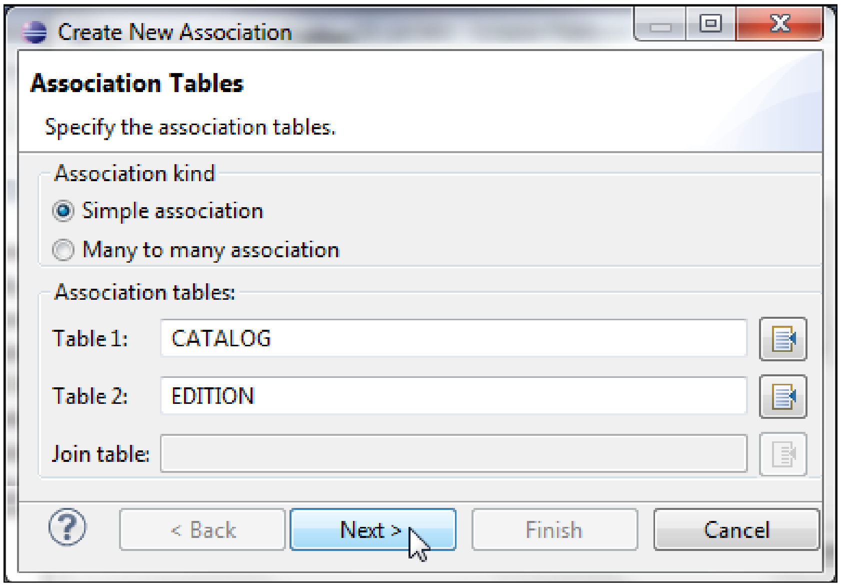
Select the **CATALOG** table and click on **OK**, as shown in the following screenshot:



Select a table:

- ARTICLE
- CATALOG
- EDITION
- SECTION

Similarly, select **EDITION** as **Table 2**. The **Association kind** is **Simple association** by default, which is what we need. Now, click on **Next** as shown in the following screenshot:



Association Tables

Specify the association tables.

Association kind

● Simple association

○ Many to many association

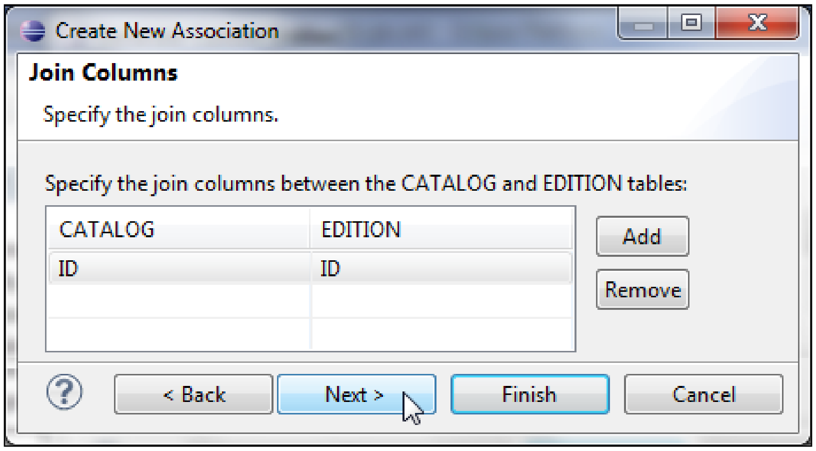
Association tables:

Table 1: CATALOG

Table 2: EDITION

Join table:

Specify the join columns between the **CATALOG** and **EDITION** tables as **ID** using the **Add** button and click on **Next** as shown in the following screenshot:



Create New Association

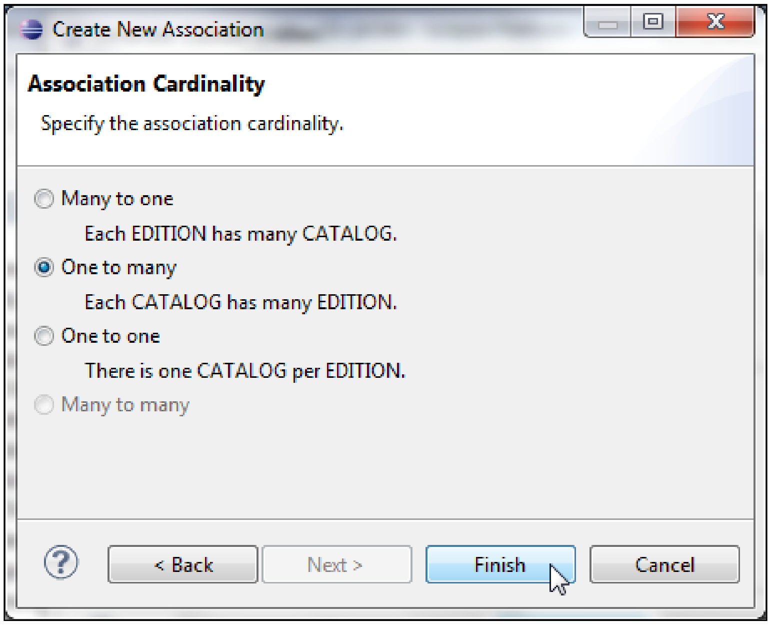
Join Columns

Specify the join columns.

Specify the join columns between the CATALOG and EDITION tables:

CATALOG	EDITION
ID	ID

As the `Catalog` entity has a one-to-many relation with the `Edition` entity, in **Association Cardinality** select **One to many** and click on **Finish** as shown in the following screenshot:



Create New Association

Association Cardinality

Specify the association cardinality.

○ Many to one

Each EDITION has many CATALOG.

● One to many

Each CATALOG has many EDITION.

○ One to one

There is one CATALOG per EDITION.

○ Many to many

The table association between the **CATALOG** and **EDITION** tables gets defined and the table join also gets defined.