

Dali Java Persistence Tools

User Guide

Release 3.2

September 2012



Dali Java Persistence Tools User Guide

Release 3.2

Copyright © 2011, 2012, Oracle and/or its affiliates. All rights reserved.

The Eclipse Foundation makes available all content in this plug-in ("Content"). Unless otherwise indicated below, the Content is provided to you under the terms and conditions of the Eclipse Public License Version 1.0 ("EPL"). A copy of the EPL is available at <http://www.eclipse.org/legal/epl-v10.html>. For purposes of the EPL, "Program" will mean the Content.

If you did not receive this Content directly from the Eclipse Foundation, the Content is being redistributed by another party ("Redistributor") and different terms and conditions may apply to your use of any object code in the Content. Check the Redistributor's license that was provided with the Content. If no such license exists, contact the Redistributor. Unless otherwise indicated below, the terms and conditions of the EPL still apply to any source code in the Content.

Contents

1 Getting started

Requirements and installation	1-1
Dali quick start	1-2
Creating a new JPA project	1-2
Creating a Java persistent entity with persistent fields	1-3

2 Concepts

Understanding Java persistence	2-1
Understanding OR mappings	2-1
Understanding Java Persistence API	2-2
The persistence.xml file	2-2
The orm.xml file	2-2
Understanding Java Architecture for XML Binding	2-2

3 Tasks

Creating a new JPA project	3-1
Creating a new JAXB project	3-5
Creating Database Web Services from Builder XML	3-7
Converting a Java project to a JPA project	3-9
Creating a JPA entity	3-11
Adding persistence to a class	3-15
Entity	3-15
Embeddable	3-17
Mapped superclass	3-18
Adding virtual attributes	3-19
Managing the persistence.xml file	3-20
Synchronizing classes	3-28
Managing the orm.xml file	3-28
Creating an orm.xml file	3-29
Working with orm.xml file	3-30
Specifying additional tables	3-31
Specifying entity inheritance	3-32
Creating queries	3-33
Mapping an entity	3-34
Basic mapping	3-35

Element collection mapping	3-36
Embedded mapping	3-37
Embedded ID mapping	3-38
ID mapping	3-39
Many-to-many mapping	3-40
Many-to-one mapping	3-41
One-to-many mapping	3-42
One-to-one mapping	3-43
Transient mapping	3-44
Version mapping	3-44
Generating entities from tables	3-45
Generating tables from entities	3-50
Generating dynamic entities from tables	3-52
Modifying persistent project properties	3-56
Converting JPA metadata to XML	3-56
Validating mappings and reporting problems	3-57
Error messages	3-58
Warning messages	3-61

4 Reference

Wizards	4-1
Generate Entities from Tables wizard	4-1
Select Tables	4-2
Table Associations	4-2
Customize Default Entity Generation	4-2
Customize Individual Entities	4-3
Generate Dynamic Entities from Tables wizard	4-3
Select Tables	4-4
Table Associations	4-4
Customize Default Entity Generation	4-4
Customize Individual Entities	4-5
Create JPA Entity wizard	4-5
Entity Class page	4-5
Entity Properties page	4-6
Create ORM Mapping File wizard	4-7
Mapping File Location	4-7
Mapping File Options	4-7
Create New JPA Project wizard	4-8
New JPA Project page	4-8
Java Page	4-9
JPA Facet page	4-9
Create New JAXB Project wizard	4-10
New JAXB Project page	4-10
Java Page	4-11
JAXB Facet page	4-11
New Database Web services from Builder XML wizard	4-11
Web Dynamic page	4-11

Select Builder XML File page	4-11
Driver Files page	4-12
Generate Tables from Entities wizard	4-12
Schema Generation	4-12
Create New Association wizard	4-12
Association Tables	4-12
Join Columns	4-13
Association Cardinality	4-13
Property pages	4-13
JPA Details view (for entities)	4-13
Entity	4-14
Embeddable	4-14
Mapped Superclass	4-15
Caching	4-15
Queries	4-17
Inheritance	4-18
Attribute Overrides	4-19
Multitenancy	4-19
Primary Key Generation	4-20
Secondary tables	4-21
Converters	4-21
Advanced	4-22
JPA Details view (for attributes)	4-22
Basic Mapping	4-23
Element Collection Mapping	4-24
Embedded Mapping	4-24
Embedded ID Mapping	4-25
ID Mapping	4-25
Many-to-Many Mapping	4-25
Many-to-One Mapping	4-26
One-to-Many Mapping	4-27
One-to-One Mapping	4-28
Version Mapping	4-29
Type information	4-30
Value	4-30
Converters	4-30
Ordering	4-31
Joining Strategy	4-32
Derived Identity	4-32
Primary Key Generation information	4-33
JPA Details view (for orm.xml)	4-34
Entity Mappings	4-34
Persistence Unit	4-35
Generators	4-35
Queries	4-36
Converters	4-36
JPA Structure view	4-36

persistence.xml Editor	4-37
General	4-37
Connection	4-38
Customization	4-40
Caching	4-42
Logging	4-44
Options	4-47
Schema Generation	4-49
Properties	4-50
Source	4-50
Preferences	4-50
Java Persistence Preferences page – JPA	4-50
Java Persistence Preferences page – Errors/Warnings	4-51
Project Properties page – JPA	4-51
Project Properties page – EclipseLink	4-52
Project Properties page – Entity Generation	4-52
Project Properties page – Errors/Warnings	4-53
Project Properties page – JAXB Options	4-53
Project Properties page – Schemas	4-53
Dialogs	4-54
Edit Join Columns dialog	4-54
Add Join Column dialog	4-54
Select Cascade dialog	4-55
New EclipseLink Mapping File dialog	4-55
Add Converter dialog	4-55
Mapping Type Selection dialog	4-56
JPA Metadata Conversion dialog	4-56
Make Persistent dialog	4-56
Add Query dialog	4-56
Add Primary Key Join Column dialog	4-57
Add Schema Location dialog	4-57
Select Schema Location dialog	4-57
Add Virtual Attribute dialog	4-57
JPA Development perspective	4-57
Icons and buttons	4-58
Icons	4-58
Buttons	4-59
Dali developer documentation	4-59

5 Tips and tricks

6 What's new

EclipseLink multitenancy support	6-1
EclipseLink static weaving support	6-2
Generating EclipseLink dynamic entities from tables	6-2
Converting JPA metadata to XML	6-3
EclipseLink 2.4 support	6-3

7 Legal

About this content..... 7-1

Index

Getting started

This section provides information on getting started with the Java Persistence Tools.

- [Requirements and installation](#)
- [Dali quick start](#)

For additional information, please visit the Dali home page at:
<http://www.eclipse.org/webtools/dali/>.

Requirements and installation

Before installing Dali, ensure that your environment meets the following *minimum* requirements:

- Eclipse 3.7 (<http://www.eclipse.org/downloads>)
- Java Runtime Environment (JRE) 1.5 (<http://java.com>)
- Eclipse Web Tools Platform (WTP) 3.3 (<http://www.eclipse.org/webtools>)
- Java Persistence API (JPA) for Java EE 5. For example, the EclipseLink implementation for JPA can be obtained from:
<http://www.eclipse.org/eclipselink/>

Refer to http://www.eclipse.org/webtools/dali/gettingstarted_main.html for additional installation information.

Dali is included as part of WTP 3.4. No additional installation or configuration is required.

Accessibility Features

Dali supports the standard accessibility features in Eclipse, including the following:

- Navigating the user interface using the keyboard.
- Specifying general accessibility preferences for the editor.

See Accessibility Features in Eclipse in the *Workbench User Guide* for details.

Help Accessibility

The documentation and help contains markup to facilitate access by the disabled community. See Help Accessibility in the *Workbench User Guide* for details.

When using the help, be aware of the following:

- Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an

otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

- This documentation may contain links to Web sites of other companies or organizations that we do not control. We neither evaluate nor make any representations regarding the accessibility of these Web sites.

Dali quick start

This section includes information to help you quickly start using Dali to create relational mappings between Java persistent entities and database tables.

- [Creating a new JPA project](#)
- [Creating a Java persistent entity with persistent fields](#)

Creating a new JPA project

This quick start shows how to create a new JPA project.

1. **Select File > New > Project.** The Select a Wizard dialog appears.

Tip: You can also select the JPA perspective and then select **File > New > JPA Project**.

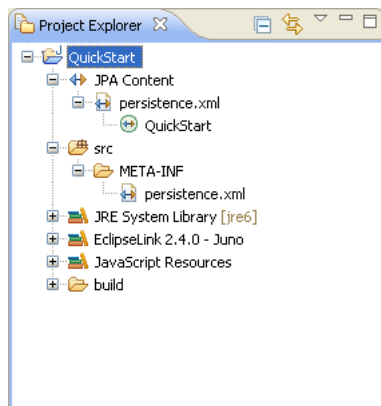
2. Select **JPA Project** and then click **Next**. The [New JPA Project page](#) appears.
3. Enter a **Project name** (such as `QuickStart`).
4. If needed, select the **Target Runtime** (such as `Apache Tomcat`) and configuration, such as **Default Configuration for Apache Tomcat** and then click **Next**. The Java source page appears.

Note: The Target Runtime is not required for Java SE development.

5. If you have existing Java source files, add them to your classpath and then click **Next**. The [JPA Facet page](#) appears.
6. On the JPA Facet dialog, select your vendor-specific JPA platform (or select **Generic 2.0**), JPA implementation library (such as EclipseLink), database connection (or create a new connection), define how Dali should manage persistent classes, and then click **Finish**.

Tip: Select **Override the Default Schema for Connection** if you require a schema other than the one that Dali derives from the connection information, which may be incorrect in some cases. Using this option, you can select a development time schema for defaults and validation.

Eclipse adds the project to the workbench and opens the JPA perspective.

Figure 1–1 JPA Project in Project Explorer

Now that you have created a project with persistence, you can continue with [Creating a Java persistent entity with persistent fields](#).

Creating a Java persistent entity with persistent fields

This quick start shows how to create a new persistent Java entity. We will create an entity to associate with a database table. You will also need to add the ADDRESS table to your database.

1. Select the JPA project in the Navigator or Project Explorer and then click **New > Other**. The Select a Wizard dialog appears.
2. Select **JPA > Entity** and then click **Next**. The [Entity Class page](#) appears.
3. Enter the package name (such as `quickstart.demo.model`), the class name (such as `Address`) and then click **Next**. The [Entity Properties page](#) appears, which enables you to define the persistence fields, which you will map to the columns of a database table.
4. Use the Entity Fields dialog (invoked by clicking **Add**) to add persistence fields to the Address class:

```
private Long id;
private String city;
private String country;
private String stateOrProvince;
private String postalCode;
private String street;
```

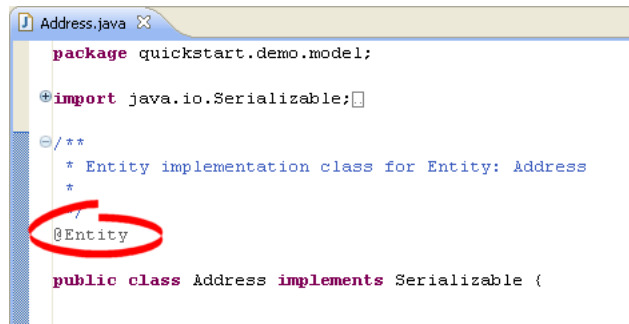
Note: You will also need to add the following columns to the ADDRESS database table:

```
NUMBER(10,0) ADDRESS_ID (primary key)
VARCHAR2(80) PROVINCE
VARCHAR2(80) COUNTRY
VARCHAR2(20) P_CODE
VARCHAR2(80) STREET
VARCHAR2(80) CITY
```

5. Click **Finish**. With the Create JPA Entity wizard completed, Eclipse displays the **Address** entity in the JPA Structure view.

Address.java includes the @Entity annotation, the persistence fields, as well as getter and setter methods for each of the fields.

Figure 1–2 Address Entity in Address.java

A screenshot of the Eclipse IDE showing the source code for Address.java. The code is as follows:

```
package quickstart.demo.model;

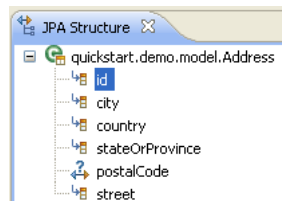
import java.io.Serializable;

/**
 * Entity implementation class for Entity: Address
 */
@Entity
public class Address implements Serializable {
```

The @Entity annotation is circled in red.

Eclipse also displays the **Address** entity in the JPA Structure view.

Figure 1–3 Address Entity in the JPA Structure View



After creating the entity, you must associate it with a database table.

1. Select the **Address** class in the Project Explorer view.
2. In the JPA Details view, notice that Dali has automatically associated the ADDRESS database table with the entity because they are named identically.

Figure 1–4 JPA Details View for Address Entity

Note: Depending on your database connection type, you may need to specify the **Schema**.

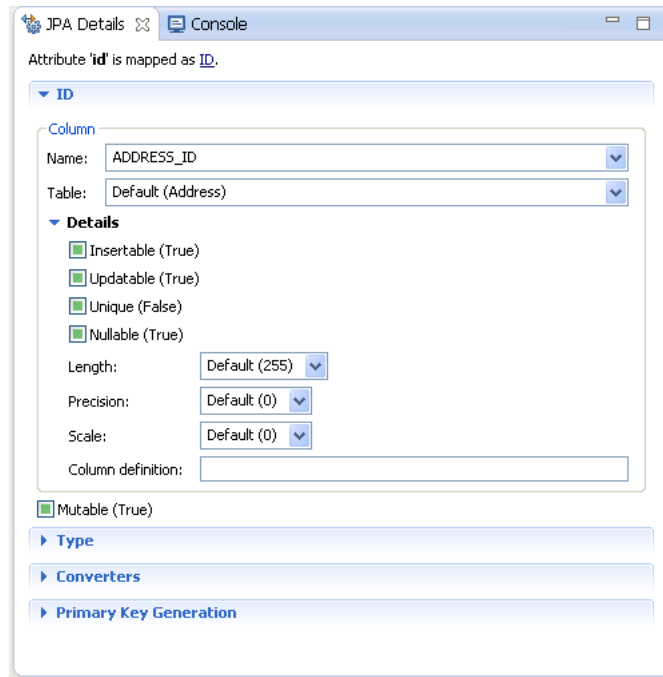
Tip: After associating the entity with the database table, you should update the `persistence.xml` file to include this JPA entity.

Right-click the `persistence.xml` file in the Project Explorer and select **JPA Tools > Synchronize Class List**. Dali adds the following to the `persistence.xml` file:

```
<class>quickstart.demo.model.Address</class>
```

Now we are ready to map each field in the `Address` class to a column in the database table.

1. Select the **id** field in the JPA Details view.
2. Right-click `id` and then select **Map As > id**.
3. In the JPA Details view, select **ADDRESS_ID** in the Name field:

Figure 1–5 JPA Details View for the addressId Field

Eclipse adds the following annotations to the Address entity:

```
@Id
@Column(name="ADDRESS_ID")
```

4. Map each of the following fields (as **Basic** mappings) to the appropriate database column:

Field	Map As	Database Column
city	Basic	CITY
country	Basic	COUNTRY
postalCode	Basic	P_CODE
provinceOrState	Basic	PROVINCE
street	Basic	STREET

Dali automatically maps some fields to the correct database column (such as the `city` field to the `City` column) if the names are identical.

This section contains an overview of concepts you should be familiar with when using Dali to create mappings for Java persistent entities.

- [Understanding Java persistence](#)
- [Understanding OR mappings](#)
- [Understanding Java Persistence API](#)

In addition to these sections, you should review the following resources for additional information:

- Eclipse Dali project: <http://www.eclipse.org/webtools/dali>
- Eclipse Web Tools Platform project: <http://www.eclipse.org/webtools>
- JSR 220 EJB 3.0 specification: <http://www.jcp.org/en/jsr/detail?id=220>
- EclipseLink project: <http://www.eclipse.org/eclipselink>

Understanding Java persistence

Persistence refers to the ability to store objects in a database and use those objects with transactional integrity. In a J2EE application, data is typically stored and persisted in the data tier, in a relational database.

Entity beans are enterprise beans that contain persistent data and that can be saved in various persistent data stores. The entity beans represent data from a database; each entity bean carries its own identity. Entity beans can be deployed using *application-managed persistence* or *container-managed persistence*.

Understanding OR mappings

The Dali OR (object-relational) Mapping Tool allows you to describe how your entity objects *map* to the data source (or other objects). This approach isolates persistence information from the object model—developers are free to design their ideal object model, and DBAs are free to design their ideal schema.

These mappings transform an object data member type to a corresponding relational database data source representation. These OR mappings can also transform object data members that reference other domain objects stored in other tables in the database and are related through foreign keys.

You can use these mappings to map simple data types including primitives (such as `int`), JDK classes (such as `String`), and large object (LOB) values. You can also use them to transform object data members that reference other domain objects by way of

association where data source representations require object identity maintenance (such as sequencing and back references) and possess various types of multiplicity and navigability. The appropriate mapping class is chosen primarily by the cardinality of the relationship.

Understanding Java Persistence API

The Java Persistence API (JPA) part of the Java EE EJB 3.0 specification, simplifies Java persistence. It provides an object-relational mapping approach that lets you declaratively define how to map Java objects to relational database tables in a standard, portable way. JPA works both inside a Java EE application server and outside an EJB container in a Java Standard Edition (Java SE) application. An application written according to the JPA specification is scalable, transactional, and secure.

The persistence.xml file

The JPA specification requires the use of a `persistence.xml` file for deployment. This file defines the database and entity manager options, and may contain more than one persistence unit.

To enable you to easily edit this information, Dali provides the [persistence.xml Editor](#). Alternatively, you can use the Eclipse XML Editor to create and maintain this information. See "[Managing the persistence.xml file](#)" on page 3-20 for more information.

The orm.xml file

Although the JPA specification emphasizes the use of annotations to specify persistence, you can also use the `orm.xml` file to store this metadata. Dali enables you to create a stub `orm.xml` file for a JPA project using the [Create ORM Mapping File wizard](#). See "[Managing the orm.xml file](#)" on page 3-28 for more information.

Note: The metadata must match the XSD specification of your selected JPA implementation.

Dali provides comprehensive support for configuring XML mapping files through the [JPA Details view \(for orm.xml\)](#) that is nearly identical to the annotation-based configuration in the Java source. Alternatively, you can also use the Eclipse XML Editor to create and maintain the metadata information in `orm.xml`.

Understanding Java Architecture for XML Binding

JAXB (Java Architecture for XML Binding – JSR 222) is the standard for XML Binding in Java. JAXB covers 100% of XML Schema concepts and EclipseLink provides a JAXB implementation with many extensions. See <http://jcp.org/en/jsr/detail?id=222> for complete information on the JAXB specification.

Although XML is a common format for the exchange of data, for many applications *objects* are the preferred programmatic representation – not XML. In order to work at

the object-level, the XML data needs to be converted to object form. The mismatch between XML and objects is known as *object-xml impedance mismatch*.

JAXB allows you to interact with XML data by using domain-like objects. Unlike DOM objects, the JAXB content model provides insight into the XML document based on the XML schema. For example, if the XML schema defines XML documents that contain customer information, your content model will contain objects such as **Customer**, **Address**, and **PhoneNumber**. Each *type* in the XML schema will have a corresponding Java class.

This section includes detailed step-by-step procedures for accessing the Dali OR mapping tool functionality.

- [Creating a new JPA project](#)
- [Creating a new JAXB project](#)
- [Creating Database Web Services from Builder XML](#)
- [Converting a Java project to a JPA project](#)
- [Creating a JPA entity](#)
- [Adding persistence to a class](#)
- [Adding virtual attributes](#)
- [Managing the persistence.xml file](#)
- [Synchronizing classes](#)
- [Managing the orm.xml file](#)
- [Specifying additional tables](#)
- [Specifying entity inheritance](#)
- [Creating queries](#)
- [Mapping an entity](#)
- [Generating entities from tables](#)
- [Generating tables from entities](#)
- [Generating dynamic entities from tables](#)
- [Modifying persistent project properties](#)
- [Converting JPA metadata to XML](#)
- [Validating mappings and reporting problems](#)

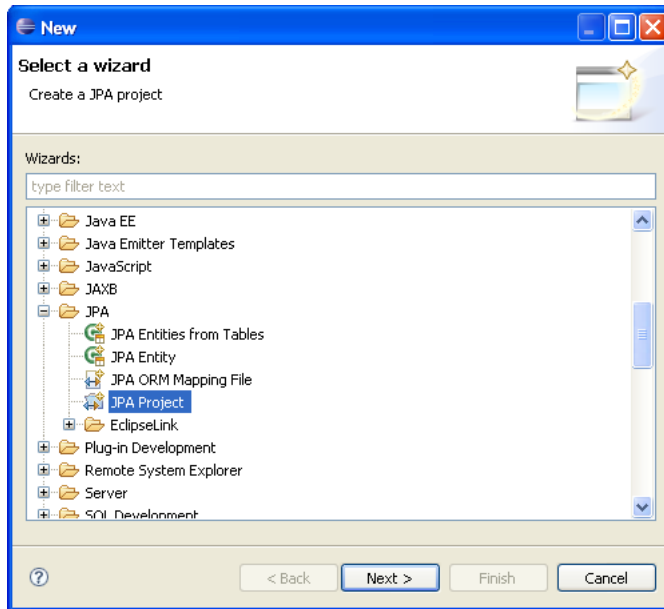
Creating a new JPA project

Use this procedure to create a new JPA project.

1. From the Navigator or Project Explorer, select **File > New > Project**. The Select a wizard dialog appears.

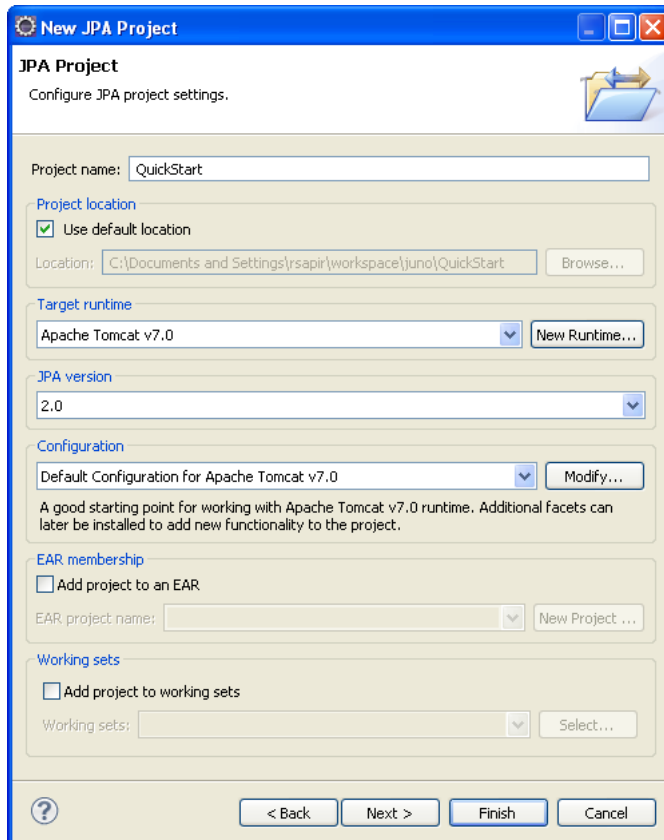
Tip: You can also select the JPA perspective and then select **File > New > JPA Project**.

Figure 3–1 *Selecting the Create a JPA Project wizard*



2. Select **JPA Project** and then click **Next**. The [New JPA Project page](#) appears.

Figure 3–2 *The JPA Project Page*

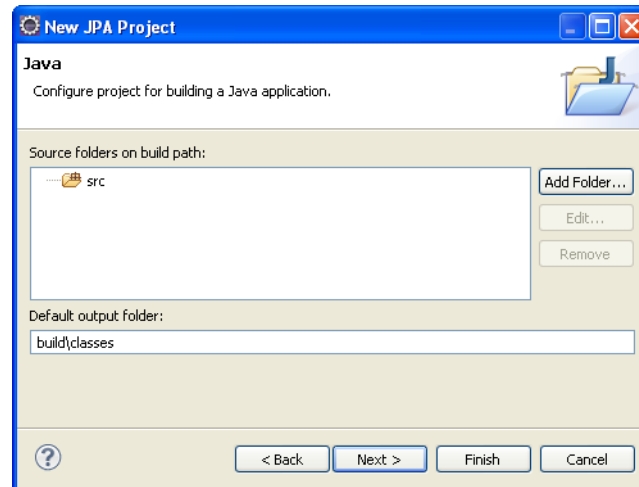


3. Complete the fields on the [New JPA Project page](#) to specify the project name and location, target runtime, and pre-defined configuration.

Note: The **Target runtime** is not required for Java SE development.

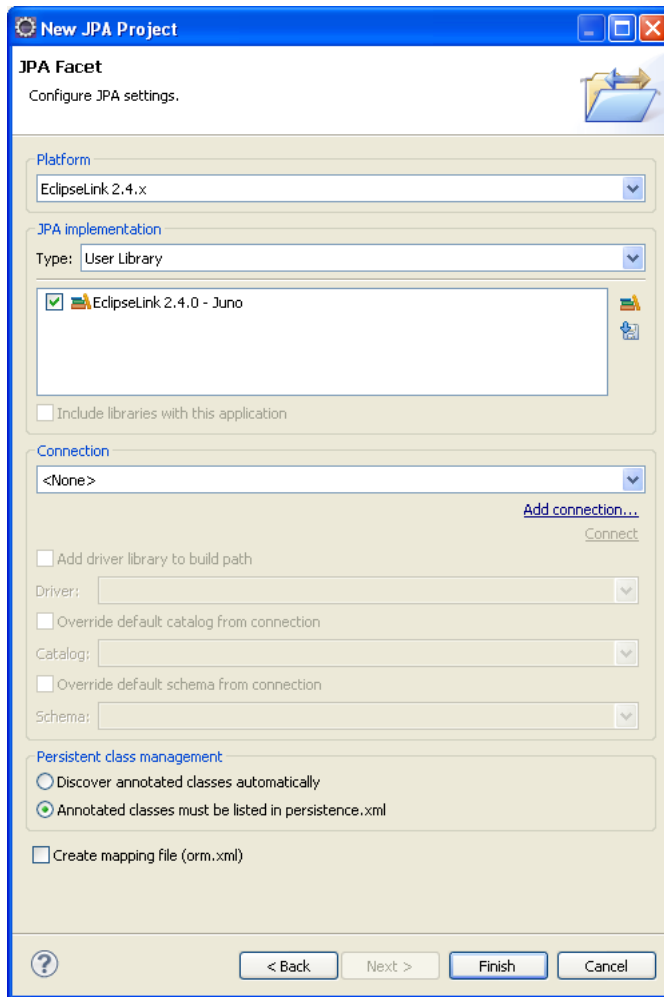
4. Click **Next**. The Java source page appears.

Figure 3–3 *The Java Source Page*



5. Click **Add Folder** to add existing Java source files to the project.
6. Click **Next**. [JPA Facet page](#) appears.

Figure 3–4 The JPA Facet Page



7. Complete the fields on the [JPA Facet page](#) to specify your vendor-specific platform, JPA implementation library, and database connection.

Click **Manage libraries** to create or update your JPA user libraries. Click **Download libraries** to obtain additional JPA implementation libraries.

If Dali derives the incorrect schema, select **Override the Default Schema for Connection**. Using this option, you can select a development time schema for defaults and validation.

If you clear the **Create mapping file (orm.xml)** option (which is selected by default), you can later add a mapping file to the project using the [Create ORM Mapping File wizard](#).

Note: If the server runtime does not provide a JPA implementation, you must explicitly select a JPA implementation library.

To insure the portability of your application, you must explicitly list the managed persistence classes that are included in the persistence unit. If the server supports EJB 3.0, the persistent classes will be discovered automatically.

Depending on your JPA implementation (for example, Generic or EclipseLink), different options may be available when creating JPA projects.

8. Click **Finish**. Dali creates the new JPA project.

You should now open the [JPA Development perspective](#).

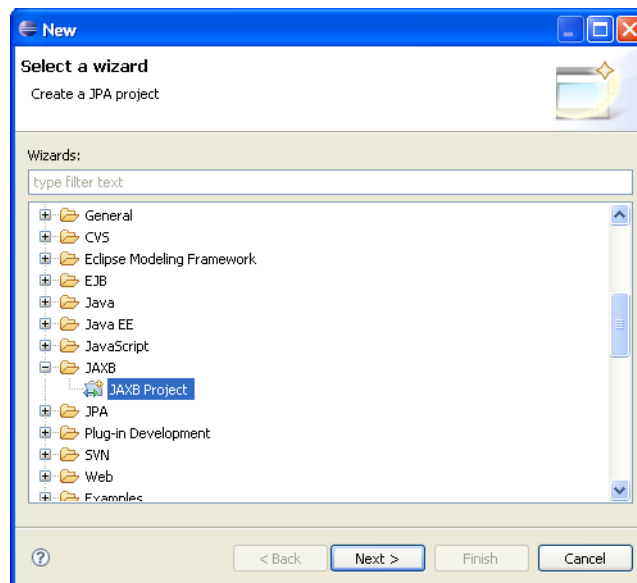
Creating a new JAXB project

Use this procedure to create a new JPA project.

1. From the Navigator or Project Explorer, select **File > New > Project**. The Select a wizard dialog appears.

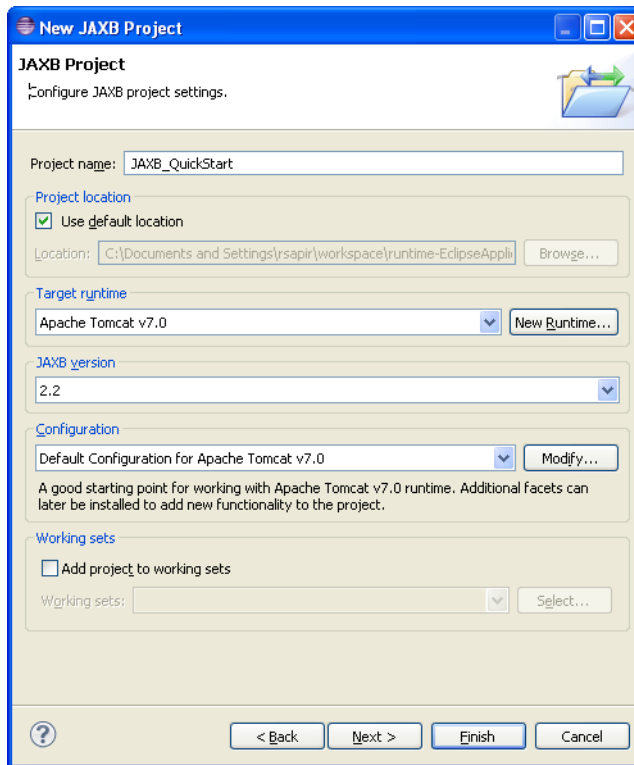
Tip: You can also select the JPA perspective and then select **File > New > JAXB Project**.

Figure 3–5 Selecting the Create a JAXB Project wizard



2. Select **JAXB Project** and then click **Next**. The [New JAXB Project page](#) appears.

Figure 3–6 The JAXB Project Page

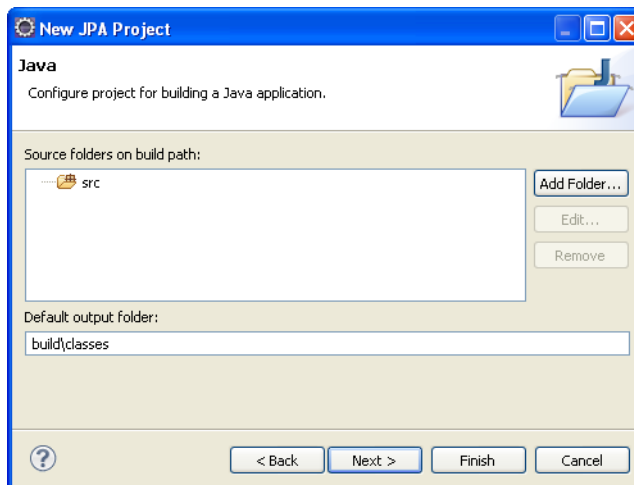


3. Complete the fields on the [New JAXB Project page](#) to specify the project name and location, target runtime, and pre-defined configuration.

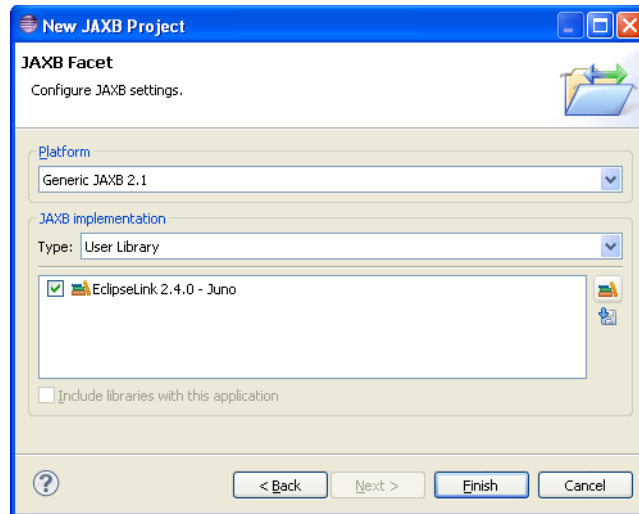
Note: The **Target runtime** is not required for Java SE development.

4. Click **Next**. The Java source page appears.

Figure 3–7 The Java Source Page



5. Click **Add Folder** to add existing Java source files to the project.
6. Click **Next**. [JAXB Facet page](#) appears.

Figure 3–8 The JAXB Facet Page

7. Complete the fields on the [JAXB Facet page](#) to specify your vendor-specific platform, JPA implementation library, and database connection.

Click **Manage libraries** to create or update your JPA user libraries. Click **Download libraries** to obtain additional JPA implementation libraries.

Note: Depending on your JAXB implementation (for example, Generic or EclipseLink), different options may be available when creating JAXB projects.

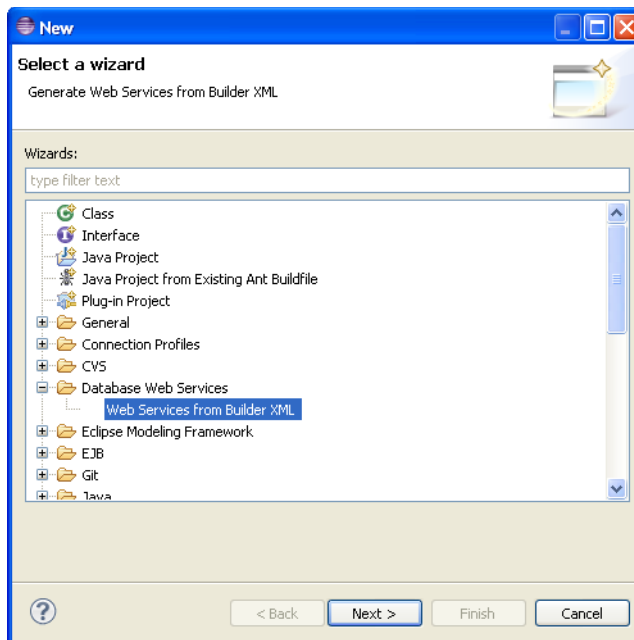
8. Click **Finish**. Dali creates the new JAXB project.
You should now open the [JPA Development perspective](#).

Creating Database Web Services from Builder XML

Use this procedure to create a new JPA project.

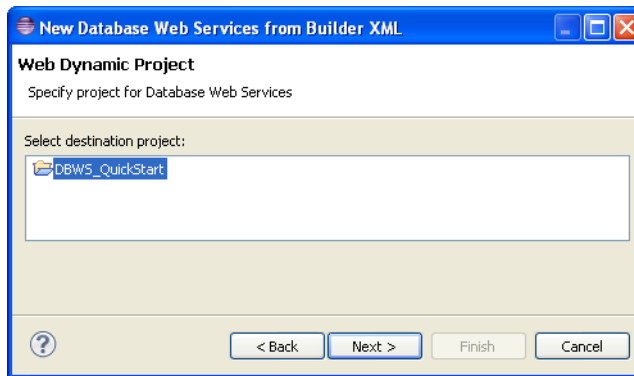
1. From the Navigator or Project Explorer, select **File > New > Project**. The Select a wizard dialog appears.

Figure 3–9 *Selecting the Web Services from Builder XML wizard*

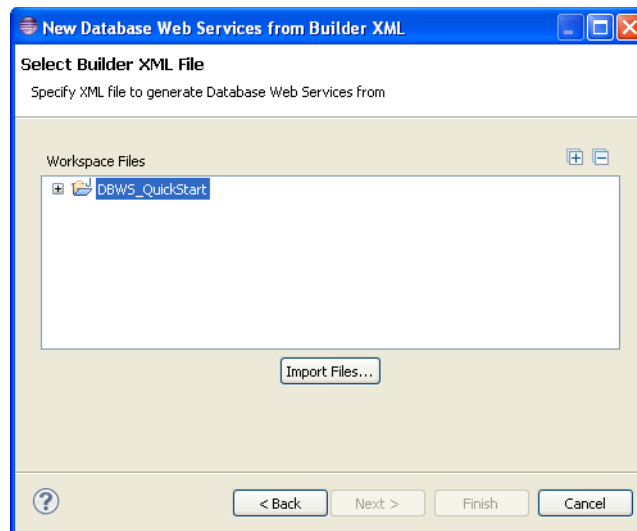


2. Select **Database Web Services > Web Services from Builder XML** and then click **Next**. The [Web Dynamic page](#) appears.

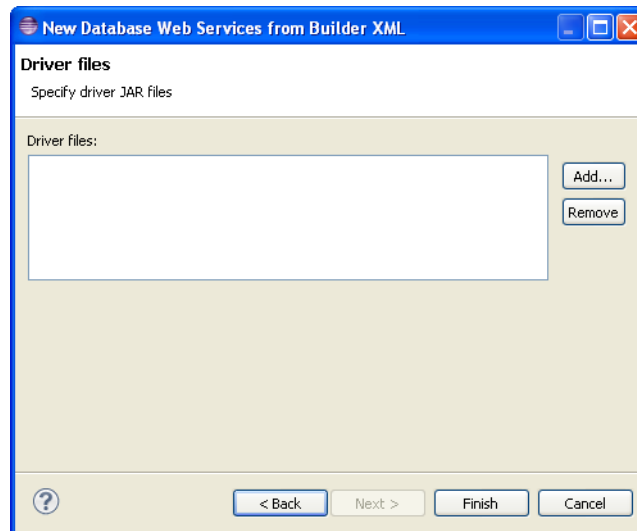
Figure 3–10 *The Web Dynamic Project Page*



3. Select the Dynamic Web Project, and click **Next**. The [Select Builder XML File page](#) appears.

Figure 3–11 The Select Builder XML File Page

4. Select the XML file and click **Next**. The [Driver Files page](#) appears

Figure 3–12 The Driver Files Page

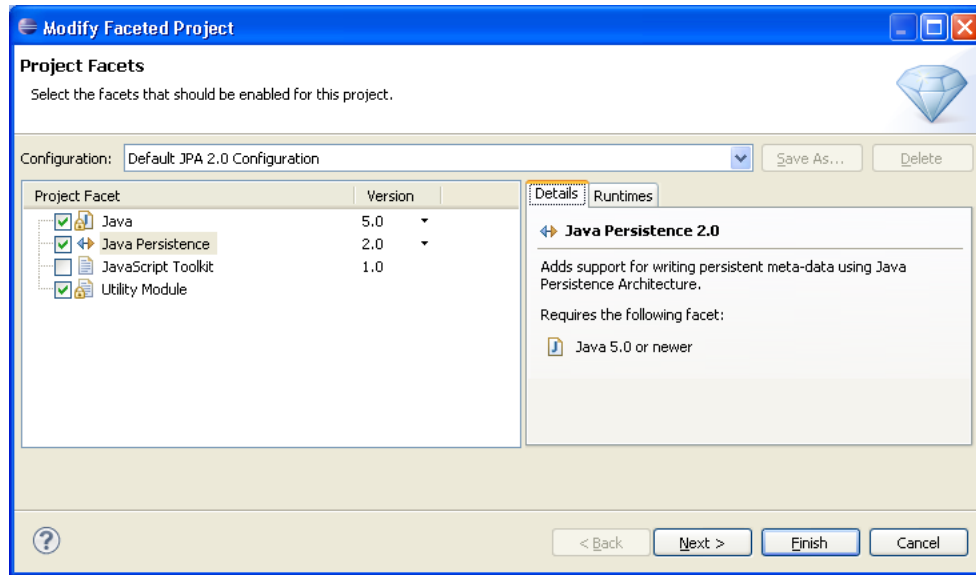
5. Click **Add** to add additional JAR files to the project.
6. Click **Finish**. Dali adds the new JAXB project.
You should now open the [JPA Development perspective](#).

Converting a Java project to a JPA project

Use this procedure to convert an existing Java project to a JPA project.

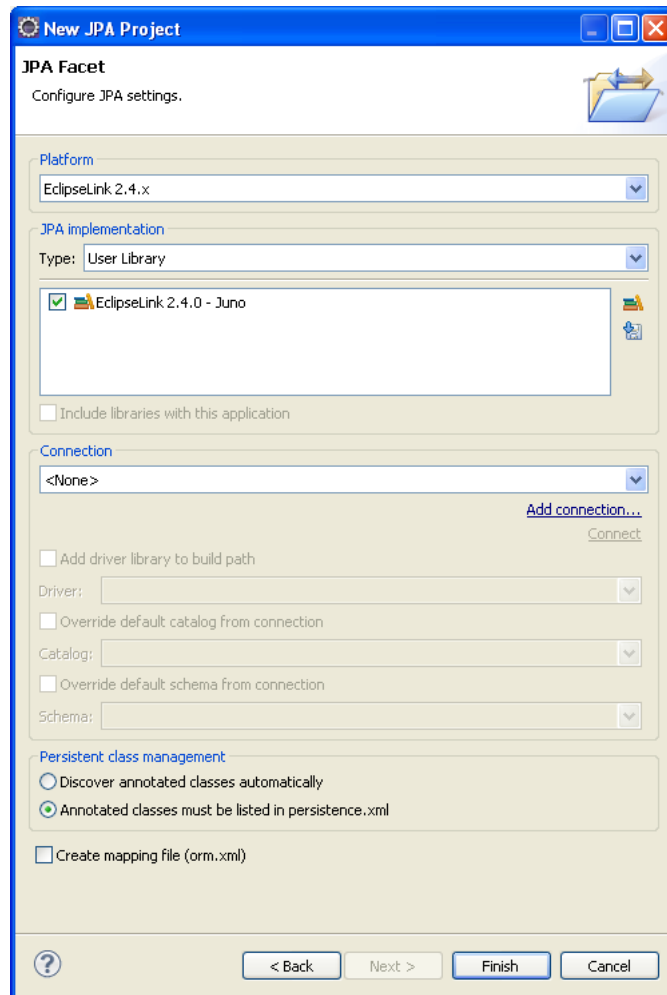
1. From the Navigator or Project explorer, right-click the Java project and then select **Configure > Convert to JPA Project**. The Project Facets page of the Modify Faceted Project wizard appears.

Figure 3–13 Modify Faceted Project Page



2. Change the **Configuration** to **Default JPA Configuration**.
3. Click **Next**. The Java source page appears (see [Figure 3–3](#)).
4. Click **Add Folder** to add existing Java source files to the project and click **Next**. The **JPA Facet page** appears.

Figure 3–14 JPA Facet Page



5. Complete the fields on the [JPA Facet page](#) to specify your vendor-specific platform, JPA implementation library, and database connection.

Click **Manage libraries** to create or update your JPA user libraries. Click **Download libraries** to obtain additional JPA implementation libraries.

If Dali derives the incorrect schema, select **Override the Default Schema for Connection**. Using this option, you can select a development time schema for defaults and validation.

If you clear the **Create mapping file (orm.xml)** option (which is selected by default), you can later add a mapping file to the project using the [Create ORM Mapping File wizard](#).

6. Click **Finish**.

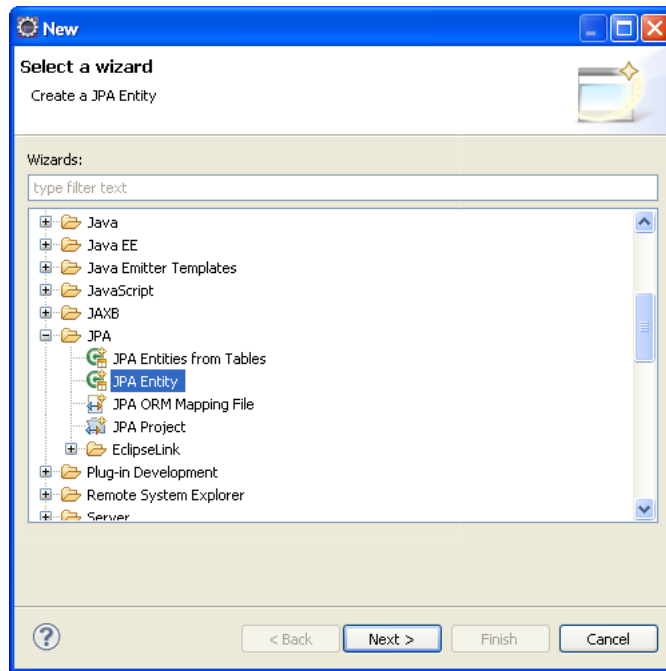
The Dali OR Mapping Tool adds the JPA implementation libraries to your project and creates the necessary `orm.xml` and `perisistence.xml` files.

Creating a JPA entity

Use this procedure to create a JPA entity with the [Create JPA Entity wizard](#):

1. From the Navigator or Project Explorer, select the JPA project and then **File > New > Other**. The Select a Wizard dialog appears.

Figure 3–15 *Selecting the Create a JPA Entity Wizard*



Tip: You can also select the JPA perspective and then select **File > New > JPA Entity**.

2. Select **JPA > JPA Entity** and then click **Next**. The [Entity Class page](#) appears.

Figure 3–16 The Entity Class Page

The screenshot shows the 'New JPA Entity' dialog box. The title bar reads 'New JPA Entity'. The main title is 'Entity class' with a subtitle 'Specify package, class name, and inheritance properties.' The dialog contains the following fields and options:

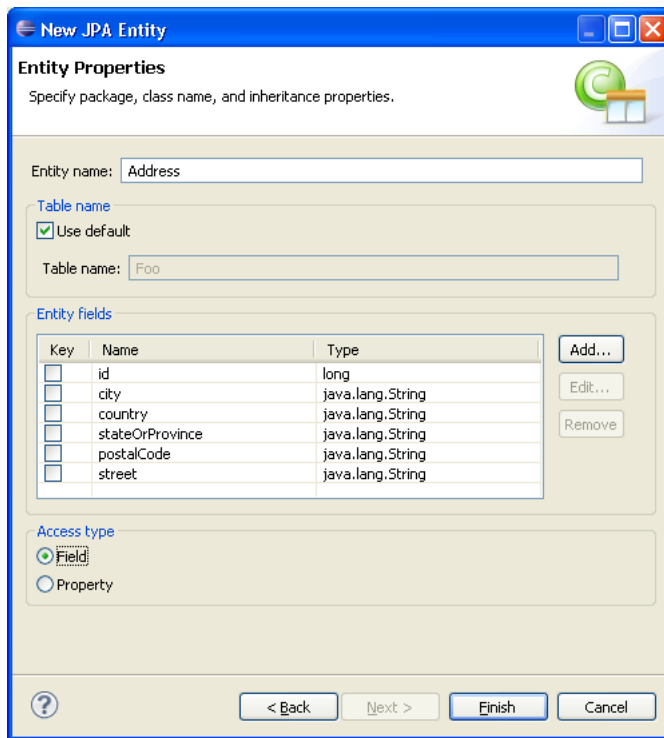
- Project:** A dropdown menu showing 'QuickStart'.
- Source folder:** A text field containing '|QuickStart|src' and a 'Browse...' button.
- Java package:** An empty text field and a 'Browse...' button.
- Class name:** An empty text field.
- Superclass:** An empty text field and a 'Browse...' button.
- Inheritance:** A section with two radio buttons: 'Entity' (selected) and 'Mapped superclass'. Below them is a checkbox labeled 'Inheritance:' followed by a dropdown menu.
- XML entity mappings:** A section with a checkbox labeled 'Add to entity mappings in XML' and a 'Mapping file:' text field with a 'Browse...' button.

At the bottom of the dialog are four buttons: a help icon (?), '< Back', 'Next >', 'Finish', and 'Cancel'.

Complete the fields on the [Entity Class page](#) as follows:

- Select the JPA project in the **Project** field.
 - In the **Source Folder** field, select, or enter, the location of the JPA project's `src` folder.
 - Select, or enter, the name of the class package for this entity in the **Java Package** field.
 - Enter the name of the Java class in the **Class name** field.
 - If needed, enter, or select a superclass.
 - If needed, complete the Inheritance section as follows (these properties are optional):
 - Accept the **Entity** option (the default) to create a Java class with the `@Entity` option.
 - Alternatively, select **Mapped superclass** (if you defined a super class).
 - Select **Inheritance** and then select one of the JSR 220 inheritance mapping strategies (`SINGLE_TABLE`, `TABLE_PER_CLASS`, `JOINED`).
 - Select **Add to entity mappings in XML** to create XML mappings in `orm.xml`, rather than annotations.
3. Click **Next** to proceed to the [Entity Properties page](#) where you define the persistent fields for the entity.

Figure 3–17 The Entity Properties Page



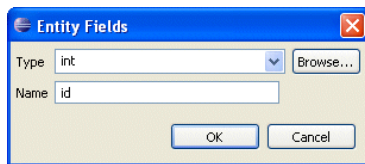
Alternatively, click **Finish** to complete the entity.

4. Complete the page as follows:
 1. If needed, enter a new name for the entity. Doing so results in adding a name attribute to the @Entity notation (@Entity(name="EntityName")).
 2. Accept **Use default** (the default setting) to use the default value for the name of the mapped table. Entering a different name results in adding the @Table notation with its name attribute defined as the new table (@Table(name="TableName")).

Note: The Entity Name-related options are not available if you selected [Mapped superclass](#) on the [Entity Class page](#)

3. Add persistence fields to the entity by clicking **Add**. The Entity Fields dialog appears.

Figure 3–18 The Entity Fields Dialog



4. Select a persistence type from the **Type** list. You can retrieve additional types using the **Browse** function.
5. Enter the field name and then click **OK**. Repeat this procedure for each field.

6. If needed, select **Key** to designate the field as a primary key.
7. Select either the **Field-based** access type (the default) or **Property-based** access type.
5. Click **Finish**. Eclipse adds the entity to your project.

Adding persistence to a class

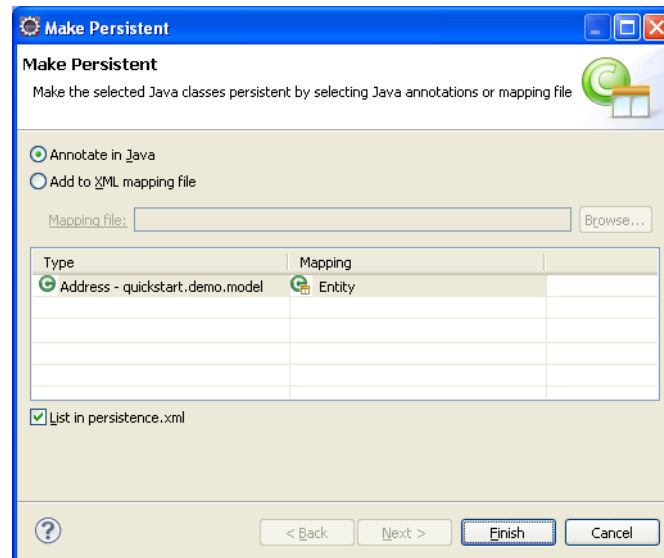
You can make a Java class into one of the following persistent types:

- [Entity](#)
- [Embeddable](#)
- [Mapped superclass](#)

To add persistence to an existing Java class:

1. Right-click the class in the Project Explorer and select **JPA Tools > Make Persistent**. The [Make Persistent dialog](#) appears.

Figure 3–19 *Make Persistence Dialog*



2. Complete the fields on the [Make Persistent dialog](#), specify the persistence mapping for each class, and click **Finish**.

Dali adds the necessary annotation or entry in the XML mapping file for the class.

Entity

An **Entity** is a persistent domain object.

An entity *can be*:

- Abstract or concrete classes. Entities may also extend non-entity classes as well as entity classes, and non-entity classes may extend entity classes.

An entity *must have*:

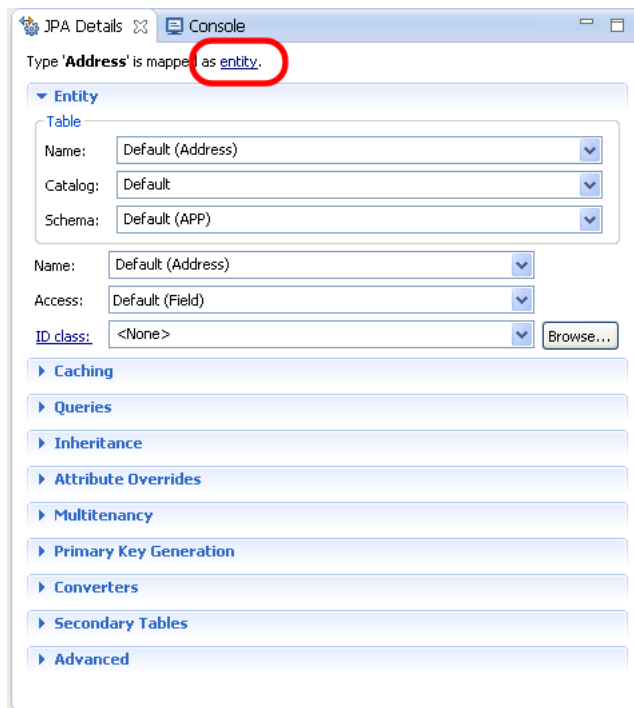
- A no-arg constructor (public or protected); the entity class may have other constructors as well.

Each persistent entity must be mapped to a database table and contain a primary key. Persistent entities are identified by the `@Entity` annotation.

Use this procedure to add persistence to an existing entity:

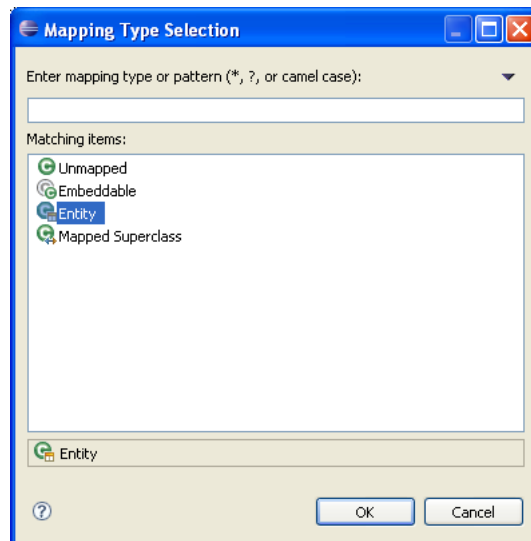
1. Open the Java class in the Project Explorer.
2. Select the class in the JPA Structure view.
3. In the JPA Details view, click the mapping type hyperlink to access the [Mapping Type Selection dialog](#). In the following figure, clicking *entity* invokes the dialog from the JPA Details View.

Figure 3–20 The Mapping Type Hyperlink



Tip: You can also change (or add) persistence for an entity by right-clicking the class in the JPA Structure View and then clicking **Map As > Entity**.

4. Select **Entity** from the [Mapping Type Selection dialog](#) and then click **OK**.

Figure 3–21 The Mapping Type Selection Dialog

5. Complete the remaining [JPA Details view \(for entities\)](#).

Embeddable

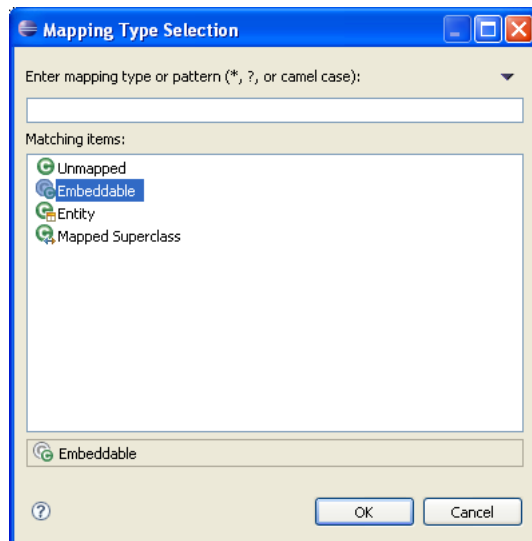
An **Embedded** class is a class whose instances are stored as part of an owning entity; it shares the identity of the owning entity. Each field of the embedded class is mapped to the database table associated with the owning entity.

To override the mapping information for a specific subclass, use the `@AttributeOverride` annotation for that specific class.

An embeddable entity is identified by the `@Embeddable` annotation.

Use this procedure to add embeddable persistence to an existing entity:

1. Open the Java class in the Project Explorer.
2. Select the class in the JPA Structure view.
3. Click the mapping type hyperlink to open the [Mapping Type Selection dialog](#).
4. Select **Embeddable** and then click **OK**.

Figure 3–22 Mapping Type Selection Dialog (Embeddable)

5. Complete the remaining [JPA Details view \(for entities\)](#).

Mapped superclass

An entity that extends a **Mapped Superclass** class inherits the persistent state and mapping information from a superclass. You should use a mapped superclass to define mapping information that is common to multiple entity classes.

A mapped superclass *can be*:

- Abstract or concrete classes

A mapped superclass *cannot be*:

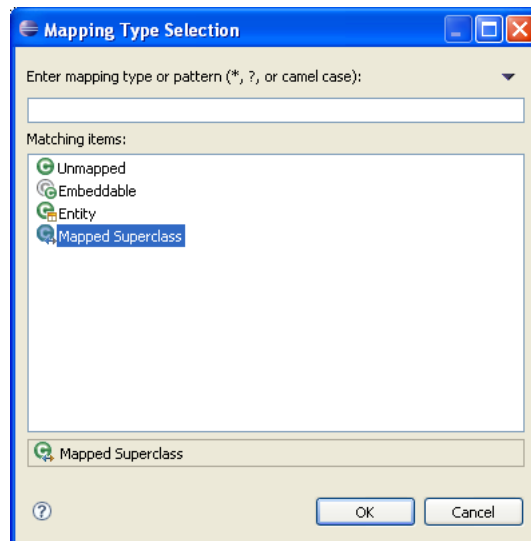
- Be queried or passed as an argument to Entity-Manager or Query operations
- Be the target of a persistent relationship

A mapped superclass does not have a defined database table. Instead, its mapping information is derived from its superclass. To override the mapping information for a specific subclass, use the `@AttributeOverride` annotation for that specific class.

A mapped superclass is identified by the `@MappedSuperclass` annotation.

Use this procedure to add Mapped Superclass persistence to an existing entity:

1. Open the Java class in the Project Explorer.
2. Select the class in the JPA Structure view.
3. In the JPA Details view, click the mapping type hyperlink to open the [Mapping Type Selection dialog](#).
4. Select **Mapped Superclass** and then **OK**.

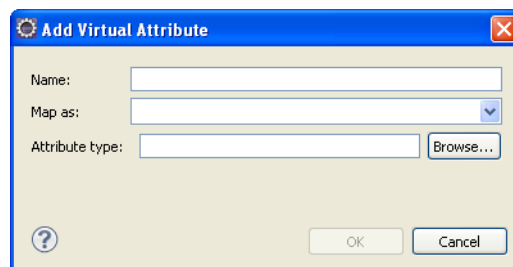
Figure 3–23 Mapping Type Selection Dialog (Mapped Superclass)

5. Complete the remaining [JPA Details view](#) (for entities).

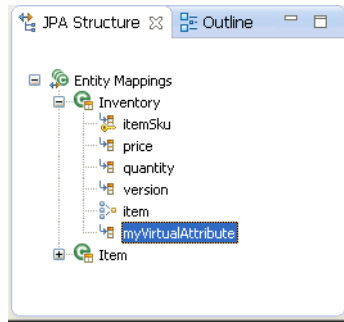
Adding virtual attributes

To add a virtual attribute to an entity:

1. Open the `eclipselink-orm.xml` mapping file.
2. In the [JPA Structure view](#), right-click an entity and select **Add Virtual Attribute**. The Add Virtual Attribute dialog appears.

Figure 3–24 Add Virtual Attribute Dialog

3. Complete the fields on the [Add Virtual Attribute dialog](#) and click **OK**. Dali adds the virtual attribute to the entity.

Figure 3–25 Virtual Attribute

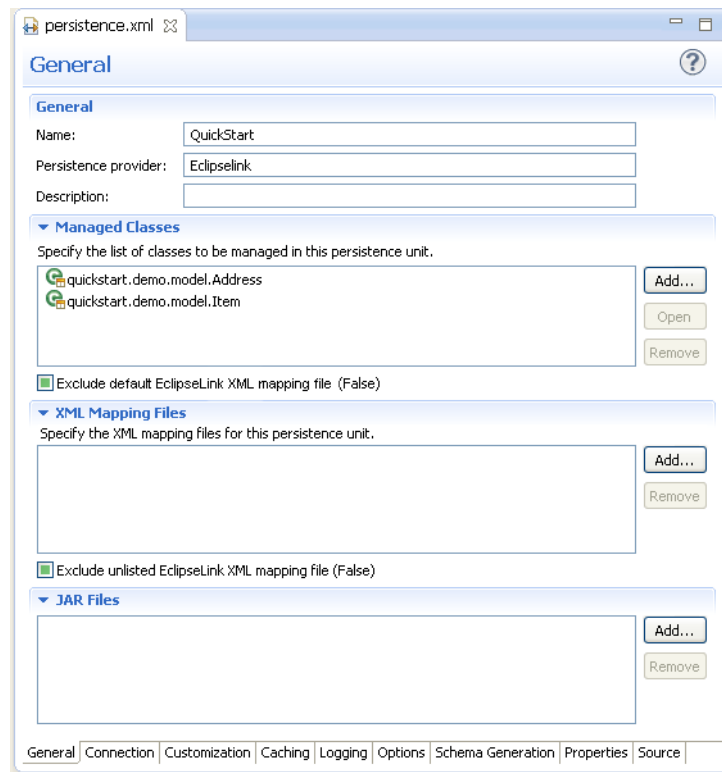
Managing the persistence.xml file

When you create a project, Eclipse creates the META-INF\persistence.xml file in the project's directory.

Example 3–1 Sample persistence.xml File

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
  <persistence-unit name="QuickStart">
    <class>quickstart.demo.model.Address</class>
    <class>quickstart.demo.model.Item</class>
  </persistence-unit>
</persistence>
```

You can manage this file either through the XML editor or through the [persistence.xml Editor](#).

Figure 3–26 The persistence.xml Editor

Note: Depending on your JPA implementation (for example, EclipseLink), the following pages may be available in the persistence.xml Editor:

- [General](#) – Use this page to define the classes, mapping files, and JAR files for the persistence unit.
- [Connection](#) – Use this page to define the datasource (JTA and non-JTA elements) for the project.
- [Customization](#) – Use this page to define change-tracking and session customizer-related properties.
- [Caching](#) – Use this page to define caching properties.
- [Logging](#) – Use this page to define logging properties.
- [Options](#) – Use this page to define session and target database properties.
- [Schema Generation](#) – Use this page to define DDL-related properties.
- [Properties](#) – Use to add or remove vendor-specific properties.
- [Source](#) – Use to view or modify the XML source of the persistence.xml file.

For projects using the EclipseLink JPA implementation, the Connections page also includes JDBC connection pool properties.

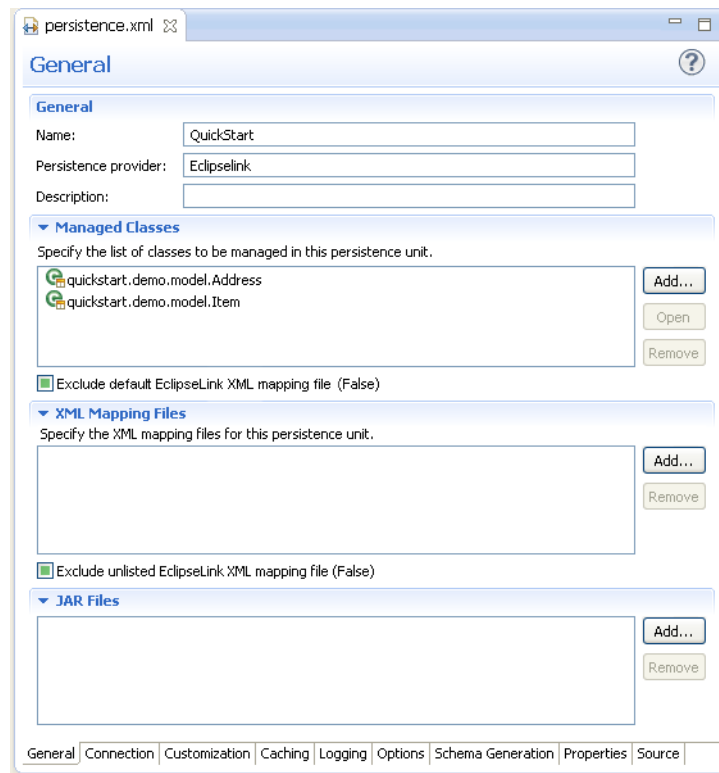
If the project uses the Generic platform, then only the [General](#), [Connection](#), [Properties](#) and [Source](#) pages are available.

To use the persistence.xml Editor:

1. Open the persistence.xml file. The [General](#) page of the editor appears.
2. Use the **General** page to define the persistence.xml files <persistent-unit>-related attributes as well as the <provider>, and <class> elements (described in the following table).

Tip: The persistence.xml Editor's Source page enables you to view and edit the raw XML file.

Figure 3–27 General tab of persistence.xml Editor



3. Complete each field on the [General](#) page.
4. Use the [Connection](#) page to define the `<jta-data-source>` and `<non-jta-data-source>` elements as follows:

To configure the JTA (Java Transaction API) source used by the persistence provider:

1. Select **JTA** from the Transaction Type list.
2. Enter the global JNDI name of the data source.

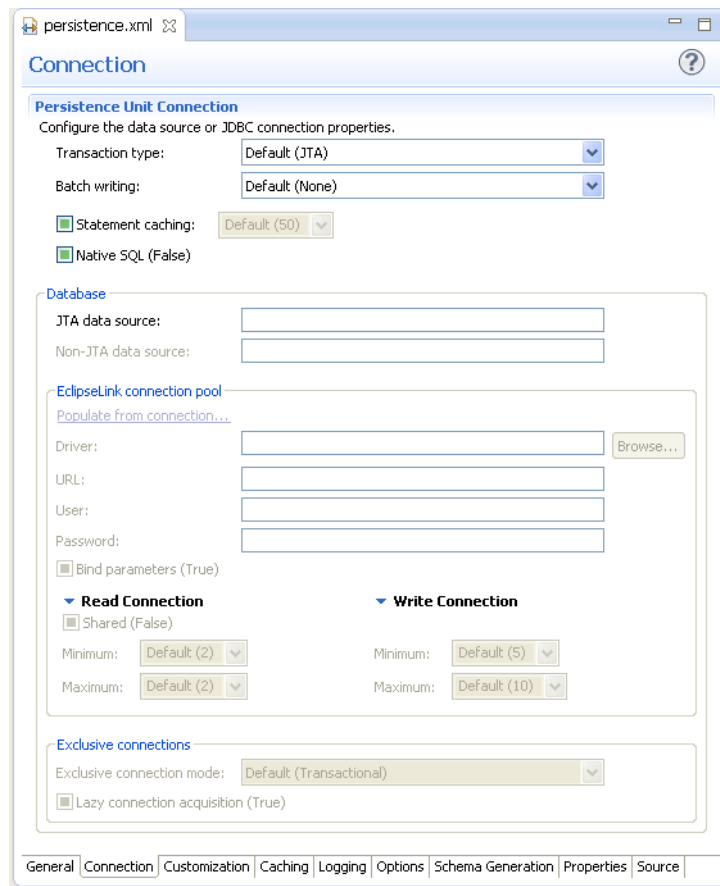
To configure a non-JTA data source:

1. Select **Resource Local** from the Transaction Type list.
2. Enter the global JNDI name of the data source.

Note: Select **Default()** to use the data source provided by the container.

For projects using the Generic platform, you can also define the EclipseLink connection pool driver, connection pool driver, URL, user name and password.

Figure 3–28 Connection tab of persistence.xml Editor



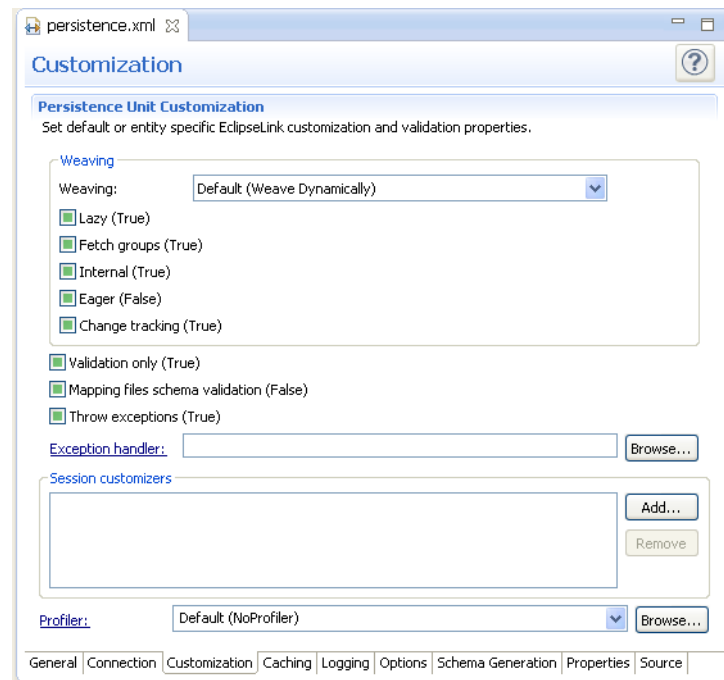
5. Complete each field on the [Connection](#) page.
6. Use the table in the Properties page to set the vendor-specific `<properties>` element.

To add `<property>` elements:

1. Click **Add**.
2. Enter the `<name>` and `<value>` attributes for the `<property>` element using the table's Name and Value fields.

To remove a `<property>` element, select a defined property in the table and then click **Remove**.

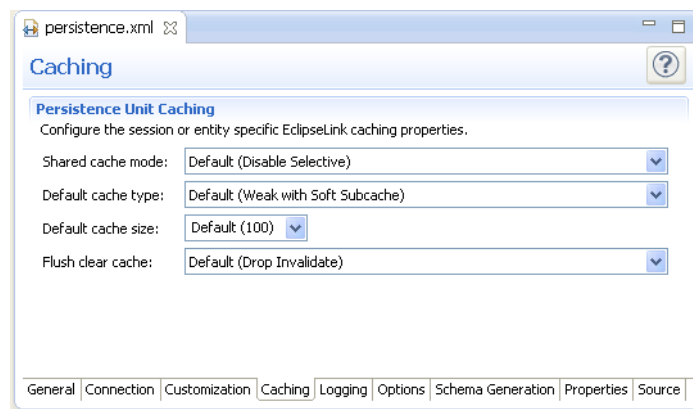
Note: If the project uses the EclipseLink platform, the connection page also includes parameters for JDBC connection pooling.

Figure 3–29 Customization tab of persistence.xml Editor

7. Complete each field on the [Customization](#) page.

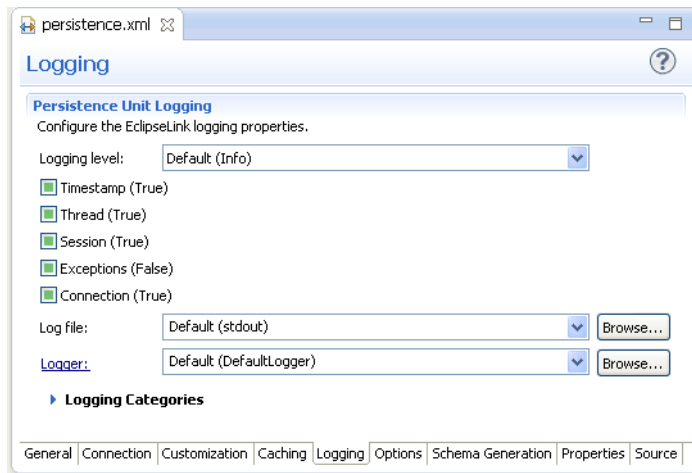
Additional pages may be available for the persistence.xml editor, depending on your JPA provider. See "[persistence.xml Editor](#)" on page 4-37 for more information.

- [Caching](#) page

Figure 3–30 Caching tab of persistence.xml Editor

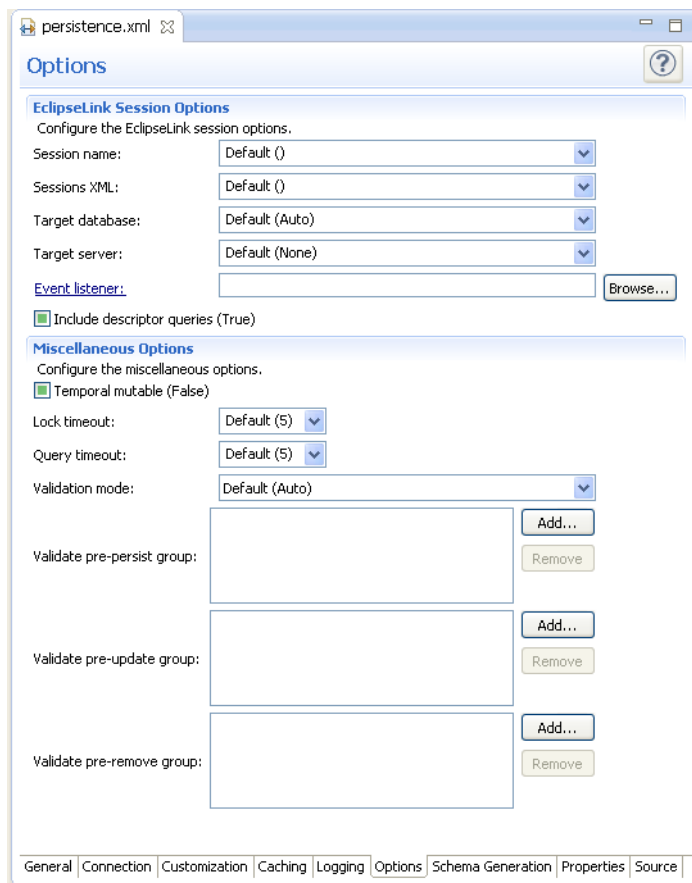
- [Logging](#) page

Figure 3–31 Logging tab of persistence.xml Editor

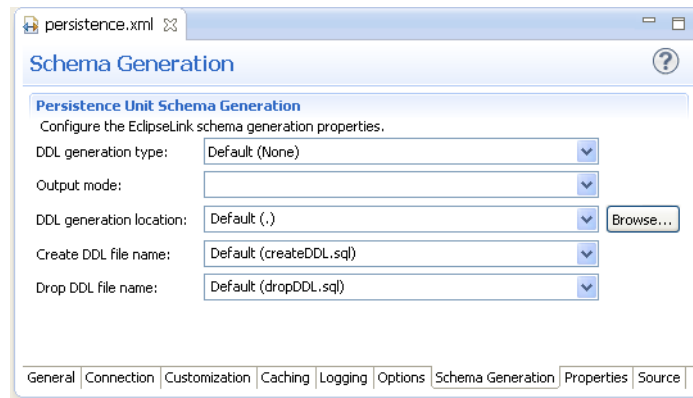


- [Options](#) page

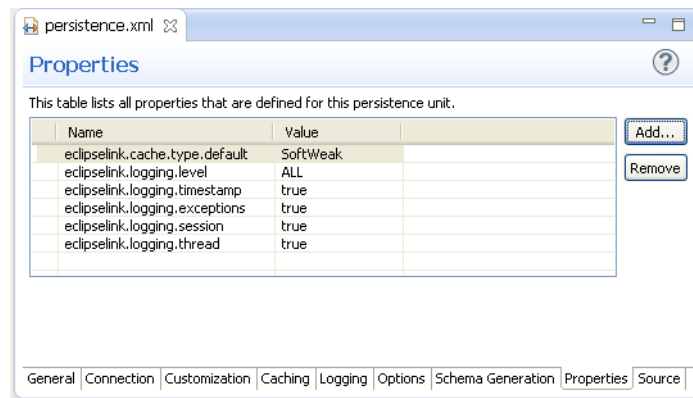
Figure 3–32 Options tab of persistence.xml Editor



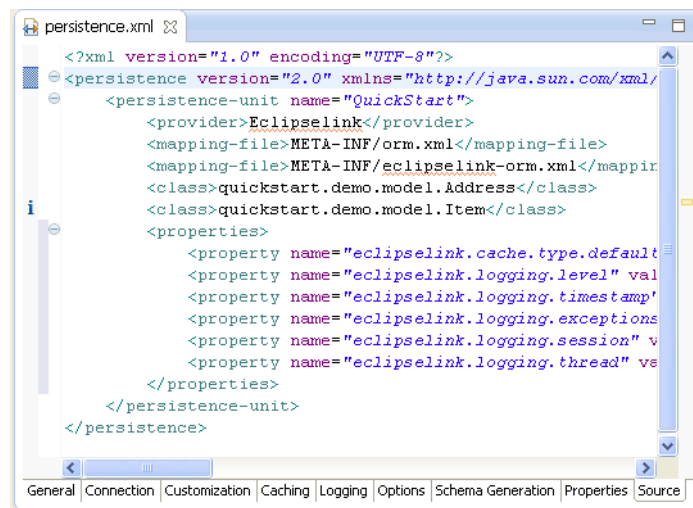
- [Schema Generation](#) page

Figure 3–33 Schema Generation tab of persistence.xml Editor

- [Properties](#) page

Figure 3–34 Properties tab of persistence.xml Editor

- [Source](#) page

Figure 3–35 Source tab of persistence.xml Editor

Synchronizing classes

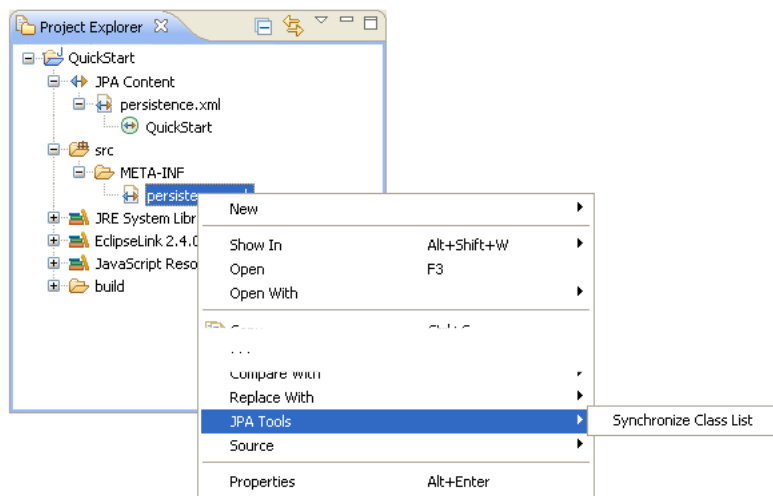
As you work with the classes in your Java project, you will need to update the `persistence.xml` file to reflect the changes.

Use this procedure to synchronize the `persistence.xml` file:

1. Right-click the `persistence.xml` file in the Project Explorer and select **JPA Tools > Synchronize Class List**.

Note: Use this function if you selected **Annotated classes must be listed in the persistence.xml option** in the [JPA Facet page](#). In general, you do not have to use this function within the container.

Figure 3–36 Synchronizing the persistence.xml File



Dali adds the necessary `<class>` elements to the `persistence.xml` file.

2. Use the [persistence.xml Editor](#) to continue editing the `persistence.xml` file.

Managing the orm.xml file

When creating a JPA project, (see "[Creating a new JPA project](#)") you can also create the `orm.xml` file that defines the mapping metadata and defaults.

Eclipse creates the `META-INF\orm.xml` file in your project's directory:

Example 3–2 Sample orm.xml File

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="<PERSISTENCE_VERSION>"
  xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="<PERSISTENCE_UNIT_NAME>">
    <provider="<PERSISTENCE_PROVIDER>" />
  </persistence-unit>
</persistence>
```

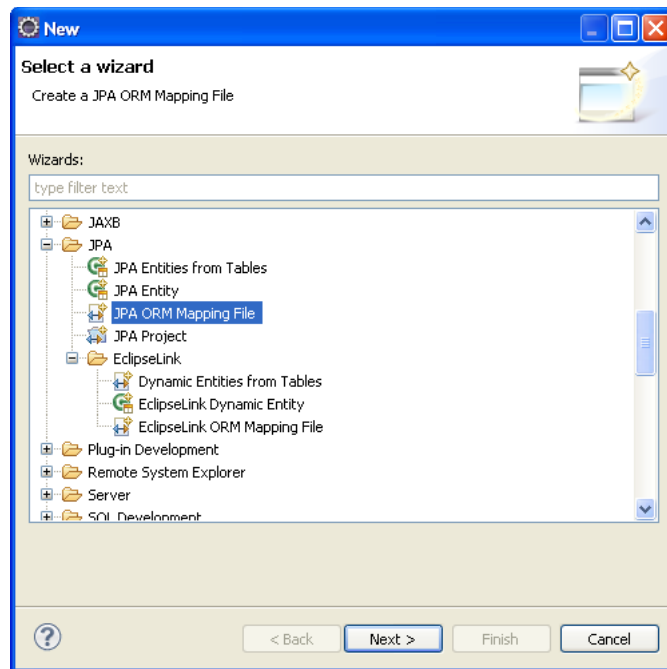
Creating an orm.xml file

If you opt not to create an `orm.xml` file when you create a JPA project, you can create one using the [Create ORM Mapping File wizard](#).

Use this procedure to create an `orm.xml` file:

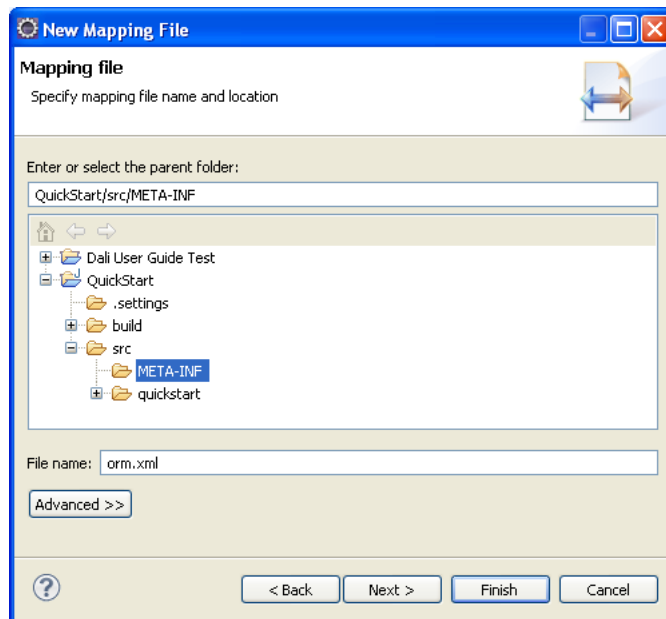
1. From the Navigator or Project Explorer, select **File > New > Other**. The Select a Wizard dialog appears.

Figure 3–37 The Select a Wizard Dialog

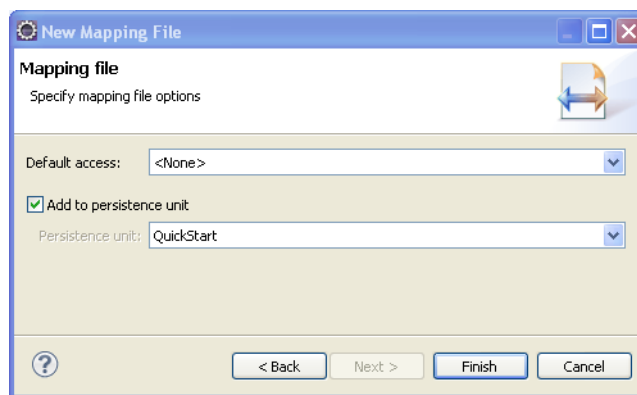


2. Select **JPA ORM Mapping File** and then click **Next**. The Mapping File page of the [Create ORM Mapping File wizard](#) appears.

If you are using EclipseLink, you can select **EclipseLink > EclipseLink ORM Mapping File**.

Figure 3–38 *New Mapping File Location Page*

3. Select the name and location of your mapping file and click **Next**. The [Mapping File Options](#) page appears.

Figure 3–39 *New Mapping File Options Page*

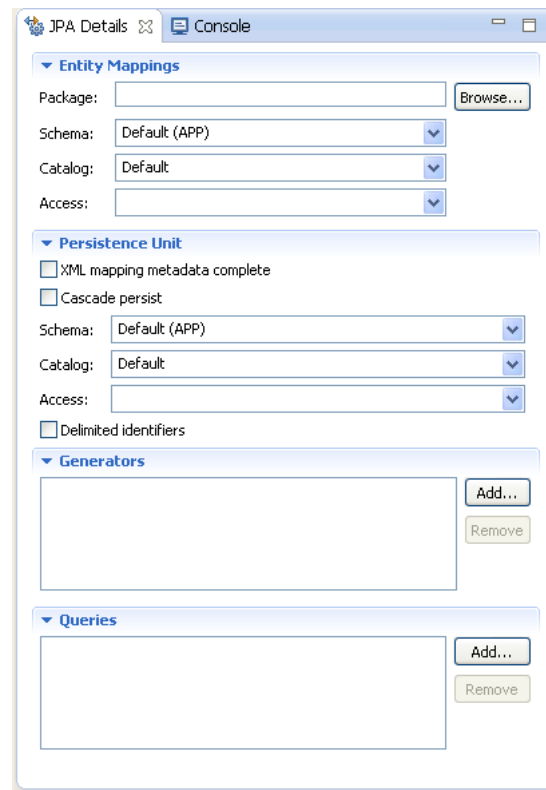
4. Define the properties in the [Mapping File Options](#) page and click **Finish**. The `orm.xml` file appears in the `src` directory of the selected JPA project. You can manage the `orm.xml` file using the JPA Details view or through the XML Editor. See also [JPA Details view \(for orm.xml\)](#).

Working with orm.xml file

You can work with the `orm.xml` by using the JPA Details view.

Use this procedure to work with the `orm.xml` file:

1. Right-click the `orm.xml` file in the Project Explorer and select **Open**.
2. In the JPA Structure view, select **EntityMappings**.
3. Use the JPA Details view to configure the entity mapping and persistence unit defaults.

Figure 3–40 JPA Details view for EntityMappings (orm.xml)

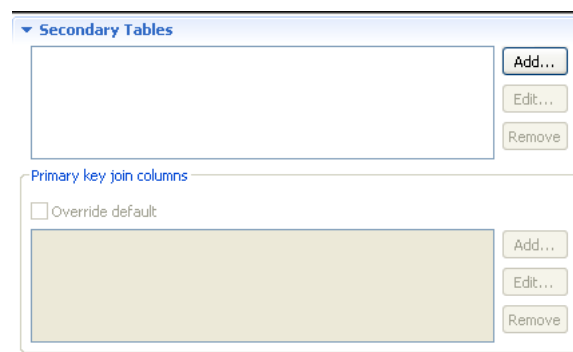
4. Complete the fields on the .

Specifying additional tables

Add a secondary table annotation to an entity if its data is split across more than one table.

To add a secondary table to the entity,

1. Select the entity in the Project Explorer.
2. In the JPA Details view, select the **Secondary Tables** information.

Figure 3–41 Specifying Secondary Tables

3. Click **Add** to associate an additional table with the entity. The Edit Secondary Table dialog appears

4. Select the **Name**, **Catalog**, and **Schema** of the additional table to associate with the entity.

Eclipse adds the following annotations the entity:

```
@SecondaryTable(name="NAME", catalog = "CATALOG", schema = "SCHEMA")
```

To override the default primary key:

1. Enable the **Overwrite default** option, then click **Add** to specify a new primary key join column. The Create New Primary Key Join Column appears.
2. Select the **Name**, **Referenced column name**, **Table**, and **Column definition** of the primary key for the entity.

Eclipse adds the following annotations the entity:

```
@SecondaryTable(name="NAME", catalog = "CATALOG", schema = "SCHEMA",
pkJoinColumns = {@PrimaryKeyJoinColumn(name="id", referencedColumnName =
"id"),@PrimaryKeyJoinColumn(name="NAME", referencedColumnName = "REFERENCED
COLUMN NAME", columnDefinition = "COLUMN DEFINITION")})
```

Specifying entity inheritance

An entity may inherit properties from other entities. You can specify a specific strategy to use for inheritance.

Use this procedure to specify inheritance (@Inheritance) for an existing entity (@Entity):

1. Select the entity in the Project Explorer.
2. In the JPA Details view, select the **Inheritance** information.

Figure 3–42 Specifying Inheritance

The screenshot shows the 'Inheritance' configuration in the Eclipse IDE. It includes a dropdown for 'Strategy' (Default (Single Table)), a dropdown for 'Discriminator value' (Default (Address)), and a section for 'Discriminator column' with dropdowns for 'Name' (Default (DTYPE)) and 'Type' (Default (String)). Below that is the 'Primary key join columns' section, which has an 'Override default' checkbox checked and a list of columns with 'Add...', 'Edit...', and 'Remove' buttons.

3. In the **Strategy** list, select one of the following the inheritance strategies:
 - A single table (default)
 - Joined table
 - One table per class
4. Complete the fields in the **Inheritance** area.

Use the following table to complete the remaining fields on the tab. See ["Inheritance"](#) on page 4-18 for additional details.

Eclipse adds the following annotations the entity field:

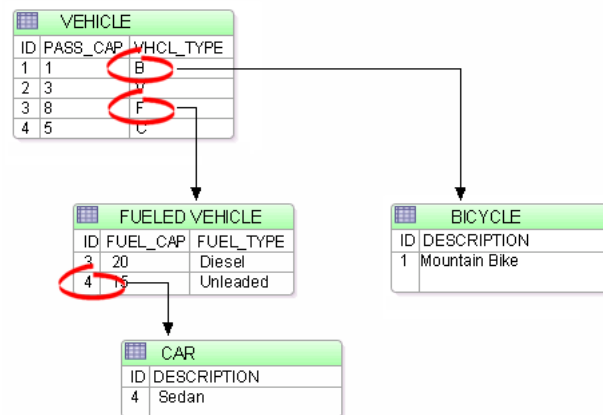
```
@Inheritance(strategy=InheritanceType.<INHERITANCE_STRATEGY>)
@DiscriminatorColumn(name="<DISCRIMINATOR_COLUMN>",
    discriminatorType=<DISCRIMINATOR_TYPE>)
@DiscriminatorValue(value="<DISCRIMINATOR_VALUE>")
@PrimaryKeyJoinColumn(name="<JOIN_COLUMN_NAME>",
    referencedColumnName = "<REFERENCED_COLUMN_NAME>")
```

The following figures illustrates the different inheritance strategies.

Figure 3–43 Single Table Inheritance

VEHICLE						
ID	PASS_CAP	VHCL_TYPE	FUEL_CAP	FUEL_TYPE	CAR_DESC	BICYCLE_DES
1	1	B				Mountain Bike
2	3	V				
3	8	F	20	Diesel		
4	5	C	15	Unleaded	Sedan	

Figure 3–44 Joined Table Inheritance



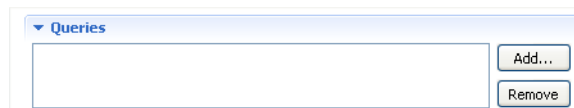
Creating queries

Named queries improve application performance because they are prepared once and they (and all of their associated supporting objects) can be efficiently reused thereafter, making them well suited for complex and frequently executed operations. Named queries use the JPA query language for portable execution on any underlying database; named native queries use the SQL language native to the underlying database.

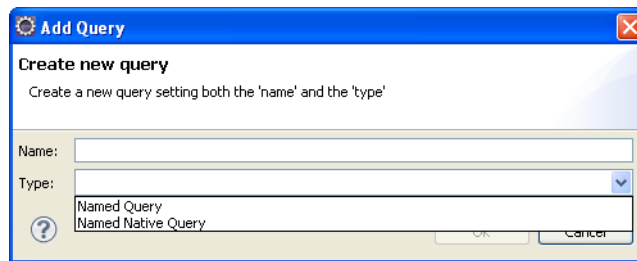
Use this procedure to add `@NamedQuery` and `@NamedNativeQuery` annotations to the entity.

To create a named query:

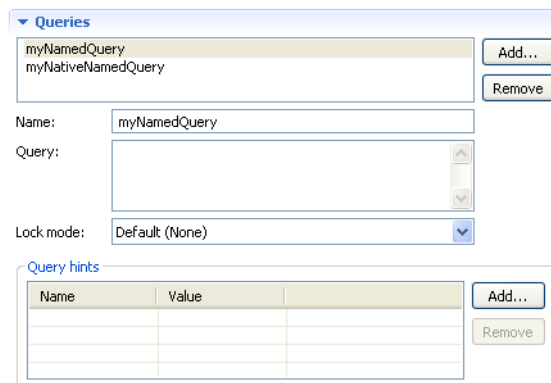
1. Select the entity in the Project Explorer.
2. In the JPA Details view, expand the [Queries](#) area.

Figure 3–45 JPA Details, Queries Tab

3. Click **Add** to add a new query. The **Add Query dialog** appears

Figure 3–46 Add Query Dialog

4. Enter the name of the query, select the query type (**Named Query** or **Named Native Query**), and click **OK**. The **Queries** area expands to show additional fields.

Figure 3–47 JPA Details, Queries Tab

5. Enter the query in the **Query** field.
6. Complete the rest of the field on the **Queries** page.
7. To add a Query hint, click **Add**. Enter the **Name** and **Value** of the hint.

Mapping an entity

Dali supports the following mapping types for Java persistent entities:

- [Basic mapping](#)
- [Element collection mapping](#)
- [Embedded mapping](#)
- [Embedded ID mapping](#)
- [ID mapping](#)
- [Many-to-many mapping](#)

- [Many-to-one mapping](#)
- [One-to-many mapping](#)
- [One-to-one mapping](#)
- [Transient mapping](#)
- [Version mapping](#)

Note: Additional mapping types (such as Basic Collection mappings) may be available when using Dali with EclipseLink.

Basic mapping

Use a **Basic Mapping** to map an attribute directly to a database column. Basic mappings may be used only with the following attribute types:

- Java primitive types and wrappers of the primitive types
- `java.lang.String`, `java.math.BigInteger`
- `java.math.BigDecimal`
- `java.util.Date`
- `java.util.Calendar`, `java.sql.Date`
- `java.sql.Time`
- `java.sql.Timestamp`
- `byte[]`
- `Byte[]`
- `char[]`
- `Character[]`
- `enums`
- any other type that implements `Serializable`

To create a basic mapping:

1. In the [JPA Structure view](#), right-click the field to map. Select **Map As > Basic**. The [JPA Details view \(for attributes\)](#) displays the properties for the selected field.

Figure 3–48 JPA Details, Basic mapping

2. Complete each field in the [Basic Mapping](#) area.
3. Complete the remaining areas in the [JPA Details view \(for attributes\)](#).

Eclipse adds the following annotations to the field:

```
@Column(name="<COLUMN_NAME>", table="<COLUMN_TABLE>",
        insertable=<INSERTABLE>, updatable=<UPDATABLE>)
@Basic(fetch=FetchType.<FETCH_TYPE>, optional = <OPTIONAL>)
@Temporal(TemporalType.<TEMPORAL>)
```

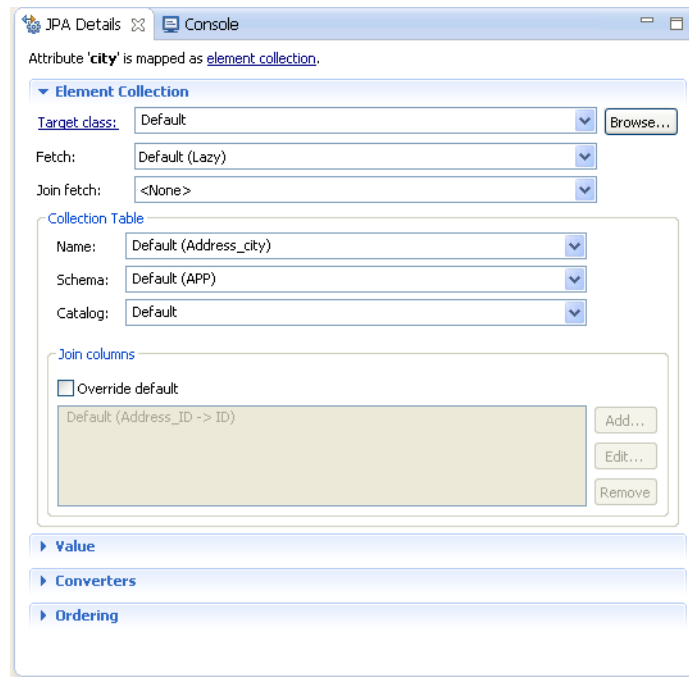
Element collection mapping

Use an **Element Collection** to define a collection of **Basic** objects. The Basic values are stored in a separate collection table. Because the target is a Basic value (instead of an Entity), you can easily define collections of simple values without having to define a class for the value.

To create an element collection mapping:

1. In the [JPA Structure view](#), right-click the field to map. Select **Map As > Element Collection**. The [JPA Details view \(for attributes\)](#) displays the properties for the selected field.

Figure 3–49 JPA Details, Element collection mapping



2. Complete each field in the [Element Collection Mapping](#) area.
3. Complete the remaining areas in the [JPA Details view \(for attributes\)](#).

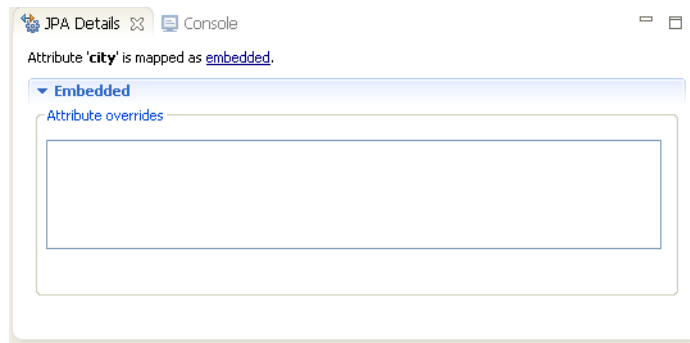
Eclipse adds the following annotations to the field:

```
@ElementCollection
@CollectionTable(
    name="<TABLE_NAME>",
    joinColumns=@JoinColumn(name="<COLUMN_TABLE>")
)
@Column(name="<COLUMN_TABLE>")
```

Embedded mapping

Use an **Embedded Mapping** to specify a persistent field or property of an entity whose value is an instance of an embeddable class.

1. In the [JPA Structure view](#), right-click the field to map.
2. Select **Map as > Embedded**. The [JPA Details view \(for attributes\)](#) displays the properties for the selected field.

Figure 3–50 JPA Details, Embedded mapping

3. Complete each field in the [Embedded Mapping](#) area.

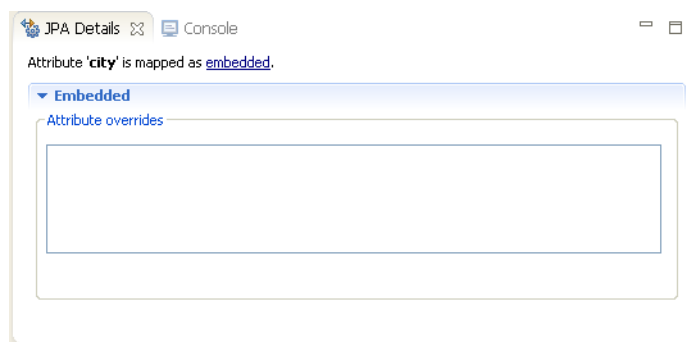
Eclipse adds the following annotations to the field:

```
@Embedded
@AttributeOverride(
    column=@Column(
        table="<COLUMN_TABLE>",
        name = "<COLUMN_NAME>"
    )
)
```

Embedded ID mapping

Use an **Embedded ID Mapping** to specify the primary key of an embedded ID. These mappings may be used with a [Embeddable](#) entities.

1. In the [JPA Structure view](#), select the field to map.
2. Right-click the field and then select **Map As > Embedded Id**. The [JPA Details view \(for attributes\)](#) displays the properties for the selected field.

Figure 3–51 JPA Details, Embedded ID mapping

3. Complete each field in the [Embedded ID Mapping](#) area.

Eclipse adds the following annotations to the field:

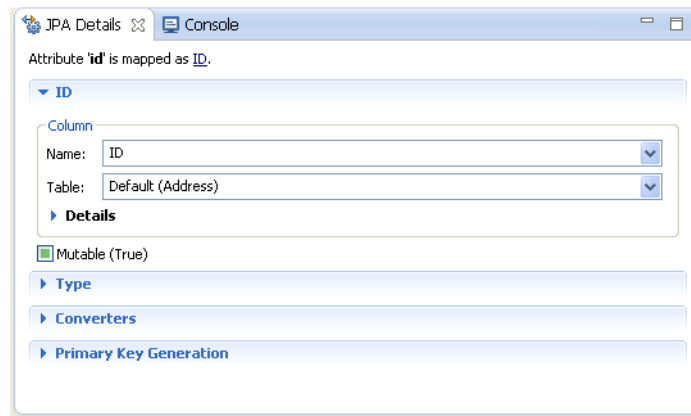
```
@EmbeddedId
```

ID mapping

Use an **ID Mapping** to specify the primary key of an entity. ID mappings may be used with a **Entity** or **Mapped superclass**. Each **Entity** must have an ID mapping.

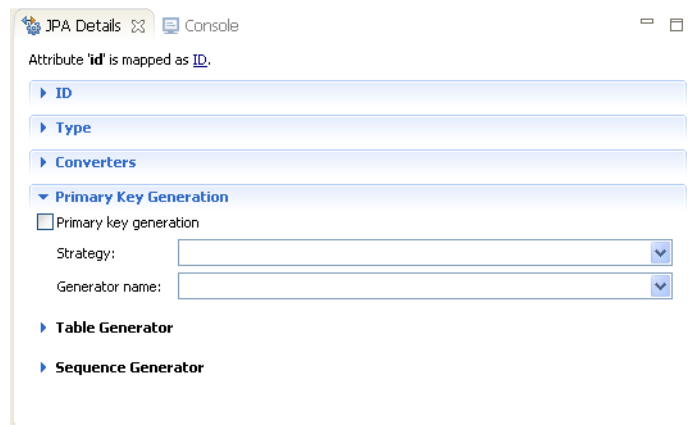
1. In the **JPA Structure view**, select the field to map.
2. Right click the field and then select **Map as > ID**. The **JPA Details view (for attributes)** displays the properties for the selected.

Figure 3–52 JPA Details, ID mapping



3. Complete each field in the **ID Mapping** area.
4. Use the **Primary Key Generation** area to specify the strategy to use for generating primary keys.

Figure 3–53 JPA Details, Primary key generation



5. Complete each field in the **Primary Key Generation information** area.
6. Complete the remaining areas in the **JPA Details view (for attributes)**.

Additional fields will appear in the **Primary Key Generation information** area, depending on the selected Strategy. See "**JPA Details view (for attributes)**" on page 4-22 for additional information.

Eclipse adds the following annotations to the field:

```
@Id
@Column(
```

```

        name="<COLUMN_NAME>",
        table="<TABLE_NAME>",
        insertable=<INSERTABLE>,
        updatable=<UPDATABLE>
    )
    @Temporal (<TEMPORAL>)
    @GeneratedValue (
        strategy=GeneratorType.<STRATEGY>,
        generator="<GENERATOR_NAME>"
    )
    @TableGenerator (
        name="<TABLE_GENERATOR_NAME>",
        table = "<TABLE_GENERATOR_TABLE>",
        pkColumnName = "<TABLE_GENERATOR_PK>",
        valueColumnName = "<TABLE_GENERATOR_VALUE_COLUMN>",
        pkColumnValue = "<TABLE_GENERATOR_PK_COLUMN_VALUE>"
    )
    @SequenceGenerator (name="<SEQUENCE_GENERATOR_NAME>",
        sequenceName="<SEQUENCE_GENERATOR_SEQUENCE>")

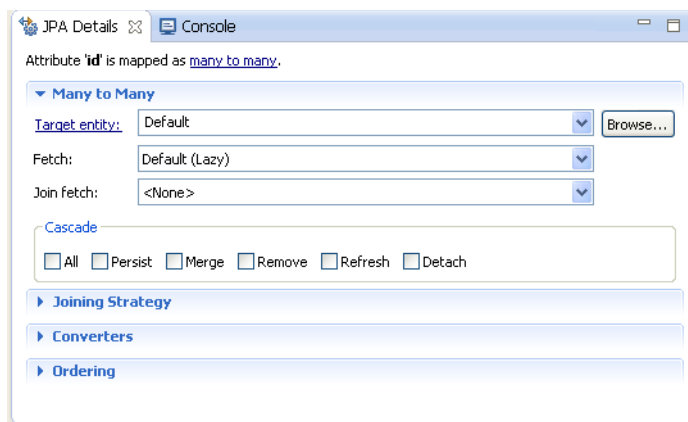
```

Many-to-many mapping

Use a **Many-to-Many Mapping** to define a many-valued association with many-to-many multiplicity. A many-to-many mapping has two sides: the *owning side* and *non-owning side*. You must specify the join table on the owning side. For bidirectional mappings, either side may be the owning side.

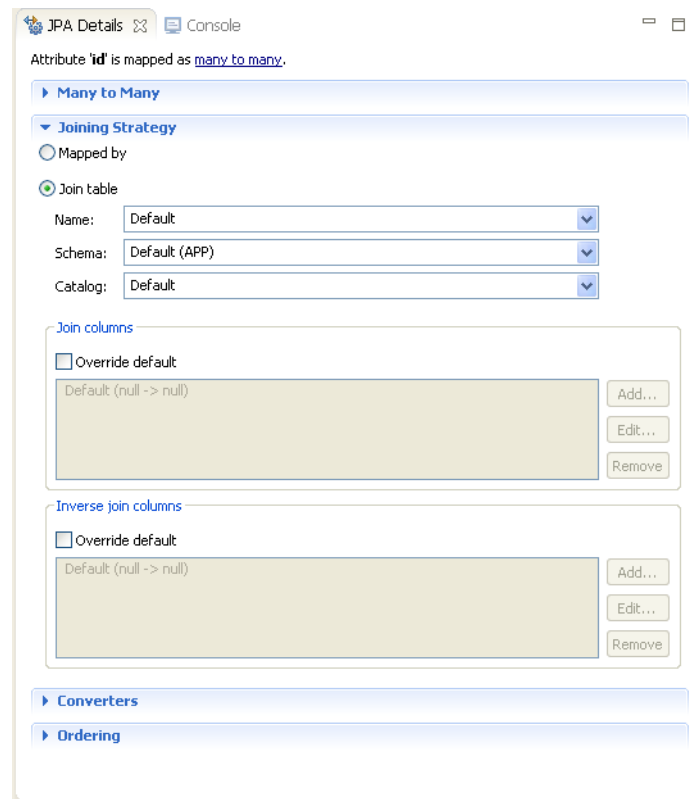
1. In the [JPA Structure view](#), select the field to map.
2. Right-click the field and then select **Map As > Many-to-Many**. The [JPA Details view \(for attributes\)](#) displays the properties for the selected field.

Figure 3–54 JPA Details, Many to many mapping



3. Complete each field in the [Many-to-Many Mapping](#) area.
4. Use the Joining Strategy area to specify the join strategy (or table) for the mapping.

Figure 3–55 JPA Details, Joining Strategy



5. Complete each field in the [Joining Strategy](#) area.
6. Complete the remaining areas in the [JPA Details view \(for attributes\)](#).

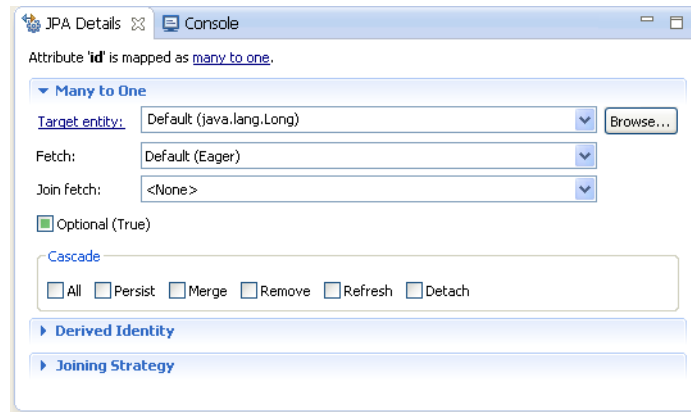
Eclipse adds the following annotations to the field:

```
@JoinTable(
    joinColumns=@JoinColumn(name="<JOIN_COLUMN>"),
    name = "<JOIN_TABLE_NAME>"
)
@ManyToMany(=Type.<_TYPE>, fetch=FetchType.<FETCH_TYPE>,
    targetEntity=<TARGET_ENTITY>, mappedBy = "<MAPPED_BY>")
@OrderBy("<ORDER_BY>")
```

Many-to-one mapping

Use a **Many-to-One** mapping to defines a single-valued association to another entity class that has many-to-one multiplicity.

1. In the [JPA Structure view](#), select the field to map.
2. Right click the field and then select **Map As > Many-to-One**. The [JPA Details view \(for attributes\)](#) displays the properties for the selected.

Figure 3–56 JPA Details, Many-to-one mapping

3. Complete each field in the [Many-to-Many Mapping](#) area.
4. Complete the remaining areas in the [JPA Details view \(for attributes\)](#).

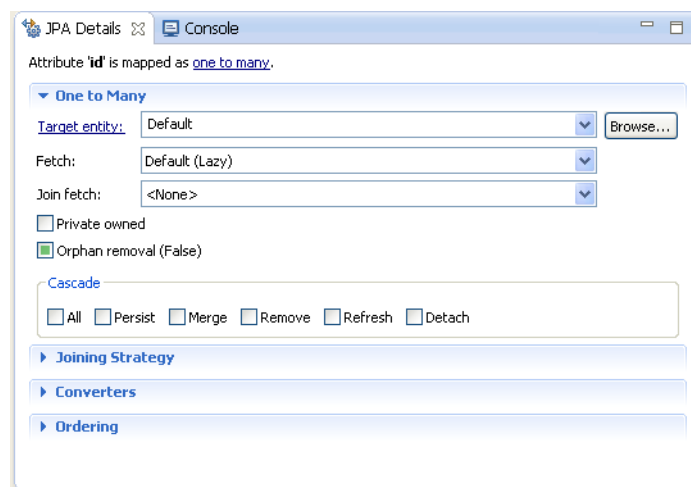
Eclipse adds the following annotations to the field:

```
@JoinTable(joinColumns=@JoinColumn(name="<JOIN_COLUMN>"),
    name = "<JOIN_TABLE_NAME>")
@ManyToOne(
    targetEntity=<TARGET_ENTITY>,
    fetch=<FETCH_TYPE>,
    =<_TYPE>
)
```

One-to-many mapping

Use a **One-to-Many Mapping** to define a relationship with one-to-many multiplicity.

1. In the [JPA Structure view](#), select the field to map.
2. Right-click the field and then select **Map As > One-to-many**. The [JPA Details view \(for attributes\)](#) displays the properties for the selected.

Figure 3–57 JPA Details, One-to-many Mapping

3. Complete each field in the [One-to-Many Mapping](#) area.

4. Complete the remaining areas in the [JPA Details view \(for attributes\)](#):

- [Joining Strategy](#)
- [Converters](#)
- [Ordering](#)

Eclipse adds the following annotations to the field:

```
@OneToMany(targetEntity=<TARGET_ENTITY>)
@Column(name="<COLUMN>")

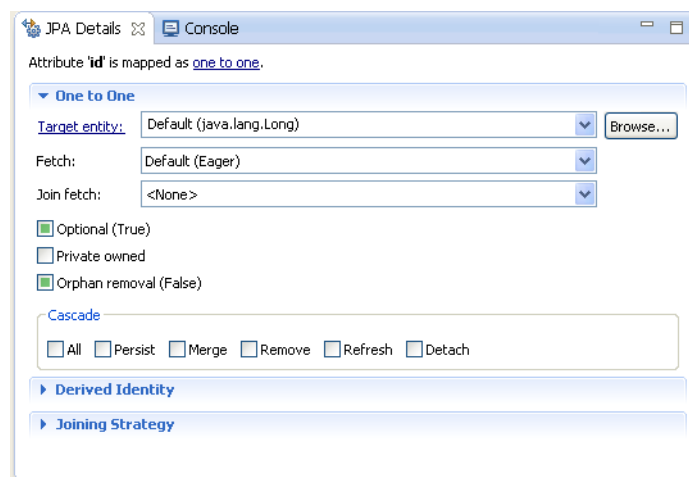
@OneToMany(targetEntity=<TARGET_ENTITY>.class,
           =Type.<_TYPE>,
           fetch = FetchType.<FETCH_TYPE>,
           mappedBy = "<MAPPED_BY>"
)
@OrderBy("<ORDER_BY>")
@JoinTable(name="<JOIN_TABLE_NAME>", joinColumns=@JoinColumn(name=
"<JOIN_COLUMN_NAME>", referencedColumnName="<JOIN_COLUMN_REFERENCED_COLUMN>"),
inverseJoinColumns=@JoinColumn(name="<INVERSE_JOIN_COLUMN_NAME>",
referencedColumnName="<INVERSE_JOIN_COLUMN_REFERENCED_COLUMN>"))
```

One-to-one mapping

Use a **One-to-One Mapping** to define a relationship with one-to-many multiplicity.

1. In the [JPA Structure view](#), select the field to map.
2. Right-click the field and then select **Map As > One-to-One**. The [JPA Details view \(for attributes\)](#) displays the properties for the selected.

Figure 3-58 JPA Details, One-to-one Mapping



3. Complete each field in the [One-to-One Mapping](#) area.
4. Complete the remaining areas in the [JPA Details view \(for attributes\)](#):
 - [Joining Strategy](#)
 - [Derived Identity](#)

Eclipse adds the following annotations to the field:

```
@OneToOne(targetEntity=<TARGET_ENTITY>, =Type.<_TYPE>,
    fetch = FetchType.<FETCH_TYPE>, mappedBy = "<MAPPED_BY>")
@JoinColumn(name="<JOIN_COLUMN_NAME>", referencedColumnName=
    "<JOIN_COLUMN_REFERENCED_COLUMN>", insertable = <INSERTABLE>,
    updatable = <UPDATABLE>)
```

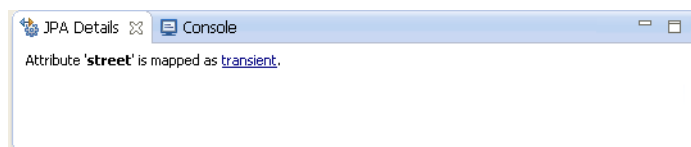
Transient mapping

Use the Transient Mapping to specify a field of the entity class that *is not* persistent.

To create a transient mapping:

1. In the [JPA Structure view](#), select the field to map.
2. Right-click the field and then select **Map As Transient**. The [JPA Details view \(for attributes\)](#) displays the properties for the selected.

Figure 3–59 JPA Details, Transient Mapping



There are no additional options for Transient mappings. Eclipse adds the following annotation to the field:

```
@Transient
```

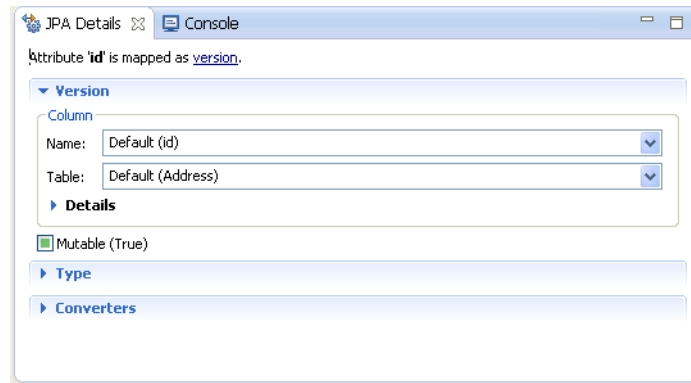
Version mapping

Use a **Version Mapping** to specify the field used for optimistic locking. If the entity is associated with multiple tables, you should use a version mapping only with the primary table. You should have only a single version mapping per persistent entity. Version mappings may be used only with the following attribute types:

- int
- Integer
- short, Short
- long, Long
- Timestamp

To create a version mapping:

1. In the [JPA Structure view](#), select the field to map.
2. Right-click the field and then select **Map As > Version**. The [JPA Details view \(for attributes\)](#) displays the properties for the selected.

Figure 3–60 JPA Details, Version Mapping

3. Complete each field in the [Version Mapping](#) area.
4. Complete the remaining areas in the [JPA Details view \(for attributes\)](#):
 - [Type information](#)
 - [Converters](#)

Eclipse adds the following annotations to the field:

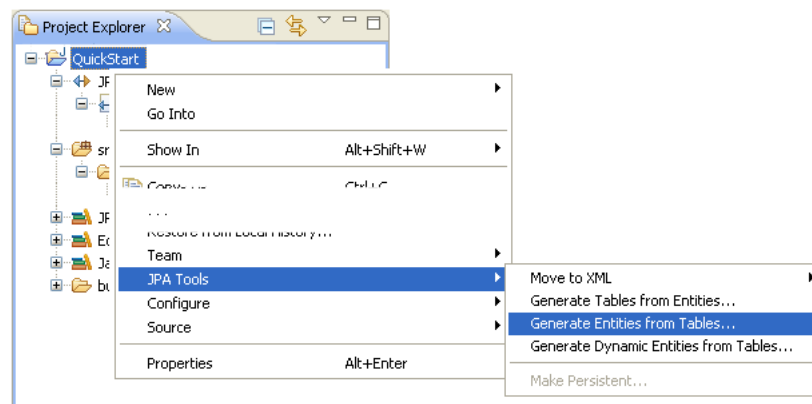
```
@Version
@Column(table="<COLUMN_TABLE>", name="<COLUMN_NAME>")
```

Generating entities from tables

Use this procedure to generate Java persistent entities from database tables. You must create a JPA project and establish a database connection *before* generating persistent entities. See ["Creating a new JPA project"](#) on page 3-1 for more information.

To use the [Generate Tables from Entities wizard](#):

1. Right-click the JPA project in the Project Explorer and select **JPA Tools > Generate Entities from Tables**.

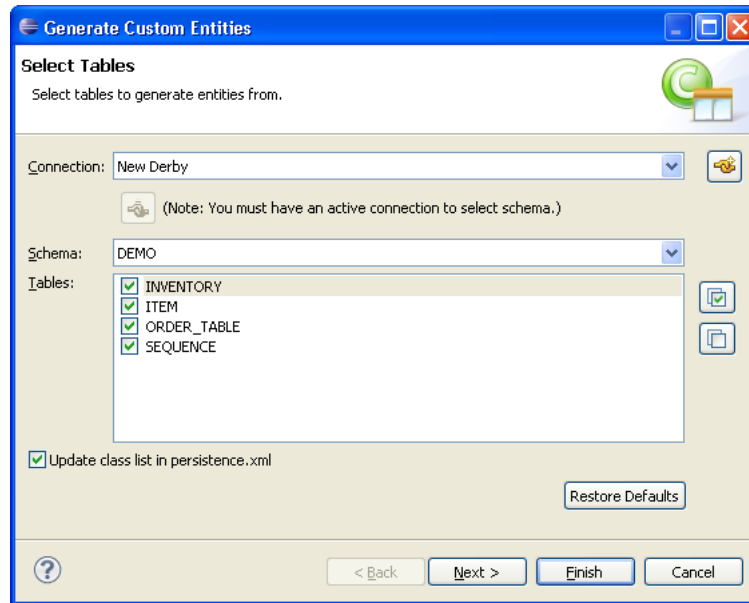
Figure 3–61 Generating Entities from Tables

2. On the [Select Tables](#) page of the [Generate Entities from Tables wizard](#), select your database connection and schema.

To create a new database connection, click **Add connection**.

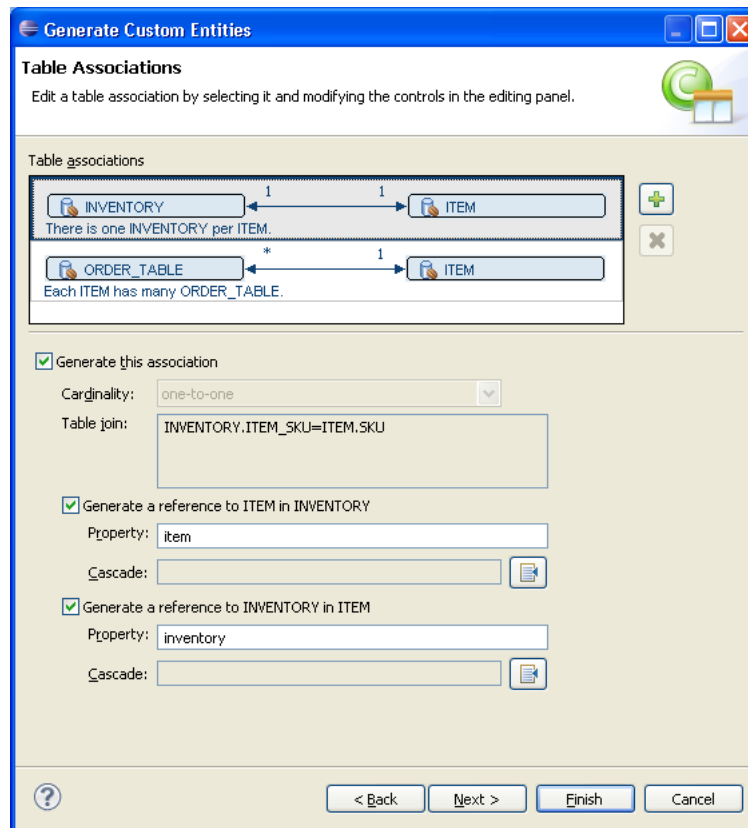
If you are not currently connected to the database, the Database Connection page appears. Select your database connection and schema, and click **Reconnect**.

Figure 3–62 *Select Tables*



3. After selecting a schema, select the tables from which to generate Java persistent entities and click **Next**.
4. On the [Table Associations](#) page, select the associations to generate. You can specify to generate specific references for each association.

To create a new association, click **Add Association**. Use the [Create New Association wizard](#) wizard to define the association.

Figure 3–63 Table Associations

5. After editing the table associations, click **Next**.
6. On the [Customize Default Entity Generation](#) page, customize the mapping and class information for each generated entity.

Figure 3–64 Customize Default Entity Generation

The screenshot shows a dialog box titled "Generate Custom Entities" with a sub-tab "Customize Defaults". The main instruction is "Please specify a sequence name".

Mapping defaults

- Key generator:
- Sequence name:
You can use the patterns \$table and/or \$pk in the sequence name. These patterns will be replaced by the table name and the primary key column name when a table mapping is generated.
- Entity access: Field Property
- Associations fetch: Default Eager Lazy
- Collection properties type: java.util.Set java.util.List
- Always generate optional JPA annotations and DDL parameters

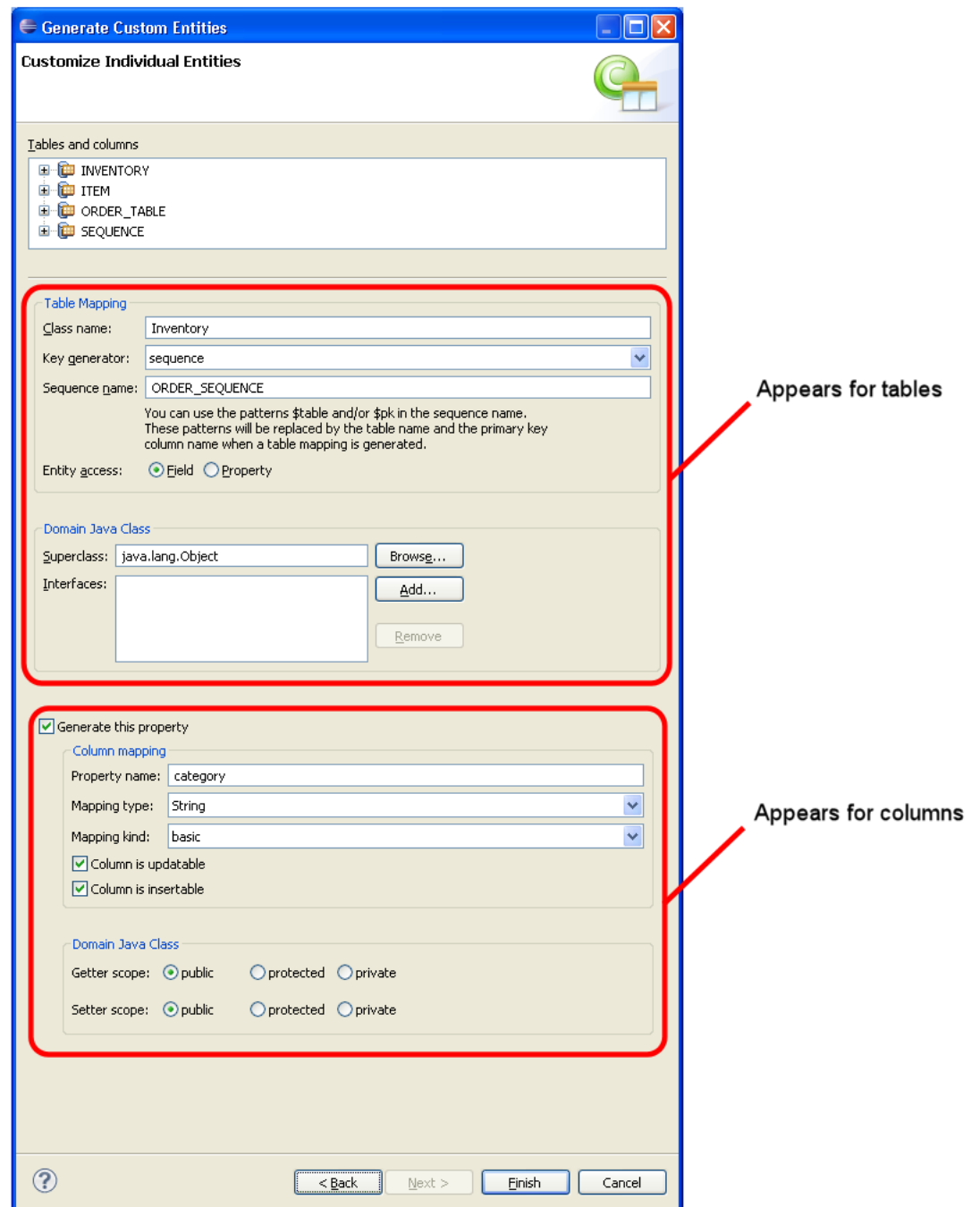
Domain java class

- Source folder:
- Package:
- Superclass:
- Interfaces:

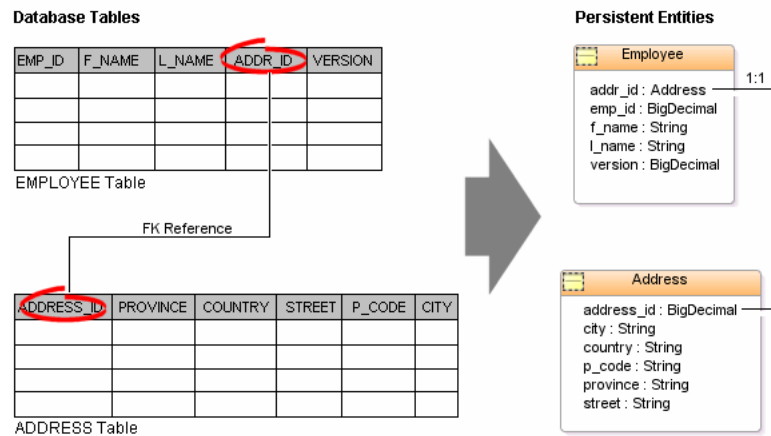
At the bottom, there are navigation buttons: "?", "< Back", "Next >", "Finish", and "Cancel".

7. After customizing the mappings, click **Next**.
8. On the [Customize Individual Entities](#) page, review the mapping and class information for each entity that will be generated, then click **Finish**.

Figure 3–65 Customize Individual Entities



Eclipse creates a Java persistent entity for each database table. Each entity contains fields based on the table's columns. Eclipse will also generate entity relationships (such as one-to-one) based on the table constraints. Figure 3–66 illustrates how Eclipse generates entities from tables.

Figure 3–66 Generating Entities from Tables

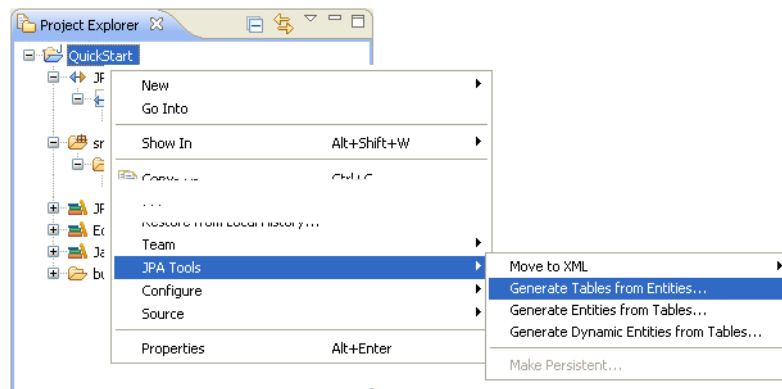
Generating tables from entities

When using a vendor-specific platform, you can create a DDL script from your persistent entities.

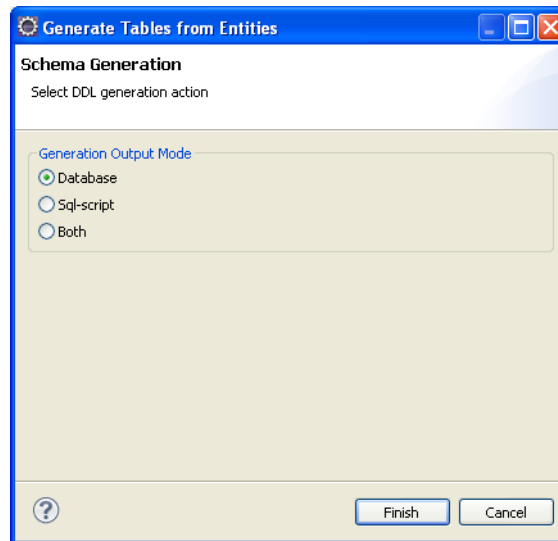
Note: The DDL script will DROP existing tables on the database and CREATE new tables, based on the entities in your project.

To generate a DDL script:

1. Right-click the JPA project in the Project Explorer and select **JPA Tools > Generate Tables from Entities**.



2. On the [Schema Generation](#) page, select the generation output mode.

Figure 3–67 Schema Generation

3. Click **Finish**. Dali generates the selected DDL for the entities, as shown in [Example 3–3](#).

If you are not currently connected to the database, the Database Connection page appears. Select your database connection and schema, and click **Reconnect**.

Example 3–3 Sample Generated Output

```
[EL Config]: metadata: The access type for the persistent class [class
quickstart.demo.model.Address] is set to [FIELD].
[EL Config]: metadata: The alias name for the entity class [class quickstart.demo.model.Address] is
being defaulted to: Address.
[EL Config]: metadata: The table name for entity [class quickstart.demo.model.Address] is being
defaulted to: ADDRESS.
[EL Config]: metadata: The column name for element [street] is being defaulted to: STREET.
[EL Config]: metadata: The column name for element [city] is being defaulted to: CITY.
[EL Config]: metadata: The column name for element [country] is being defaulted to: COUNTRY.
[EL Info]: EclipseLink, version: Eclipse Persistence Services - 2.4.0.vXXXX
[EL Fine]: connection: Detected database platform:
org.eclipse.persistence.platform.database.JavaDBPlatform
[EL Config]: connection: Connection(7896086)--connecting(DatabaseLogin(
  platform=>JavaDBPlatform
  user name=> ""
  datasource URL=> "jdbc:derby:C:\MyDB;create=true"
))
[EL Config]: connection: Connection(28523022)--Connected: jdbc:derby:C:\MyDB
User: APP
Database: Apache Derby Version: 10.9.1.0 - (XXXX)
Driver: Apache Derby Embedded JDBC Driver Version: 10.9.1.0 - (XXXX)
[EL Config]: connection: Connection(27817788)--connecting(DatabaseLogin(
  platform=>JavaDBPlatform
  user name=> ""
  datasource URL=> "jdbc:derby:C:\MyDB;create=true"
))
[EL Config]: connection: Connection(11557581)--Connected: jdbc:derby:C:\MyDB
User: APP
Database: Apache Derby Version: 10.9.1.0 - (XXXX)
Driver: Apache Derby Embedded JDBC Driver Version: 10.9.1.0 - (XXXX)
```

```
[EL Info]: connection: file:/C:/workspace/runtime-EclipseApplication/QuickStart/build/classes/_
QuickStart_url=jdbc:derby:C:\MyDB;create=true login successful
[EL Fine]: sql: Connection(28523022)--DROP TABLE ADDRESS
[EL Fine]: sql: Connection(28523022)--CREATE TABLE ADDRESS (ADDRESS_ID BIGINT NOT NULL, CITY
VARCHAR(255), COUNTRY VARCHAR(255), P_CODE VARCHAR(255), PROVINCE VARCHAR(255), STREET
VARCHAR(255), PRIMARY KEY (ADDRESS_ID))
[EL Config]: connection: Connection(28523022)--disconnect
[EL Info]: connection: file:/C://workspace/runtime-EclipseApplication/QuickStart/build/classes/_
QuickStart_url=jdbc:derby:C:\MyDB;create=true logout successful
[EL Config]: connection: Connection(7896086)--disconnect
[EL Config]: connection: Connection(11557581)--disconnect
```

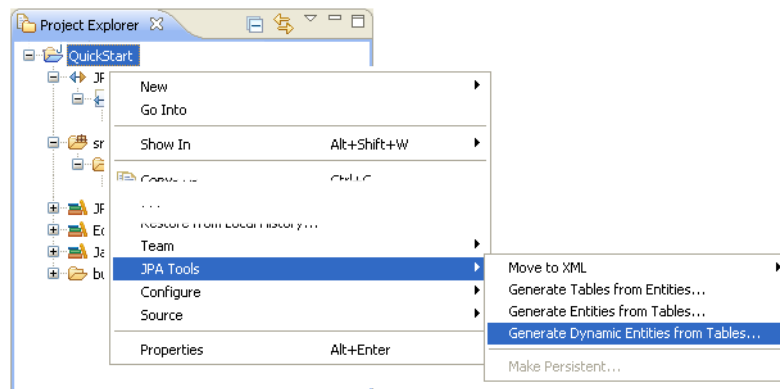
Generating dynamic entities from tables

When using EclipseLink JPA, you can create dynamic entities from your database tables. This dynamic persistence provides access to a relational database with all the benefits of JPA *without coding* or maintaining Java classes.

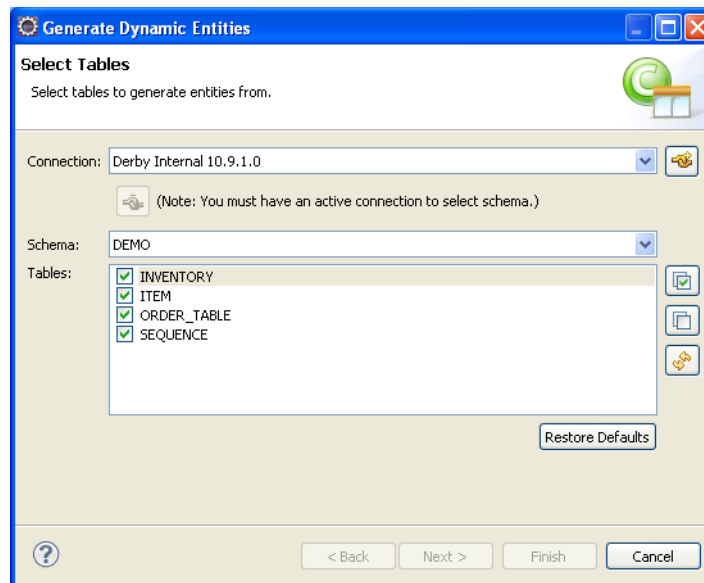
Dali dynamically creates the classes at runtime, as needed.

To generate dynamic entities:

1. Right-click the JPA project in the Project Explorer and select **JPA Tools > Generate Dynamic Entities from Tables**.

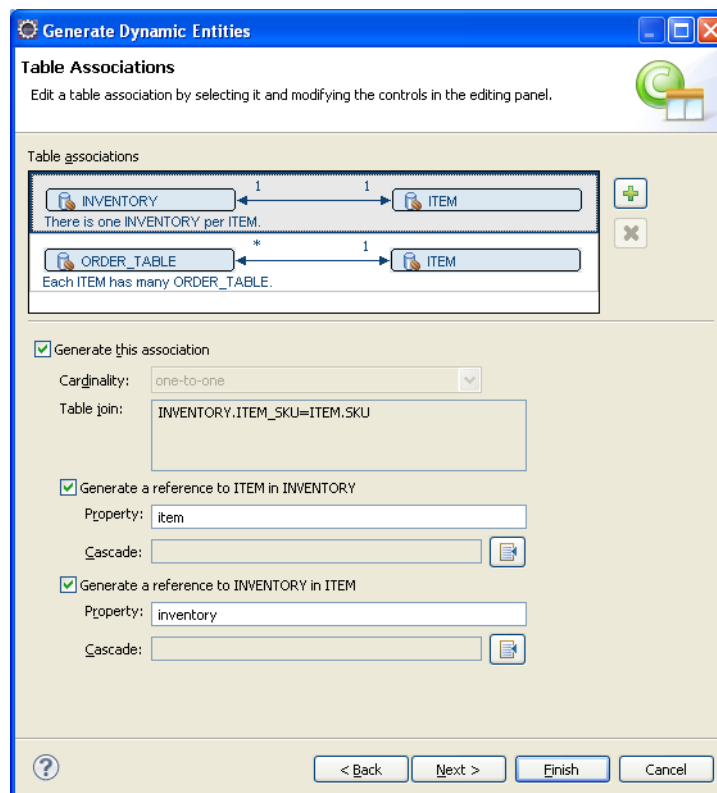


The Select Tables page of the [Generate Dynamic Entities from Tables wizard](#) appears.

Figure 3–68 Select Tables

2. On the [Select Tables](#) page, select the tables from which to generate the entities and click [Next](#).

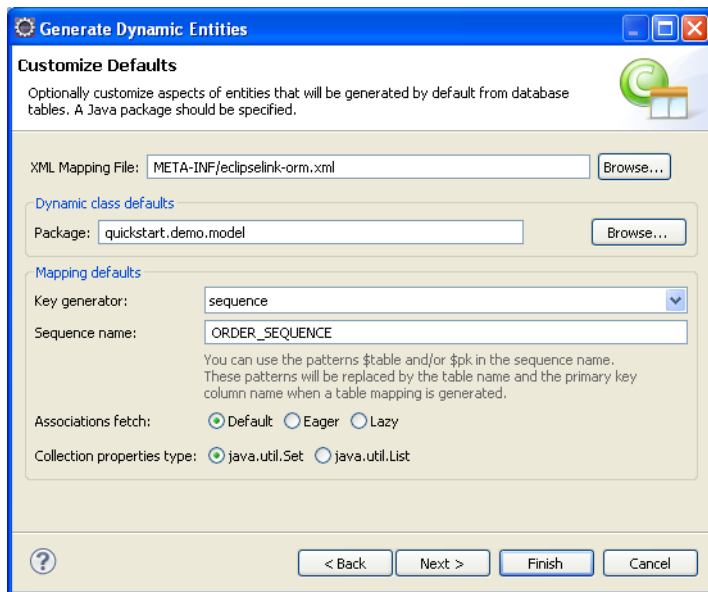
The [Table Associations](#) page appears.

Figure 3–69 Table Associations

3. On the [Table Associations](#) page, specify which table associations should be generated. Use the [Create New Association wizard](#) to create additional relationships.

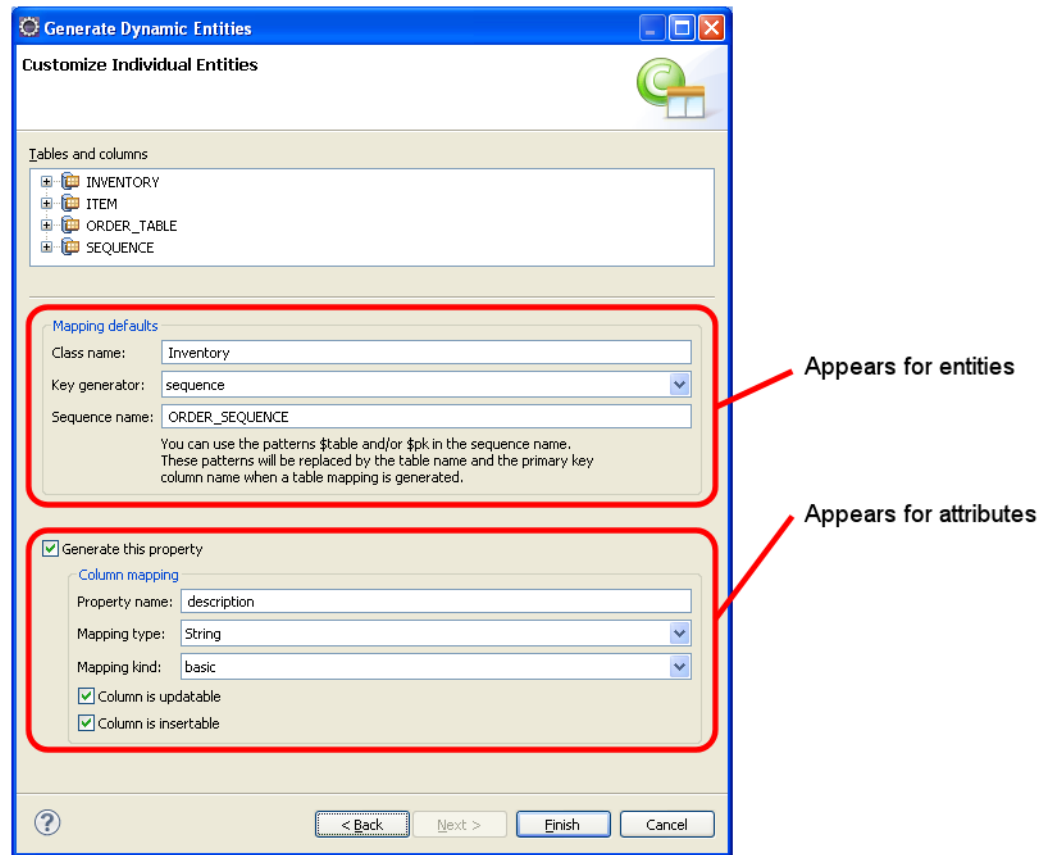
Click **Next**. The [Customize Default Entity Generation](#) page appears.

Figure 3–70 *Customize Defaults*



4. On the [Customize Default Entity Generation](#) page, specify the default information to use when generating the entities, and click **Next**.

The [Customize Individual Entities](#) page appears.

Figure 3–71 Customize Individual Entities

5. Use the [Customize Individual Entities](#) page to customize specific generated entities.

6. Click **Finish** to complete the wizard and generate the entities.

Dali generates the dynamic entities, using the `VIRTUAL` access type, as shown in [Example 3–4](#).

Example 3–4 Sample `eclipselink-orm.xml` File with Dynamic Entities

```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings version="2.4"
xmlns="http://www.eclipse.org/eclipselink/xsds/persistence/orm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.eclipse.org/eclipselink/xsds/persistence/orm
http://www.eclipse.org/eclipselink/xsds/eclipselink_orm_2_4.xsd">
  <entity class="quickstart.demo.model.Inventory" access="VIRTUAL">
    <attributes>
      <id name="itemSku" attribute-type="long">

        </id>
    </attributes>
  </entity>
  <entity ...">
  </entity>
</entity-mappings>
```

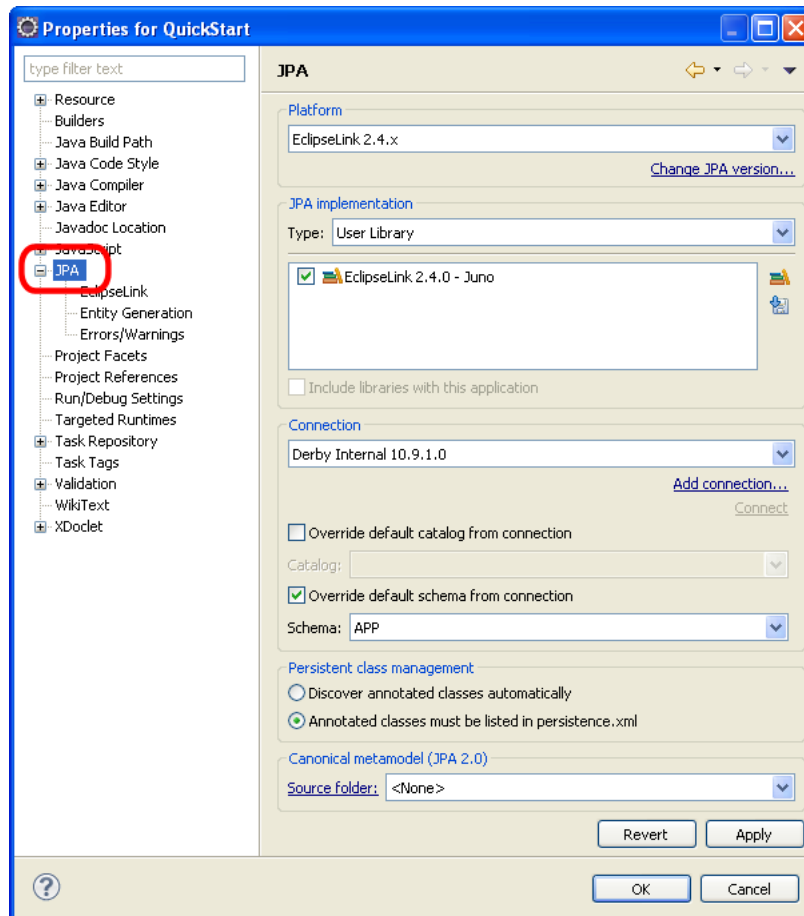
Modifying persistent project properties

Each persistent project must be associated with a database connection. To create a new database connection, click **Database Connection** use the New Connection wizard.

Use this procedure to modify the vendor-specific platform and database connection associated with your JPA project.

1. Right-click the project in the Explorer view and select **Properties**. The Properties page appears.
2. Select **JPA**.

Figure 3–72 The Properties Page

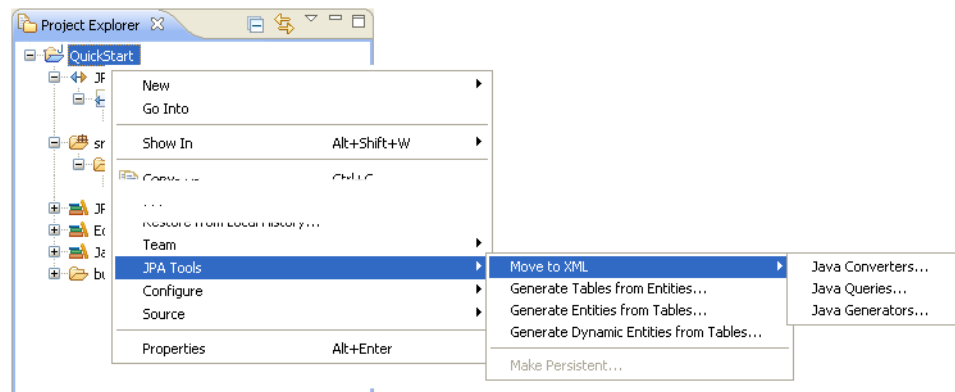


3. Complete each field on the [Project Properties page – JPA](#) click **OK**.

Converting JPA metadata to XML

Starting in Release 3.2, Dali can convert metadata (such as converters, queries, and generators) into an XML mapping file. This allows you to maintain the global metadata for a persistence unit (such as queries and generators) in an XML mapping file.

1. Right-click the project in the Explorer view and select **JPA Tools > Move to XML > specific metadata (such as Java Converters, Java Queries, or Java Generators)**.

Figure 3–73 Moving Metadata to XML

The [JPA Metadata Conversion dialog](#) page appears.

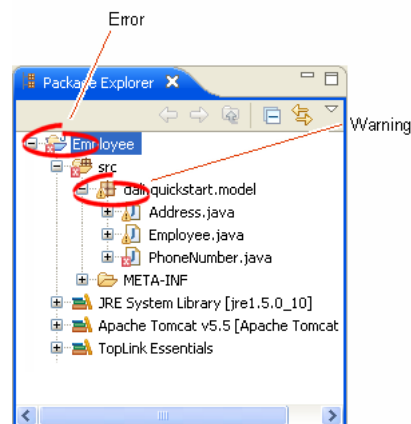
2. Enter the filename and location of the XML mapping file, and click **Finish**.

Dali generates the XML file.

Validating mappings and reporting problems

Errors and warnings on persistent entities and mappings are indicated with a red error or yellow warning next to the resource with the error, as well as the parent containers up to the project.

Tip: Use the [Project Properties page – Errors/Warnings](#) and [Java Persistence Preferences page – Errors/Warnings](#) to specify which problems Dali will report.

Figure 3–74 Sample Errors and Warnings

This section contains information on the following:

- [Error messages](#)
- [Warning messages](#)

Error messages

This section contains information on error messages (including how to resolve the issue) you may encounter while working with Dali.

An exception handler class should be specified.

When using a custom exception handler, you must select (or create) a Java class to handle exceptions. See ["Customization"](#) on page 4-40.

Attribute "<ATTRIBUTE_NAME>" has invalid mapping type in this context

The mapped attribute is invalid. Either change the mapping type or change the entity type.

See ["Mapping an entity"](#) on page 3-34 for more information.

Attribute "<ATTRIBUTE_NAME>" cannot be resolved.

Dali cannot map the attribute to a database table and column. Verify that your database connection information is correct.

See ["Creating a new JPA project"](#) on page 3-1 for more information.

Class "<CLASS_NAME>" is not annotated as a persistent class.

The class has not been identified as a persistent class. Configure the class as an Entity, Mapped Superclass, or Embeddable persistent entity.

See ["Adding persistence to a class"](#) on page 3-15.

Column "<COLUMN_NAME>" cannot be resolved.

You mapped an entity's field to an incorrect or invalid column in the database table. By default, Dali will attempt to map each field in the entity with an identically named row in the database table. If the field's name differs from the row's name, you must explicitly create the mapping.

Map the field to a valid row in the database table as shown in ["Mapping an entity"](#) on page 3-34.

Converter is unnamed. All converters require a name.

When creating a converter, you must specify its name. See ["Converters"](#) on page 4-21.

Converter name must not be a reserved converter name.

When creating a converter, you must not use the following reserved names:

- serialized
- class-instance
- none

Duplicate class "<CLASS_NAME>".

You created two persistence classes with the same name. Each Java class must have a unique name. See ["Adding persistence to a class"](#) on page 3-15 for more information.

Entity does not have an Id or Embedded Id.

You created a persistent entity without identifying its primary key. A persistent entity must have a primary key field designated with an @Id or @EmbeddedId annotation.

Add an ID mapping to the entity as shown in ["ID mapping"](#) on page 3-39 or ["Embedded ID mapping"](#) on page 3-38.

Multiple generators named "<GENERATOR_NAME>" defined in this persistence unit.

When creating generators, the converter **Name** must be unique within the persistence unit. See ["Primary Key Generation"](#) on page 4-20.

Multiple persistence.xml files in project.

You created a JPA project with more than one `persistence.xml` file. Each JPA project must contain a *single* `persistence.xml` file.

See ["Managing the persistence.xml file"](#) on page 3-20 for more information.

Multiple converters named "<CONVERTER_NAME>" defined in this persistence unit

When creating converters, the converter **Name** must be unique within the persistence unit. See ["Add Converter dialog"](#) on page 4-55.

No persistence unit defined.

There is no persistence unit defined in the `persistence.xml` file. Use the `<persistence-unit name="<PERSISTENCE_UNIT_NAME>"` tag to define the persistent unit.

See ["Managing the orm.xml file"](#) on page 3-28 for more information.

No persistence.xml file in project.

You created a JPA project without a `persistence.xml` file. Each JPA project must contain a *single* `persistence.xml` file.

See ["Managing the persistence.xml file"](#) on page 3-20 for more information.

Property "<PROPERTY_NAME>" will be ignored as shared-cache-mode is set to NONE.

Because the **Shared cache mode** option is set to **NONE**, Dali will ignore the property. See ["Caching"](#) on page 4-42.

Referenced column "<COLUMN_NAME>" in join column "<COLUMN_NAME>" cannot be resolved.

The column that you selected to join a relationship mapping does not exist on the database table. Either select a different column on the [Joining Strategy](#) or create the necessary column on the database table.

See ["JPA Details view \(for attributes\)"](#) on page 4-22 for more information.

Schema "<SCHEMA_NAME>" cannot be resolved for table/join table "<TABLE_NAME>"

Define the default database schema information in the persistence unit.

See ["Managing the orm.xml file"](#) on page 3-28 for more information.

Table "<TABLE_NAME>" cannot be resolved.

You associated a persistent entity to an incorrect or invalid database table. By default, Dali will attempt to associate each persistent entity with an identically named database table. If the entity's name differs from the table's name, you must explicitly create the association.

Associate the entity with a valid database table as shown in ["Adding persistence to a class"](#) on page 3-15.

The @Cache annotation on entity <ENTITY_NAME> has both expiry() and expiryTimeOfDay() specified.

You attempted to include both `expiry` and `expiryTimeOfDay` in the `@Cache` annotation. You may use only one. See ["Caching"](#) on page 4-15.

The converter class "<CLASS_NAME>" does not exist on the project classpath.

You defined a convert class but did not include the class within the project. See ["Converters"](#) on page 4-21.

The converter class "<CLASS_NAME>" does not implement the org.eclipse.persistence.mappings.converters.Converter interface

When creating a converter, its class must implement the `org.eclipse.persistence.mappings.converters.Converter` interface. See ["Converters"](#) on page 4-21.

The converter class must be defined.

You attempted to use a converter without defining the class. See ["Converters"](#) on page 4-21.

The entity customizer class "<CLASS_NAME>" does not implement the org.eclipse.persistence.config.DescriptorCustomizer interface.

When using a customer class for an entity, the class must implement the `org.eclipse.persistence.config.DescriptorCustomizer` interface. See ["Advanced"](#) on page 4-22

The exception handler class "<CLASS_NAME>" does not implement the org.eclipse.persistence.exceptions.ExceptionHandler interface.

When using a custom exception handler, you must select (or create) a Java class that implements the `org.eclipse.persistence.exceptions.ExceptionHandler` class. See ["Customization"](#) on page 4-40.

The persistent field or property for a Version mapping must be of type int, Integer, short, Short, long, Long, or Timestamp.

Version mappings may be used only with the following attribute types:

- `int`
- `Integer`
- `short`, `Short`
- `long`, `Long`
- `Timestamp`

See ["Version mapping"](#) on page 3-44.

The struct converter class "<CLASS_NAME>" does not implement the `org.eclipse.persistence.platform.database.converters.StructConverter` interface.

When creating a Struct converter (to enable custom processing of `java.sql.Struct` types), its class must implement the

`org.eclipse.persistence.mappings.converters.StructConverter` interface.

See ["Converters"](#) on page 4-21.

Unresolved generator "<GENERATOR_NAME>" is defined in persistence unit.

You created a persistence entity that uses sequencing or a table generator, but did not define the generator in the persistence unit. Either define the generator by using an annotation or including it in the XML mapping file.

Warning messages

This section contains information on warning messages (including how to resolve the issue) you may encounter while working with Dali.

Connection "<CONNECTION_NAME>" is not active. No validation will be done against the data source.

The database connection you specified to use with the JPA project is not active. The JPA project requires an active connection.

No connection specified for the project. No data-specific validation will be performed.

You created a JPA project without specifying a database connection. The JPA project requires an active connection.

See ["Creating a new JPA project"](#) on page 3-1 or ["Modifying persistent project properties"](#) on page 3-56 for information on specifying a database connection.

This section includes detailed help information for each of the following elements in the Dali OR Mapping Tool:

- [Wizards](#)
- [Property pages](#)
- [Preferences](#)
- [Dialogs](#)
- [JPA Development perspective](#)
- [Icons and buttons](#)
- [Dali developer documentation](#)

Wizards

This section includes information on the following wizards:

- [Generate Entities from Tables wizard](#)
- [Generate Dynamic Entities from Tables wizard](#)
- [Create JPA Entity wizard](#)
- [Create ORM Mapping File wizard](#)
- [Create New JPA Project wizard](#)
- [Create New JAXB Project wizard](#)
- [New Database Web services from Builder XML wizard](#)
- [Generate Tables from Entities wizard](#)
- [Create New Association wizard](#)

Generate Entities from Tables wizard

Use the Generate Custom Entities Wizard to create JPA entities from your database tables.

The wizard consists of the following pages:

- [Select Tables](#)
- [Table Associations](#)
- [Customize Default Entity Generation](#)

- [Customize Individual Entities](#)

Select Tables

Use the Select Tables dialog to specify the database connection and tables from which to create entities.

Property	Description
Connection	Select a database connection or click Add Connection to create a new connection.
Schema	Select the database schema from which to select tables.
Tables	Select the tables from which to create Java persistent entities. The tables shown are determined by the database connection and schema selections.
Update class list in persistence.xml	Specify if Dali should update the persistence.xml file to include the generated classes.

Table Associations

Use this page to create or edit the association between the database table and entity.

Property	Description
Table associations	Select an association to modify or click New Association to create a new table association with the Create New Association wizard wizard.
Generate this association	Specify if Dali should create the selected association. If enabled, you can specify the Cardinality and Table join for the table association.
Generate a reference to <ROW> in <TABLE>	Specify if the entity should contain a reference to the specified table. If enabled, you can also enter the Property name and select the Cascade method (all, persist, merge, remove, or refresh) for the reference.

Customize Default Entity Generation

Use this page to specify the default information Dali will use when generating the entities from the database tables. You will be able to override this information for specific entities.

Property	Description
Mapping defaults	Use these options to define the table mapping information for the entity.
Key generator	Select the generator used for this mapping.
Sequence name	Enter a name for the sequence. You can use \$table and \$pk as variables in the name. These will be replaced by the table name and primary key column name (respectively) when Dali generates a table mapping.
Entity access	Specify the default entity access method: Field (default) or Property .
Associations fetch	Specify the default fetch mode for associations: Default , as defined by the application (default), or Lazy .

Property	Description
Collection properties type	Specify if the collection properties are a Set or List .
Always generate optional JPA annotations and DDL parameters	Specify if Dali should include this information in the entity.
Domain Java class	Use these options to define the Source folder and class information (Package , Superclass , and Interfaces) for the entity.

Customize Individual Entities

Use this page to customize each generated entity. Select an item in the **Table and columns** area, then complete the following fields for each item.

Property	Description
Mapping defaults	Use these options to define the table mapping information for the entity.
Class name	Name of the entity class
Key generator	Select the generator used for this mapping.
Sequence name	Enter a name for the sequence. You can use \$table and \$pk as variables in the name. These will be replaced by the table name and primary key column name (respectively) when Dali generates a table mapping.
Entity access	Specify the default entity access method: Field (default) or Property .
Domain Java Class	Use these options to define the class information (Superclass and Interfaces) for the entity.
Generate this property	Enable this option to generate the following properties for the selected column.
Column mapping	
Property name	The name of the property derived from the column
Mapping type	The attribute type
Mapping kind	The type of mapping for the attribute
Column is updatable	Specify if the column is included in SQL <code>UPDATE</code> statements.
Column is insertable	Specify if the column is included in SQL <code>INSERT</code> statements.
Domain Java Class	Use these options to define the getter and setter scope for the entity.

Generate Dynamic Entities from Tables wizard

Use the Generate Dynamic Custom Entities wizard to create dynamic EclipseLink JPA entities from your database tables.

The wizard consists of the following pages:

- [Select Tables](#)
- [Table Associations](#)
- [Customize Default Entity Generation](#)
- [Customize Individual Entities](#)

Select Tables

Use the Select Tables dialog to specify the database connection and tables from which to create entities.

Property	Description
Connection	Select a database connection or click Add Connection to create a new connection.
Schema	Select the database schema from which to select tables.
Tables	Select the tables from which to create Java persistent entities. The tables shown are determined by the database connection and schema selections.
Update class list in persistence.xml	Specify if Dali should update the <code>persistence.xml</code> file to include the generated classes.

Table Associations

Use this page to create or edit the association between the database table and entity.

Property	Description
Table associations	Select an association to modify or click New Association to create a new table association with the Create New Association wizard wizard.
Generate this association	Specify if Dali should create the selected association. If enabled, you can specify the Cardinality and Table join for the table association.
Generate a reference to <ROW> in <TABLE>	Specify if the entity should contain a reference to the specified table. If enabled, you can also enter the Property name and select the Cascade method (all, persist, merge, remove, or refresh) for the reference.

Customize Default Entity Generation

Use this page to specify the default information Dali will use when generating the entities from the database tables. You will be able to override this information for specific entities.

Property	Description	Default
XML Mapping File	The name and location of the mapping file.	META-INF/eclipse-link-orm.xml
Dynamic Class Defaults		
Package	Default package name for dynamic classes	model
Mapping defaults		
Key generator	Default generation strategy for primary keys: <ul style="list-style-type: none"> ▪ Auto ▪ Identity ▪ Sequence ▪ Table ▪ None 	None

Property	Description	Default
Sequence name	When using a Key generator , specify its name. Note: You can use the variables <code>\$table</code> and <code>\$pk</code> in the Sequence name . Dali will replace them with the <i>table name</i> and <i>primary key column name</i> , respectively, when generating a mapping table.	
Associations fetch	Specify the default fetch strategy for generated entities: <ul style="list-style-type: none"> ▪ Default ▪ Eager ▪ Lazy 	Default
Collection properties type	Specify the default collection type, for generated entities: <ul style="list-style-type: none"> ▪ <code>java.util.Set</code> ▪ <code>java.util.List</code> 	<code>java.util.List</code>

Customize Individual Entities

Use this page to customize each generated entity. Select an item in the **Table and columns** area, then complete the following fields for each item.

Property	Description
Mapping defaults	Use these options to define the table mapping information for the entity.
Class name	
Key generator	Select the generator used for this mapping.
Sequence name	Enter a name for the sequence. You can use \$table and \$pk as variables in the name. These will be replaced by the table name and primary key column name (respectively) when Dali generates a table mapping.
Entity access	Specify the default entity access method: Field (default) or Property .
Domain Java Class	Use these options to define the class information (Superclass and Interfaces) for the entity.

Create JPA Entity wizard

The Create JPA wizard enables you to quickly add an entity and also add persistence fields to that entity. In addition, this wizard adds the accessor methods (`getter` and `setter`) in the class file. The wizard consists of the following pages:

- [Entity Class page](#)
- [Entity Properties page](#)

Entity Class page

This table lists the properties of the Entity Class page of the [Create JPA Entity wizard](#).

Property	Description	Default
Project	The name of the JPA project.	
Source Folder	The location of the JPA project's src folder.	src
Java Package	The name of the class package.	
Class name	The name of the Java class.	
Superclass	Select the superclass.	
Inheritance	<p>Because the wizard creates a Java class with an <code>@Entity</code> notation, the Entity option is selected by default.</p> <p>Select Mapped Superclass if you defined a superclass.</p> <p>To add an <code>@Inheritance</code> notation to the entity, select Inheritance and then select one of the inheritance mapping strategies (described in JSR 220):</p> <ul style="list-style-type: none"> ▪ <code>SINGLE_TABLE</code> -- All classes in a hierarchy as mapped to a single table. This annotation is without an attribute for the inheritance strategy. ▪ <code>TABLE_PER_CLASS</code> -- Each class is mapped to a separate table. ▪ <code>JOINED</code> -- The root of the class hierarchy is represented by a single table. Each subclass is represented by a separate table that contains those fields that are specific to the subclass (not inherited from its superclass), as well as the column(s) that represent its primary key. The primary key column(s) of the subclass table serves as a foreign key to the primary key of the superclass table. 	Entity
XML Entity Mappings	<p>Select Add to entity mappings in XML to create XML mappings in <code>orm.xml</code>, rather than annotations.</p> <p>Use the Mapping file field to specify the file to use. By default, mappings are stored in the <code>META-INF/orm.xml</code> file.</p>	

Entity Properties page

This table lists the properties of the Entity Properties page of the [Create JPA Entity wizard](#).

Property	Description	Default
Entity name	The name of the entity. By default, this value is the same as the one entered as the class name. If the entity name differs from the class name, then the entity name is added as an attribute. For example: <code>@Entity(name="EntityName")</code> .	Determined by server.
Table name	Select Use default to match the name of the mapped table name to the entity name. Otherwise, clear the Use default option and enter the name in the <i>Table Name</i> field. These options result in the addition of the <code>@Table</code> option to the Java class file.	Use default.

Property	Description	Default
Entity fields	Click the Add button to add persistence fields using the Entity Fields dialog. This dialog enable you to build a field by entering a field name and selecting among persistence types. The Key option enables you to mark a field as a primary key. The dialog's Browse function enables you to add other persistence types described in the JPA specification. The Edit button enables you to change the name or type set for a persistent field.	
Access type	Select whether the entity's access to instance variables is field-based or property-based, as defined in the JPA specification. <ul style="list-style-type: none"> ▪ Field – Instance variables are accessed directly. All non-transient instance variables are persistent. ▪ Property – Persistent state accessed through the property accessor methods. The property accessor methods must be public or private. 	Field

Create ORM Mapping File wizard

The New Mapping File wizard enables you to add an `orm.xml` file to a JPA project if no object map exists at the location specified. For example, if you cleared the **Create `orm.xml`** option on the [JPA Facet page](#), you can later add the `orm.xml` file to the src file of the project using this wizard.

The [Create ORM Mapping File wizard](#) consists of the following pages:

- [Mapping File Location](#)
- [Mapping File Options](#)

Mapping File Location

Use this page of the [Create ORM Mapping File wizard](#) to specify the location of the ORM mapping file.

Property	Description	Default
Project	The name of the JPA project.	Selected.
Source folder	The location of the project's src folder. If needed, click Browse to point the wizard to the src file's location.	Selected.
File Path	The location for the new <code>orm.xml</code> file.	Selected.
File name	Name of the OR mapping file.	<code>orm.xml</code>

Mapping File Options

Use this page of the [Create ORM Mapping File wizard](#) to specify additional options for the ORM mapping file.

Property	Description	Default
Default Access	Select whether the access to the entity is field-based or property-based, as defined in JPA specification. <ul style="list-style-type: none"> ▪ None – No access type specified. ▪ Property-based – Persistent state accessed through the property accessor methods. The property accessor methods must be public or private. ▪ Field-based – Instance variables are accessed directly. All non-transient instance variables are persistent. 	None
Add to persistence unit	Designates the persistence unit for this object map file.	Current project's default persistence unit

Create New JPA Project wizard

The Create New JPA Project wizard allows you to create a new Java project using JPA. The wizard consists of the following pages:

- [New JPA Project page](#)
- [Java Page](#)
- [JPA Facet page](#)

New JPA Project page

This table lists the properties available on the New JPA Project page of the [Create New JPA Project wizard](#).

Property	Description	Default
Project name	Name of the Eclipse JPA project.	
Project location	Location of the workspace in which to save the project. Unselect the Use default location option and click Browse to select a new location.	Current workspace
Target runtime	Select a pre-defined target for the project. Click New Runtime to create a new environment with the New Server Runtime wizard.	<None>
JPA Version	Select the Java Persistence API version for the project.	2.0
Configurations	Select a project configuration with pre-defined facets. Select Modify to manually select the facets for this project.	
EAR membership	Specify if this project should be included in an EAR file for deployment. Select the EAR Project Name , or click New Project to create a new EAR project.	

Property	Description	Default
Working sets	Specify if this project should be included in an existing working set. The drop down field shows a list of previous selected working sets. Select Add project to working sets , then select a Working set in which to add this project.	

Java Page

This table lists the properties available on the Java page of the [Create New JPA Project wizard](#).

Property	Description	Default
Source folders on build path	Click Add Folder to select an existing Java source folder to add to this project.	src
Default output folder	Specify the location of the .class files.	build\classes

JPA Facet page

This table lists the properties available on the JPA Facet page of the [Create New JPA Project wizard](#).

Property	Description	Default
Platform	Vendor-specific JPA implementation.	Generic
JPA Implementation	Select a specific JPA library configuration. Click Manage libraries to create or update a user library. Click Download libraries to download a specific library configuration. Depending on your JPA implementation (for example, Generic or EclipseLink), different options may be available when working with JPA projects	
Type	Select User Library to select from the available user-defined or downloaded libraries. If you select Disable , you must manually include the JPA implementation library on the project classpath.	User Library
Include libraries with this application	Specify if the selected libraries are included when deploying the application.	Selected
Connection	Select the database connection to use with the project. Dali requires an active database connection to use and validate the persistent entities and mappings. Click Add connection to create a new database connection.	
Add driver library to build path	Specify if the connection driver libraries are included when deploying the application.	

Property	Description	Default
Override default schema from connection	Select a schema other than the default one that is derived from the connection information. Use this option if the default schema cannot be used. For example, use this option when the deployment login differs from the design-time login.	The value calculated by Dali.
Persistent class management	Specify if Dali will discover annotated classes automatically , or if the annotated classes must be listed in the persistence.xml file. Note: To insure application portability, you should explicitly list the managed persistence classes that are included in the persistence unit.	Determined by server.
Create mapping file (orm.xml)	Specify if Dali should create a default <code>orm.xml</code> file for your entity mappings and persistence unit defaults.	Selected

Create New JAXB Project wizard

The Create New JAXB Project wizard allows you to create a new Java project using JAXB. The wizard consists of the following pages:

- [New JAXB Project page](#)
- [Java Page](#)
- [JAXB Facet page](#)

New JAXB Project page

This table lists the properties available on the New JPA Project page of the [Create New JPA Project wizard](#).

Property	Description	Default
Project name	Name of the Eclipse JPA project.	
Project location	Location of the workspace in which to save the project. Unselect the Use default location option and click Browse to select a new location.	Current workspace
Target runtime	Select a pre-defined target for the project. Click New Runtime to create a new environment with the New Server Runtime wizard.	<None>
JAXB Version	Select the Java Architecture for XML Binding (JAXB) version for the project.	2.2
Configurations	Select a project configuration with pre-defined facets. Select Modify to manually select the facets for this project.	
Working sets	Specify if this project should be included in an existing working set. The drop down field shows a list of previous selected working sets. Select Add project to working sets , then select a Working set in which to add this project.	

Java Page

This table lists the properties available on the Java page of the [Create New JAXB Project wizard](#).

Property	Description	Default
Source folders on build path	Click Add Folder to select an existing Java source folder to add to this project.	src
Default output folder	Specify the location of the .class files.	build\classes

JAXB Facet page

This table lists the properties available on the JPA Facet page of the [Create New JAXB Project wizard](#).

Property	Description	Default
Platform	Vendor-specific JPA implementation.	Generic
JAXB Implementation	Select a specific JPA library configuration. Click Manage libraries to create or update a user library. Click Download libraries to download a specific library configuration. Depending on your JPA implementation (for example, Generic or EclipseLink), different options may be available when working with JPA projects	
Type	Select User Library to select from the available user-defined or downloaded libraries. If you select Disable , you must manually include the JPA implementation library on the project classpath.	User Library
Include libraries with this application	Specify if the selected libraries are included when deploying the application.	Selected

New Database Web services from Builder XML wizard

The New Database Web services from Builder XML wizard allows you to add database web services (DBWS) to an existing dynamic web services project, from an XML source. The wizard consists of the following pages:

- [Web Dynamic page](#)
- [Select Builder XML File page](#)
- [Driver Files page](#)

Web Dynamic page

Use this page to select the dynamic web services project in which to add the Database Web Services.

Select Builder XML File page

Use **this** page to select the XML files from which to generate the database web services.

Click **Import** to use the Import Wizard to import an existing XML file.

Driver Files page

Use this page to add JAR files that contain driver information.

Generate Tables from Entities wizard

Use the Generate Tables from Entities Wizard to quickly create DDL scripts from your persistent entities. Dali automatically creates the necessary primary and foreign keys, based on the entity mappings.

WARNING: Generating tables will **DROP** any existing tables and **CREATE** new tables, based on the entities in your project.

The [Generate Tables from Entities wizard](#) consists of the [Schema Generation](#) page.

Schema Generation

This table lists the properties of the Schema Generation page of the [Generate Tables from Entities wizard](#).

Property	Description	Default
Generation Output Mode	Specify how Dali should generate the DDL: <ul style="list-style-type: none"> ■ Database – DDL will be generated and written to the database only. ■ SQL-script – DDL will be generated and written to a file only. ■ Both – DDL will be generated and written to both the database and a file. 	Database

Create New Association wizard

Use the Create New Association wizard to specify association tables when generating an entity.

The wizard consists of the following pages:

- [Association Tables](#)
- [Join Columns](#)
- [Association Cardinality](#)

Association Tables

Use this page to specify the association tables for an entity.

Property	Description
Association kind	Specify if the association is Simple (1:M) or Many to Many (M:M).
Association tables	Click Table Selection , then select the two tables to associate. When creating a Many to Many association, you can select a Join Table for the association.

Join Columns

Use this dialog to specify the join columns of an association table.

Click Add to specify the join columns between the two tables.

Association Cardinality

Use this dialog to specify cardinality of an association table. Depending on the **Association Kind** and **Join Columns** that you selected previously, some associations may not be available.

- Many to one
- One to many
- One to one
- Many to many

Property pages

This section includes information each property page in the following views:

- [JPA Details view \(for entities\)](#)
- [JPA Details view \(for attributes\)](#)
- [JPA Details view \(for orm.xml\)](#)
- [JPA Structure view](#)
- [persistence.xml Editor](#)

JPA Details view (for entities)

The JPA Details view displays the persistence information for the currently selected entity and contains the following tabs:

Entity Type

Clicking the name of the mapping type, which is represented as a hyperlink, invokes the Mapping Type Selection dialog. Use this dialog to specify the type of entity: Mapped Superclass, Embeddable or the default mapping type.

- [Entity](#)
- [Embeddable](#)
- [Mapped Superclass](#)

Additional Information

Depending on the entity type, the following additional areas will be available:

- [Caching](#)
- [Queries](#)
- [Inheritance](#)
- [Attribute Overrides](#)
- [Multitenancy](#)
- [Primary Key Generation](#)

- [Converters](#)
- [Secondary tables](#)
- [Advanced](#)

Entity

This table lists the Entity information fields available in the JPA Details view for an [Entity](#).

Property	Description	Default
Table	The default database table information for this entity. These fields can be overridden by the information in the Attribute Overrides area.	
Name	The name of the primary database table associated with the entity.	
Catalog	The database catalog that contains the Table .	As defined in <code>orm.xml</code> .
Schema	The database schema that contains the Table .	As defined in <code>orm.xml</code> .
Name	The name of this entity. By default, the class name is used as the entity name.	
Access	Specify how the entity its access instance variables. <ul style="list-style-type: none"> ■ Property – Persistent state accessed through the property accessor methods. The property accessor methods must be public or private. ■ Field – Instance variables are accessed directly. All non-transient instance variables are persistent. <p>Note: This field is for display only, based on the properties in the <code>orm.xml</code>: If only the methods of the class are annotated, property access type is used. In all other cases, field access type is used.</p>	Field
ID class	Click Browse and select the primary key for the entity. Clicking the field name, which is represented as a hyperlink, allows you to create a new class.	

Embeddable

This table lists the Embeddable information fields available in the JPA Details view for [Embeddable](#) entity type.

Property	Description	Default
Access	Specify how the entity its access instance variables. <ul style="list-style-type: none"> ■ Property – Persistent state accessed through the property accessor methods. The property accessor methods must be public or private. ■ Field – Instance variables are accessed directly. All non-transient instance variables are persistent. <p>Note: This field is for display only, based on the properties in the <code>orm.xml</code>: If only the methods of the class are annotated, property access type is used. In all other cases, field access type is used.</p>	Field

Mapped Superclass

This table lists the Embeddable information fields available in the JPA Details view for [Mapped superclass](#) entity type.

Property	Description	Default
Access	<p>Specify how the entity its access instance variables.</p> <ul style="list-style-type: none"> Property – Persistent state accessed through the property accessor methods. The property accessor methods must be public or private. Field – Instance variables are accessed directly. All non-transient instance variables are persistent. <p>Note: This field is for display only, based on the properties in the <code>orm.xml</code>: If only the methods of the class are annotated, property access type is used. In all other cases, field access type is used.</p>	Field
ID class	Click Browse and select the primary key for the entity. Clicking the field name, which is represented as a hyperlink, allows you to create a new class.	

Caching

This table lists the Caching information fields available in the JPA Details view for each entity type.

Property	Description	Default	Available for Entity Type
Cachable	<p>Specifies if the entity is cachable.</p> <p>This field corresponds to the <code>@Cachable</code> annotation.</p>	True	Entity and Mapped superclass

Property	Description	Default	Available for Entity Type
Type	<p>Select one of the following as the Default Cache Type:</p> <ul style="list-style-type: none"> Weak with Soft Subcache—This option is similar to Weak with Hard Subcache except that it maintains a most frequently used subcache that uses soft references. The size of the subcache is proportional to the size of the identity map. The subcache uses soft references to ensure that these objects are garbage-collected only if the system is low on memory. Use this identity map in most circumstances as a means to control memory used by the cache. Weak with Hard Subcache—This option is similar to Soft with Weak subcache except that it maintains a most frequently used subcache that uses hard references. Use this identity map if soft references are not suitable for your platform. Weak—This option is similar to Full, except that objects are referenced using weak references. This option uses less memory than Full, allows complete garbage collection and provides full caching and guaranteed identity. Use this identity map for transactions that, once started, stay on the server side. Soft—This option is similar to Weak except that the map holds the objects using soft references. This identity map enables full garbage collection when memory is low. It provides full caching and guaranteed identity. Full—This option provides full caching and guaranteed identity: all objects are cached and not removed. Note: This process may be memory-intensive when many objects are read. None—This option does not preserve object identity and does not cache objects. This option is not recommended. 	Weak with Soft Subcache	Entity and Mapped superclass
Size	Defines the size of cache to use (number of objects).	100	Entity and Mapped superclass
Advanced			Entity and Mapped superclass
Expiry	Enables the expiration of the cached instance after a fixed period of time (milliseconds). Queries executed against the cache after this will be forced back to the database for a refreshed copy.	No expiry	Entity and Mapped superclass
Always refresh	Specifies if all queries that go to the database should always refresh the cache.	False	Entity and Mapped superclass

Property	Description	Default	Available for Entity Type
Refresh only if newer	Specifies if all queries that go to the database should refresh the cache only if the data received from the database by a query is newer than the data in the cache (as determined by the optimistic locking field). Notes: <ul style="list-style-type: none"> This option only applies if one of the other refreshing options, such as <code>alwaysRefresh</code>, is already enabled. A version field is necessary to apply this feature. 	False	Entity and Mapped superclass
Disable hits	Specifies if all queries should bypass the cache for hits, but still resolve against the cache for identity. This forces all queries to hit the database.	False	Entity and Mapped superclass
Coordination type	Specify the cache coordination mode: <ul style="list-style-type: none"> Send Object Changes Invalidate Changed Objects Send New Objects with Changes None 	Send Object Changes	Entity and Mapped superclass
Existence checking	Specify how Dali should check to determine if an entity is new or exists. <ul style="list-style-type: none"> Check Cache – If the object’s primary key does not include null and it is in the cache, then it must exist. Check Cache then Database – Perform a "does exist check" on the database. Assume Existence – If the object’s primary key does not include null then it must exist. You may use this option if the application guarantees or does not care about the existence check. Assume Non-existence – Assume that the object does not exist. You may use this option if the application guarantees or does not care about the existence check. This will always force an INSERT operation. 	Check Cache then Database	Entity and Mapped superclass

Queries

Use the queries area of the JPA Details view to create named queries and named native queries. Refer to "[Creating queries](#)" on page 3-33 for additional information.

Property	Description	Default	Available for Entity Type
Queries	Displays the existing Named and Native queries. Click Add to add a named or named native query by using the Add Query dialog .		Entity and Mapped superclass
Named Queries			
Name	Name of the query.		Entity and Mapped superclass
Query	The query SQL.		Entity and Mapped superclass

Property	Description	Default	Available for Entity Type
Lock mode	Specify the JPA locking policy. <ul style="list-style-type: none"> None Optimistic Optimistic Force Increment Pessimistic Force Increment Pessimistic Read Pessimistic Write Read Write 	None	Entity and Mapped superclass
Native Queries			
Name	Name of the query.		Entity and Mapped superclass
Result class	The class of the result. Note: This field appears for Native Named Queries only.		Entity and Mapped superclass
Query	The query SQL.		Entity and Mapped superclass
Query hints	Displays the existing query hints (Name and Value). Click Add to add a new query hint.		Entity and Mapped superclass

Inheritance

This table lists the fields available on the Inheritance area in the JPA Details view for each entity type.

Property	Description	Default	Available for Entity Type
Strategy	Specify the strategy to use when mapping a class or class hierarchy: <ul style="list-style-type: none"> Single table – All classes in the hierarchy are mapped to a single table. Joined – The root of the hierarchy is mapped to a single table; each child maps to its own table. Table per class – Each class is mapped to a separate table. This field corresponds to the @Inheritance annotation.	Single table	Entity
Discriminator Value	Specify the discriminator value used to differentiate an entity in this inheritance hierarchy. The value must conform to the specified Discriminator Type .		Entity
Discriminator Column	These fields are available when using a Single or Joined inheritance strategy. This field corresponds to the @DiscriminatorColumn annotation. Use the Details area to define the Length and Column definition of this Discriminator Column.		
Name	Name of the discriminator column		Entity

Property	Description	Default	Available for Entity Type
Type	Set this field to set the discriminator type to Char or Integer (instead of its default: String). The Discriminator Value must conform to this type.	String	Entity
Length	The column length for String-based discriminator types.	0	Entity
Column definition	The SQL fragment that is used when generating the DDL for the discriminator column.		Entity
Primary Key Join Columns	Use to override the default primary key join columns. Select Override Default , then click Add to select new Join Column with the Add Primary Key Join Column dialog . This field corresponds with @PrimaryKeyJoinColumn annotation.		Entity

Refer to "[Specifying entity inheritance](#)" on page 3-32 for additional information.

Attribute Overrides

Use the Attribute Overrides area in the JPA Details view to override the default settings specified in the [Entity](#) area of an attribute. Attribute overrides generally override/configure attributes that are inherited or embedded.

This table lists the Attribute override fields available in the JPA Details view for each entity type.

Property	Description	Default	Available for Entity Type
Attribute Overrides	Specify a property or field to be overridden (from the default mappings). Select Override Default .		Entity
Join Columns			Entity

Multitenancy

Use the Multitenancy area in the JPA Details view to specify that a given entity is shared among multiple tenants of an application.

Property	Description	Default	Available for Entity Type
Multitenancy strategy	Specify the multitenant strategy to use: <ul style="list-style-type: none"> ■ Single table ■ Table per tenant ■ VPD 	Single table	Entity and Mapped superclass
Include criteria	Specify if the database requires the tenant criteria to be added to the SELECT, UPDATE, and DELETE queries.	True	Entity and Mapped superclass
Tenant descriptor columns	Use to limit what a persistence context can access in single-table multitenancy		Entity and Mapped superclass
Override default			

Property	Description	Default	Available for Entity Type
Name	The name of column to be used for the tenant discriminator.	eclipselink.tenant-id	
Table	The name of the table that contains the column.		
Context property	The name of the context property to apply to the tenant discriminator column.		
Discriminator type	The type of object/column to use as a class discriminator: <ul style="list-style-type: none"> Character Integer String 	String	
Length	The column length for String-based discriminator types.		
Column definition	The SQL fragment that is used when generating the DDL for the discriminator column.		
Primary key	Specifies that the tenant discriminator column is part of the primary key of the tables.	False	

Primary Key Generation

Use the Primary Key Generation area in the JPA Details view to specify how to generate a primary key for a given entity.

Property	Description	Default	Available for Entity Type
Table Generator	These fields define the database table used for generating the primary key and correspond to the <code>@TableGenerator</code> annotation.		Entity
Name	Unique name of the generator.		Entity
Table	Database table that stores the generated ID values.		Entity
Schema	Database schema of the Table .		Entity
Catalog	Database catalog of the Table .		Entity
Primary key column	The column in the table generator's Table that contains the primary key.		Entity
Value column	The column that stores the generated ID values.		Entity
Allocation size	The value for the Primary Key Column in the generator table.	50	Entity
Initial value	The starting value of the generated primary key.	0	Entity
Sequence Generator	These fields define the specific sequence used for generating the primary key and correspond to the <code>@SequenceGenerator</code> annotation. These fields apply only when Strategy = Sequence .		Entity
Name	Name of the sequence table to use for defining primary key values.		Entity
Sequence	Unique name of the sequence.		Entity

Property	Description	Default	Available for Entity Type
Schema	Database schema of the Sequence .		Entity
Catalog	Database catalog of the Sequence .		Entity
Allocation size	These fields define the specific sequence used for generating the primary key and correspond to the <code>@SequenceGenerator</code> annotation. These fields apply only when Strategy = Sequence .	50	Entity
Initial value	The starting value of the generated primary key.	1	Entity

Secondary tables

Use the Secondary Tables area in the JPA Details view to associate additional tables with an entity. Use this area if the data associated with an entity is spread across multiple tables.

Refer to "[Specifying additional tables](#)" on page 3-32 for additional information.

Converters

Use the Converter area in the JPA Details view to specify a way to modify data value(s) during the reading and writing of a mapped attribute.

Property	Description	Default	Available for Entity Type
Converters	Click Add and use the Add Converter dialog to create a new converter. <ul style="list-style-type: none"> ■ Object Type – converts a fixed number of database data value(s) to Java object value(s) ■ Type – modifies data values ■ Struct – Enable custom processing of <code>java.sql.Struct</code> types ■ Custom 		Entity , Embeddable , and Mapped superclass
Name	The <code>String</code> name for your converter, must be unique across the persistence unit		Entity , Embeddable , and Mapped superclass
Class	The class of your converter. This class must implement the <code>org.eclipse.persistence.mappings.converters.Converter</code> interface. Appears for Custom and Struct converters only.		Entity , Embeddable , and Mapped superclass
Data type	The type stored in the database. Appears for Object Type and Type converters only.		Entity , Embeddable , and Mapped superclass
Object type	The type stored on the entity. Appears for Object Type and Type converters only.		Entity , Embeddable , and Mapped superclass
Conversion values	The array of conversion values (instances of <code>ConversionValue: String objectValue</code> and <code>String dataValue</code>). Appears for Object Type converters only.		Entity , Embeddable , and Mapped superclass

Property	Description	Default	Available for Entity Type
Default object value	Set the value of this attribute to the default object value. Note that this argument is for dealing with legacy data if the data value is missing. Appears for Object Type converters only.		Entity , Embeddable , and Mapped superclass

Advanced

Use the Advanced area in the JPA Details view to configure additional settings for an entity.

Property	Description	Default	Available for Entity Type
Read-only	Specifies if a class is read-only.	False	Entity and Mapped superclass
Customizer class	Specifies a class that implements <code>DescriptorCustomizer</code> and is to run against an entity's class descriptor after all metadata processing has been completed.	True	Entity , Embeddable , and Mapped superclass
Change tracking	Specifies the <code>ObjectChangePolicy</code> to use: <ul style="list-style-type: none"> ▪ Attribute – Objects with changed attributes will be processed in the commit process to include any changes in the results of the commit. Unchanged objects will be ignored. ▪ Object – Changed objects will be processed in the commit process to include any changes in the results of the commit. Unchanged objects will be ignored. ▪ Deferred – Defers all change detection to the <code>UnitOfWork</code>'s change detection process. ▪ Auto – Does not set any change tracking policy; change tracking will be determined at runtime. 	Auto	Entity , Embeddable , and Mapped superclass

JPA Details view (for attributes)

The JPA Details view displays the persistence information for the currently selected mapped attribute and contains the following areas:

Mapping Type

- [Basic Mapping](#)
- [Element Collection Mapping](#)
- [Embedded Mapping](#)
- [Embedded ID Mapping](#)
- [ID Mapping](#)
- [Many-to-Many Mapping](#)
- [Many-to-One Mapping](#)
- [One-to-Many Mapping](#)
- [One-to-One Mapping](#)

- [Version Mapping](#)

Additional Information

Depending on the mapping type, the following additional areas will be available:

- [Value](#)
- [Type information](#)
- [Converters](#)
- [Ordering](#)
- [Joining Strategy](#)
- [Derived Identity](#)
- [Primary Key Generation information](#)

Basic Mapping

Property	Description	Default
Column	The database column that contains the value for the attribute. This field corresponds to the @Column annotation.	By default, the Column is assumed to be named identically to the attribute and always included in the INSERT and UPDATE statements.
Name	The database column that contains the value for the attribute.	
Table	Name of the database table that contains the selected column.	
Details		
Insertable	Specifies if the column is always included in SQL INSERT statements.	True
Updatable	Specifies if this column is always included in SQL UPDATE statements.	True
Unique	Sets the UNIQUE constraint for the column.	False
Nullable	Specifies if the column allows null values.	True
Length	Sets the column length.	255
Precision	Sets the precision for the column values.	0
Scale	Sets the number of digits that appear to the right of the decimal point.	0
Column definition	The SQL fragment that is used when generating the DDL for the column.	
Fetch	Defines how data is loaded from the database: <ul style="list-style-type: none"> ▪ Eager – Data is loaded in before it is actually needed. ▪ Lazy – Data is loaded only when required by the transaction. 	Eager
Optional	Specifies if this field is can be null.	True

Property	Description	Default
Mutable	Specify if the value of a complex field type can be changed (or not changed) instead of being replaced.	True

Basic mappings also include the following areas:

- [Type information](#)
- [Converters](#)

Element Collection Mapping

Property	Description	Default
Target class	The class (basic or embeddable) that is the element type of the collection.	
Fetch	Defines how data is loaded from the database: <ul style="list-style-type: none"> ■ Eager – Data is loaded in before it is actually needed. ■ Lazy – Data is loaded only when required by the transaction. 	Lazy
Join fetch	The type of fetch to use: <ul style="list-style-type: none"> ■ Inner – Provides the inner join fetching of the related object. Note: Inner joining does not allow for null or empty values ■ Outer – Provides the outer join fetching of the related object Note: Outer joining allows for null or empty values. 	None

Collection Table

Name	Name of the database table used for the mapping.
Schema	Database schema of the Collection Table .
Catalog	Database Catalog of the Collection Table .

Join Columns

Override default	Click Add to create a new join column, by using the Add Join Column dialog .	0
------------------	---	---

Element collection mappings also include the following areas:

- [Value](#)
- [Converters](#)
- [Ordering](#)

Embedded Mapping

Embedded ID Mapping

ID Mapping

Property	Description	Default
Column	The database column that contains the value for the attribute. This field corresponds to the @Column annotation.	By default, the Column is assumed to be named identically to the attribute.
Name	The database column that contains the value for the attribute.	
Table	Name of the database table that contains the selected column.	
Details		
Insertable	Specifies if the column is always included in SQL INSERT statements.	True
Updatable	Specifies if this column is always included in SQL UPDATE statements.	True
Unique	Sets the UNIQUE constraint for the column.	False
Nullable	Specifies if the column allows null values.	True
Length	Sets the column length.	255
Precision	Sets the precision for the column values.	0
Scale	Sets the number of digits that appear to the right of the decimal point.	0
Column definition	The SQL fragment that is used when generating the DDL for the column.	
Mutable	Specify if the value of a complex field type can be changed (or not changed) instead of being replaced.	True

ID mappings also include the following areas:

- [Type information](#)
- [Converters](#)
- [Primary Key Generation information](#)

Many-to-Many Mapping

Property	Description	Default
Target entity	The persistent entity to which the attribute is mapped.	

Property	Description	Default
Fetch	Defines how data is loaded from the database: <ul style="list-style-type: none"> ▪ Eager – Data is loaded in before it is actually needed. ▪ Lazy – Data is loaded only when required by the transaction. 	Lazy
Join fetch	The type of fetch to use: <ul style="list-style-type: none"> ▪ Inner – Provides the inner join fetching of the related object. Note: Inner joining does not allow for null or empty values ▪ Outer – Provides the outer join fetching of the related object Note: Outer joining allows for null or empty values. 	None
Cascade	Specify which operations are propagated throughout the entity. <ul style="list-style-type: none"> ▪ All – All operations ▪ Persist ▪ Merge ▪ Move ▪ Remove ▪ Refresh 	True

One-to-many mappings also include the following areas:

- [Joining Strategy](#)
- [Converters](#)
- [Ordering](#)

Many-to-One Mapping

Property	Description	Default
Target entity	The persistent entity to which the attribute is mapped. You do not need to explicitly specify the target entity, since it can be inferred from the type of object being referenced.	null
Fetch	Defines how data is loaded from the database: <ul style="list-style-type: none"> ▪ Eager – Data is loaded in before it is actually needed. ▪ Lazy – Data is loaded only when required by the transaction. 	Lazy

Property	Description	Default
Join fetch	<p>The type of fetch to use:</p> <ul style="list-style-type: none"> Inner – Provides the inner join fetching of the related object. Note: Inner joining does not allow for null or empty values Outer – Provides the outer join fetching of the related object Note: Outer joining allows for null or empty values. 	
Optional	Specifies if this field is can be null.	True
Cascade	<p>Specify which operations are propagated throughout the entity.</p> <ul style="list-style-type: none"> All – All operations Persist Merge Move Remove Refresh 	True

Many-to-one mappings also include the following areas:

- [Joining Strategy](#)
- [Derived Identity](#)

One-to-Many Mapping

Property	Description	Default
Target entity	The persistent entity to which the attribute is mapped.	
Fetch	<p>Defines how data is loaded from the database:</p> <ul style="list-style-type: none"> Eager – Data is loaded in before it is actually needed. Lazy – Data is loaded only when required by the transaction. 	Lazy
Join fetch	<p>The type of fetch to use:</p> <ul style="list-style-type: none"> Inner – Provides the inner join fetching of the related object. Note: Inner joining does not allow for null or empty values Outer – Provides the outer join fetching of the related object Note: Outer joining allows for null or empty values. 	

Property	Description	Default
Private owned	Specify that a relationship is privately owned; target object is a dependent part of the source object and is not referenced by any other object and cannot exist on its own.	
Orphan removal		False
Cascade	Specify which operations are propagated throughout the entity. <ul style="list-style-type: none"> ▪ All – All operations ▪ Persist ▪ Merge ▪ Move ▪ Remove ▪ Refresh 	True

One-to-many mappings also include the following areas:

- [Joining Strategy](#)
- [Converters](#)
- [Ordering](#)

One-to-One Mapping

Property	Description	Default
Target entity	The persistent entity to which the attribute is mapped.	
Fetch	Defines how data is loaded from the database: <ul style="list-style-type: none"> ▪ Eager – Data is loaded in before it is actually needed. ▪ Lazy – Data is loaded only when required by the transaction. 	Eager
Join fetch	The type of fetch to use: <ul style="list-style-type: none"> ▪ Inner – Provides the inner join fetching of the related object. Note: Inner joining does not allow for null or empty values ▪ Outer – Provides the outer join fetching of the related object Note: Outer joining allows for null or empty values. 	
Optional	Specifies if this field is can be null.	True
Private owned	Specify that a relationship is privately owned; target object is a dependent part of the source object and is not referenced by any other object and cannot exist on its own.	False
Orphan removal		False

Property	Description	Default
Cascade	Specify which operations are propagated throughout the entity. <ul style="list-style-type: none"> ▪ All – All operations ▪ Persist ▪ Merge ▪ Move ▪ Remove ▪ Refresh 	True

One-to-one mappings also include the following areas:

- [Derived Identity](#)
- [Joining Strategy](#)

Version Mapping

Property	Description	Default
Column	The database column that contains the value for the attribute. This field corresponds to the @Column annotation.	By default, the Column is assumed to be named identically to the attribute.
Name	The database column mapped to the entity attribute. By default, the Column is assumed to be named identically to the attribute and always included in the INSERT and UPDATE statements.	
Table	Name of the database table that contains the selected column.	
Details		
Insertable	Specifies if the column is always included in SQL INSERT statements.	True
Updatable	Specifies if this column is always included in SQL UPDATE statements.	True
Unique	Sets the UNIQUE constraint for the column.	False
Nullable	Specifies if the column allows null values.	True
Length	Sets the column length.	255
Precision	Sets the precision for the column values.	0
Scale	Sets the number of digits that appear to the right of the decimal point.	0
Column definition	The SQL fragment that is used when generating the DDL for the column.	
Mutable	Specify if the value of a complex field type can be changed (or not changed) instead of being replaced.	True

Version mappings also include the following areas:

- [Type information](#)
- [Converters](#)

Type information

Property	Description	Available for Mapping Type
Default		Basic mapping, ID mapping, and Version mapping
LOB	Specify if the field is mapped to <code>java.sql.Clob</code> or <code>java.sql.Blob</code> . This field corresponds to the <code>@Lob</code> annotation.	Basic mapping
Temporal	Specifies if this field is one of the following: <ul style="list-style-type: none"> ■ Date – <code>java.sql.Date</code> ■ Time – <code>java.sql.Time</code> ■ Timestamp – <code>java.sql.Timestamp</code> This field corresponds to the <code>@Temporal</code> annotation.	Basic mapping, ID mapping, and Version mapping
Enumerated	Specify how to persist enumerated constraints if the <code>String</code> value suits your application requirements or to match an existing database schema. <ul style="list-style-type: none"> ■ <code>ordinal</code> ■ <code>String</code> This field corresponds to the <code>@Enumerated</code> annotation.	Basic mapping
Converted	Converter name: <ul style="list-style-type: none"> ■ None (default) – ■ <code>class-instance</code> – ■ <code>serialized</code> – 	Basic mapping, ID mapping, and Version mapping

Value

Converters

Use this area to specify a custom converter for modification of the data value(s) during the reading and writing of a mapped attribute.

Property	Description	Default	Available for Entity Type
Converters	Click Add and use the Add Converter dialog to create a new converter. <ul style="list-style-type: none"> Object Type Type Struct Custom 		Basic mapping , Element collection mapping , ID mapping , Many-to-many mapping , One-to-many mapping , and Version mapping
Name	Name of the converter.		
Class	Appears for Custom and Struct converters only.		
Data type	Appears for Object Type and Type converters only.		
Object type	Appears for Object Type and Type converters only.		
Conversion values	Appears for Object Type converters only.		
Default object value	Appears for Object Type converters only.		

Ordering

Specify the default order for objects returned from a query. These options correspond to the `@OrderBy` annotation.

Property	Description	Default	Available for Entity Type
Name			Element collection mapping , Many-to-many mapping , and One-to-many mapping
Primary key			
Custom			
Order column			
Insertable	Specify if the column is included in SQL INSERT statements	True	
Updatable	Specify if the column is included in SQL UPDATE statements	True	
Nullable	Specify if the database column is nullable	True	
Column definition	The SQL fragment that used to generate the DDL for the column	Generated SQL to create a column of the inferred type	

Joining Strategy

Use this area to specify a mapped column for joining an entity association. By default, the mapping is assumed to have a single join.

Property	Description	Available for Mapping Type
Mapped by	The field in the database table that "owns" the relationship. This field is required only on the non-owning side of the relationship.	Many-to-many mapping, One-to-many mapping, and One-to-one mapping
Attribute		
Primary key join columns		One-to-one mapping
Join columns	By default, the name is assumed to be the primary tables associated with the entities concatenated with an underscore. Select Override Default , then Add, Edit, or Remove the join columns.	Many-to-one mapping, One-to-many mapping, and One-to-one mapping
Join table		Many-to-many mapping, Many-to-one mapping, One-to-many mapping, and One-to-one mapping
Name	Name of the join table that contains the foreign key column. You must specify the join table on the owning side. By default, the name is assumed to be the primary tables associated with the entities concatenated with an underscore.	
Schema	Schema of the table	
Catalog	Catalog of the table	
Join columns	The foreign key columns of the join table which reference the primary table of the entity <i>owning</i> the association. (that is, the <i>owning</i> side of the association).	
Inverse join columns	The foreign key columns of the join table which reference the primary table of the entity that <i>does not</i> own the association. (that is, the <i>inverse</i> side of the association).	

Derived Identity

Property	Description	Available for Mapping Type
None		Many-to-one mapping and One-to-one mapping

Property	Description	Available for Mapping Type
ID		Many-to-one mapping and One-to-one mapping
Maps ID		Many-to-one mapping and One-to-one mapping

Primary Key Generation information

This table lists the fields available in the Primary Key Generation area in JPA Details view for [ID mapping](#) types.

Property	Description	Default
Primary Key Generation	These fields define how the primary key is generated. These fields correspond to the <code>@GeneratedValue</code> annotation.	Generated Value
Strategy	<ul style="list-style-type: none"> ■ Auto ■ Identity – Values are assigned by the database’s Identity column. ■ Sequence – Values are assigned by a sequence table (see Sequence Generator). ■ Table – Values are assigned by a database table (see Table Generator). 	Auto
Generator Name	Unique name of the generated value.	
Table Generator	These fields define the database table used for generating the primary key and correspond to the <code>@TableGenerator</code> annotation. These fields apply only when Strategy = Table .	
Name	Unique name of the generator.	
Table	Database table that stores the generated ID values.	
Schema	Schema of the table	
Catalog	Catalog of the table	
Primary Key Column	The column in the table generator’s Table that contains the primary key.	
Value Column	The column that stores the generated ID values.	
Primary Key Column Value	The value for the Primary Key Column in the generator table.	
Allocation size		

Property	Description	Default
Initial value		
Sequence Generator	<p>These fields define the specific sequence used for generating the primary key and correspond to the <code>@SequenceGenerator</code> annotation.</p> <p>These fields apply only when Strategy = Sequence.</p>	
Name	Name of the sequence table to use for defining primary key values.	
Sequence	Unique name of the sequence.	
Schema		
Catalog		
Allocation size		
Initial value		

JPA Details view (for orm.xml)

The JPA Details view displays the default mapping and persistence information for the project and contains the following areas:

- [Entity Mappings](#)
- [Persistence Unit](#)
- [Generators](#)
- [Queries](#)
- [Converters](#) (when using EclipseLink)

These defaults can be overridden by the settings on a specific entity or mapping.

Entity Mappings

This table lists the Entity Mappings fields available in the JPA Details view for each entity type.

Property	Description	Default
Package	The Java package that contains the persistent entities. Click Browse and select the package	
Schema	<p>The database schema that contains the Table.</p> <p>This field corresponds to the <code><schema></code> element in the <code>orm.xml</code> file.</p>	
Catalog	<p>The database catalog that contains the Table.</p> <p>This field corresponds to the <code><catalog></code> element in the <code>orm.xml</code> file.</p>	
Access	<p>Specify the default access method for the variables in the project:</p> <ul style="list-style-type: none"> ■ Property ■ Field <p>This field corresponds to the <code><access></code> element in the <code>orm.xml</code> file.</p>	

Persistence Unit

This table lists the Persistence Unit information fields available in the JPA Details view for each entity type. These fields are contained in the `<persistence-unit-metadata>` element in the `orm.xml` file.

Property	Description	Default
XML Mapping Data Complete	Specifies that the Java classes in this persistence unit are fully specified by their metadata. Any annotations will be ignored. This field corresponds to the <code><xml-mapping-metadata-complete></code> element in the <code>orm.xml</code> file.	
Cascade Persist	Adds <code>cascade-persist</code> to the set of cascade options in entity relationships of the persistence unit. This field corresponds to the <code><cascade-persist></code> element in the <code>orm.xml</code> file.	
Schema	The database schema that contains the Table . This field corresponds to the <code><schema></code> element in the <code>orm.xml</code> file.	
Catalog	The database catalog that contains the Table . This field corresponds to the <code><catalog></code> element in the <code>orm.xml</code> file.	
Access	Specify how the entity its access instance variables. <ul style="list-style-type: none"> ▪ Property – Persistent state accessed through the property accessor methods. The property accessor methods must be public or private. ▪ Field – Instance variables are accessed directly. All non-transient instance variables are persistent. 	
Delimited identifiers		

Generators

This table lists the Generator fields available in the JPA Details view for the `orm.xml` file.

Property	Description	Default
Generators	Click Add to create a new table or sequence generator, or select an existing generator and click Remove , to add or remove a generator.	
Sequence Generators		
Name		
Sequence		
Schema		
Catalog		
Allocation size		50
Initial value		1

Property	Description	Default
Table Generators		
Name		
Table		
Schema		
Catalog		
Primary key column value		
Allocation size		50
Initial value		1

Queries

This table lists the Query information fields available in the JPA Details view for the `orm.xml` file.

Property	Description	Default
Queries	Displays the existing Named and Native queries. Click Add to add a named query, or Add Native for a native query. For named queries, enter the query in the Query field. For native queries, select a result class, then enter the query in the Query field.	
Named Queries		
Name		
Query		
Lock mode		None
Native Queries		
Name		
Result class		
Query		
Query Hints	Displays the existing query hints (Name and Value). Click Add to add a new query hint.	

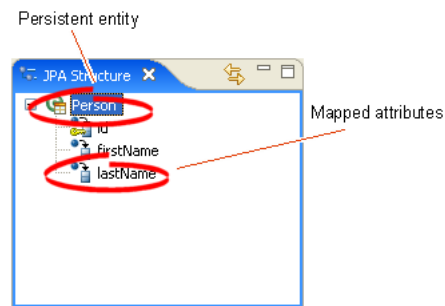
Converters

The Converters information in the JPA Details view applies only when using EclipseLink.

Click **Add** to create a new converter, using the [Add Converter dialog](#).

JPA Structure view

The JPA Structure view displays an outline of the structure (its attributes and mappings) of the entity that is currently selected or opened in the editor. The structural elements shown in the outline are the entity and its fields.

Figure 4–1 Sample JPA Structure View

persistence.xml Editor

The persistence.xml Editor provides an interface that enables you to update the persistence.xml file. For projects using the EclipseLink platform, the persistence.xml Editor consists of the following pages:

- [General](#)
- [Connection](#)
- [Customization](#)
- [Caching](#)
- [Logging](#)
- [Options](#)
- [Schema Generation](#)
- [Properties](#)
- [Source](#)

For projects using the Generic platform, the following subset of these pages is available:

- [General](#)
- [Connection](#)
- [Properties](#)
- [Source](#)

General

The following table lists properties available in the General page of the [persistence.xml Editor](#).

Property	Description	Default
General		
Name	Enter the name of the persistence unit. Defines the <persistence-unit name> element.	The project name
Persistence provider	Enter the name of the persistence provider. Defines the <provider> element.	Determined by the server

Property	Description	Default
Description	Enter a description for this persistence unit. This is an optional property. Defines the <description> element.	
Managed Classes		
Classes	Click Add to add a new class, or select an existing class and click Remove , to add or remove the classes managed through the persistence unit. Defines the <class> element. Select a class and click Open to modify the class in the editor.	
Exclude Unlisted Classes	Select to include all annotated entity classes in the root of the persistence unit. Defines the <exclude-unlisted-classes> element.	False
XML Mapping Files		
Files	Click Add to select an XML mapping file, or select an existing file and click Remove , to add or remove an object/relational mapping XML files that define the classes to be managed by the persistence unit. Defines the <mapping-file> element.	Meta-INF\orm.xml
Exclude default EclipseLink XML mapping file	Select to include all annotated EclipseLink mapping files. Defines the eclipselink.exclude-eclipselink-orm property element. Note: This field applies only when using EclipseLink JPA implementation	False
JAR Files		
Files	Click Add to select a JAR file, or select an existing file and click Remove , to add or remove JAR files and libraries in the persistence unit.	

Connection

The following table lists the properties available in the Connection page of the [persistence.xml Editor](#).

Property	Description	Default
Transaction type	Specify if the connection for this persistence unit uses one of the following transaction types: <ul style="list-style-type: none"> ▪ Default -- Select to use the container used by the container. ▪ JTA (Java Transaction API) -- Transactions of the Java EE server. ▪ Resource Local -- Native actions of a JDBC driver that are referenced by a persistence unit. 	JTA

Property	Description	Default
Batch writing	<p>Specify the use of batch writing to optimize transactions with multiple write operations.</p> <p>Set the value of this property into the session at deployment time.</p> <p>Note: This property applies when used both in a Java SE and Java EE environment.</p> <p>The following are the valid values for <code>oracle.toplink.config.BatchWriting</code>:</p> <ul style="list-style-type: none"> ▪ JDBC–Use JDBC batch writing. ▪ Buffered–Do not use either JDBC batch writing nor native platform batch writing. ▪ OracleJDBC–Use both JDBC batch writing and Oracle native platform batch writing. ▪ None–Do not use batch writing (turn it off). 	None
Statement caching	<p>Specify if the query caches its JDBC statement. If enabled, you can also set the number of statements to cache.</p>	50
Native SQL	<p>Specify if Dali includes platform-specific (that is, "native") SQL statements. If false, Dali uses generic SQL.</p>	False
Database		
JTA Data Source Name	<p>If you selected JTA as the transaction type, then enter the name of the default JTA data source for the persistence unit.</p>	
Non-JTA Data Source Name	<p>If you selected Resource Local as the transaction type, then enter the name of the non-JTA data source.</p> <p>This property is not available for projects using the Generic platform.</p>	
EclipseLink connection pool	<p>Define the connection pool driver, URL, user name and password.</p> <p>These properties are not available for projects using the Generic platform.</p>	
Bind parameters	<p>Control whether or not the query uses parameter binding.</p> <p>Note: This property applies when used in a Java SE environment.</p> <p>This property is not available for projects using the Generic platform.</p>	True
Read Connection	<p>The maximum and minimum number of connections allowed in the JDBC read connection pool.</p> <p>Note: These property apply when used in a Java SE environment.</p> <p>These properties are not available for projects using the Generic platform</p>	Minimum: 2 Maximum: 2

Property	Description	Default
Write Connection	The maximum and minimum number of connections allowed in the JDBC read connection pool. Note: These property apply when used in a Java SE environment. These properties are not available for projects using the Generic platform	Minimum: 5 Maximum 10
Exclusive connections	These fields are available only when Transaction Type is Local Resource .	
Exclusive connection mode	Specify when Dali performs reads through the write connection. <ul style="list-style-type: none"> ■ Always – Create an exclusive isolated client session if reading an isolated entity, otherwise create an exclusive client session. ■ Isolated – Create an exclusive isolated client session if reading an isolated entity, otherwise raise an error. ■ Transactional – Create an isolated client session if some or all entities require isolated cache, otherwise create a client session. 	Transactional
Lazy connection acquisition	Specify if Dali acquires write connections lazily.	True

Customization

The following table lists the properties available in the Customization page of the [persistence.xml Editor](#).

Property	Description	Default
Weaving	Specifies if weaving of the entity classes is performed. The EclipseLink JPA persistence provider uses weaving to enhance JPA entities for such properties as lazy loading, change tracking, fetch groups, and internal optimizations. Select from the following options: <ul style="list-style-type: none"> ■ No Weaving ■ Weave Dynamically ■ Weave Statically -- Use this option if you plan to execute your application outside of a Java EE 5 container in an environment that does not permit the use of <code>-javaagent:eclipselink.jar</code> on the JVM command line. This assumes that classes have already been statically woven. Run the static weaver on the classes before deploying them. 	Weave Dynamically
Lazy	Select this option to enable lazy weaving.	True

Property	Description	Default
Fetch Groups	<p>Select this option to enable fetch groups through weaving. Set this option to false if:</p> <ul style="list-style-type: none"> There is no weaving. Classes should not be changed during weaving (for example, when debugging). <p>Set this property to false for platforms where it is not supported.</p>	True
Internal	<p>Specify if Dali uses internal optimizations through weaving.</p> <p>If enabled, enables lazy one-to-one and many-to-one mappings through weaving.</p>	True
Eager	Specify if Dali uses indirection on eager relationships.	False
Change Tracking	Select this option to use weaving to detect which fields or properties of the object change.	True
Validation only	Specify if Dali should validate deployments by initializing descriptors but not connecting to the data source.	True
Mapping files schema validation		False
Throw exceptions	Select this option to set EclipseLink to throw an exception or log a warning when it encounters a problem with any of the files listed in a persistence.xml file <code><mapping-file></code> element.	True
Exception handler	Select (or create) a Java class (that implements the <code>org.eclipse.persistence.exceptions.ExceptionHandler</code> interface) to handle exceptions.	
Session Customizer	Select a session customizer class: a Java class that implements the <code>eclipselink.tools.sessionconfiguration.SessionCustomizer</code> interface and provides a default (zero-argument) constructor. Use this class' <code>customize</code> method, which takes an <code>eclipselink.sessions.Session</code> , to programmatically access advanced EclipseLink session API.	
Profiler	<p>Specify which performance profiler to use in order to capture runtime statistics.</p> <ul style="list-style-type: none"> No Profiler – Do not use a performance profiler. Performance Profiler – Use EclipseLink performance profiler (<code>org.eclipse.persistence.tools.profiler.PerformanceProfiler</code> class). Query Monitor – Monitor query executions and cache hits (<code>org.eclipse.persistence.tools.profiler.QueryMonitor</code> class). 	NoProfiler

Note: This page is not available for projects using the **Generic** platform.

Caching

This table lists the properties of the Caching page of the [persistence.xml Editor](#).

Property	Description	Default
Shared cache mode	Select one of the following as the shared cache mode: <ul style="list-style-type: none">■ All –■ None –■ Enable Selective –■ Disable Selective –■ Unspecified –	Disable selective

Property	Description	Default
Default Cache Type	<p>Select one of the following as the Default Cache Type:</p> <ul style="list-style-type: none"> <p>■ Weak with Soft Subcache—This option is similar to Weak with Hard Subcache except that it maintains a most frequently used subcache that uses soft references. The size of the subcache is proportional to the size of the identity map. The subcache uses soft references to ensure that these objects are garbage-collected only if the system is low on memory.</p> <p>Use this identity map in most circumstances as a means to control memory used by the cache.</p> <p>■ Weak with Hard Subcache—This option is similar to Soft with Weak subcache except that it maintains a most frequently used subcache that uses hard references. Use this identity map if soft references are not suitable for your platform.</p> <p>■ Weak—This option is similar to Full, except that objects are referenced using weak references. This option uses less memory than Full, allows complete garbage collection and provides full caching and guaranteed identity.</p> <p>Use this identity map for transactions that, once started, stay on the server side.</p> <p>■ Soft—This option is similar to Weak except that the map holds the objects using soft references. This identity map enables full garbage collection when memory is low. It provides full caching and guaranteed identity.</p> <p>■ Full—This option provides full caching and guaranteed identity: all objects are cached and not removed.</p> <p>Note: This process may be memory-intensive when many objects are read.</p> <p>■ None—This option does not preserve object identity and does not cache objects. This option is not recommended.</p> 	Weak with soft subcache
Default Cache Size	Set the size (maximum number of objects) of the cache.	100

Property	Description	Default
Flush clear cache	Select one of the following as the Default Cache Type: <ul style="list-style-type: none">▪ Drop – This mode is the fastest and uses the least memory. However, after commit the shared cache might potentially contain stale data.▪ Drop Invalidate – Classes that have at least one object updated or deleted are invalidated in the shared cache at commit time. This mode is slower than Drop, but as efficient memory usage-wise, and prevents stale data.▪ Merge – Drop classes from the EntityManager’s cache of objects that have not been flushed. This mode leaves the shared cache in a perfect state after commit. However, it is the least memory-efficient mode; the memory might even run out in a very large transaction.	Drop Invalidate

Note: This page is not available for projects using the **Generic** platform.

Logging

This table lists the properties of the Logging page of the [persistence.xml Editor](#).

Note: This page is not available for projects using the **Generic** platform.

Property	Description	Default
Logging Level	<p>Specifies the amount and detail of log output by selecting the log level (in ascending order of information):</p> <p>The following are the valid values for the <code>java.util.logging.Level</code>:</p> <ul style="list-style-type: none"> ■ OFF—disables logging ■ SEVERE—logs exceptions indicating TopLink cannot continue, as well as any exceptions generated during login. This includes a stack trace. ■ WARNING—logs exceptions that do not force TopLink to stop, including all exceptions not logged with severe level. This does not include a stack trace. ■ INFO—logs the login/logout per sever session, including the user name. After acquiring the session, detailed information is logged. ■ CONFIG—logs only login, JDBC connection, and database information. ■ FINE—logs SQL. ■ FINER—similar to warning. Includes stack trace. ■ FINEST—includes additional low level information. <p>Example: <code>persistence.xml</code> file</p> <pre><property name="eclipselink.logging.level" value="INFO" /></pre>	Info
Timestamp	<p>Control whether the timestamp is logged in each log entry.</p> <p>The following are the valid values:</p> <ul style="list-style-type: none"> ■ True—log a timestamp. ■ False—do not log a timestamp. <p>Example: <code>persistence.xml</code> file</p> <pre><property name="eclipselink.logging.timestamp" value="false" /></pre>	True
Thread	<p>Control whether a thread identifier is logged in each log entry.</p> <p>The following are the valid values:</p> <ul style="list-style-type: none"> ■ true—log a thread identifier. ■ false—do not log a thread identifier. 	True

Property	Description	Default
Session	<p>Control whether an EclipseLink session identifier is logged in each log entry.</p> <p>The following are the valid values:</p> <ul style="list-style-type: none"> ■ true—log a EclipseLink session identifier. ■ false—do not log a EclipseLink session identifier. <p>Example: persistence.xml file</p> <pre><property name="eclipselink.logging.session" value="false"/></pre>	true
Exceptions	<p>Control whether the exceptions thrown from within the EclipseLink code are logged prior to returning the exception to the calling application. Ensures that all exceptions are logged and not masked by the application code.</p> <p>The following are the valid values:</p> <ul style="list-style-type: none"> ■ true—log all exceptions. ■ false—do not log exceptions. <p>Example: persistence.xml file</p> <pre><property name="eclipselink.logging.exceptions" value="true"/></pre>	false
Log file	<p>Specify a file location for the log output (instead of the standard out).</p> <p>Example: persistence.xml file</p> <pre><property name="eclipselink.logging.file" value="C:\myout\" /></pre>	stdout
Logger	<p>Select the type of logger to use:</p> <p>The following are the valid values:</p> <ul style="list-style-type: none"> ■ DefaultLogger—the EclipseLink native logger eclipselink.logging.DefaultSessionLog. ■ JavaLogger—the java.util.logging logger eclipselink.logging.JavaLog. ■ ServerLogger—the java.util.logging logger eclipselink.platform.server.ServerLog. Integrates with the application server's logging as define in the eclipselink.platform.server.ServerPlatform. ■ Fully qualified class name of a custom logger. The custom logger must implement the eclipselink.logging.SessionLog interface. <p>Example: persistence.xml file</p> <pre><property name="eclipselink.logging.logger" value="acme.loggers.MyCustomLogger" /></pre>	DefaultLogger

Property	Description	Default
Logging Categories	<p>You can also specify the logging level for the following specific categories:</p> <ul style="list-style-type: none"> ▪ SQL ▪ Connection ▪ Event ▪ Query ▪ Cache ▪ Propagation ▪ EJB ▪ DMS ▪ EJB or metadata ▪ JPA metadata ▪ Weaving ▪ Properties ▪ Server 	Info

Options

This table lists the properties of the Options page of the [persistence.xml Editor](#).

Note: This page is not available for projects using the **Generic** platform.

Property	Description	Default
Session Options		
Session Name	<p>Specify the name by which the EclipseLink session is stored in the static session manager. Use this option if you need to access the EclipseLink shared session outside of the context of the JPA or to use a pre-existing EclipseLink session configured through a EclipseLink <code>sessions.xml</code> file</p> <p>Valid values: a valid EclipseLink session name that is unique in a server deployment.</p> <p>Example: <code>persistence.xml</code> file</p> <pre><property name="eclipseLink.session-name" value="MySession" /></pre>	

Property	Description	Default
Sessions XML	<p>Specify persistence information loaded from the EclipseLink session configuration file (<code>sessions.xml</code>).</p> <p>You can use this option as an alternative to annotations and deployment XML. If you specify this property, EclipseLink will override all class annotation and the object relational mapping from the <code>persistence.xml</code>, as well as <code>ORM.xml</code> and other mapping files, if present.</p> <p>Indicate the session by setting the <code>eclipselink.session-name</code> property.</p> <p>Note: If you do not specify the value for this property, <code>sessions.xml</code> file will not be used.</p> <p>Valid values: the resource name of the sessions XML file.</p> <p>Example: <code>persistence.xml</code> file</p> <pre><property name="toplink.session-xml" value="mysession.xml" /></pre>	
Target Database	<p>Select the target database. You can also set the value to the fully qualified class name of a subclass of the <code>org.eclipse.persistence.platform.DatabasePlatform</code> class.</p> <p>Example: <code>persistence.xml</code> file</p> <pre><property name="eclipselink.target-database" value="Oracle" /></pre>	Auto
Target Server	<p>Select the target server for your JPA application.</p> <p>Example: <code>persistence.xml</code> file</p> <pre><property name="eclipselink.target-server" value="OC4J_10_1_3" /></pre>	None
Event Listener	<p>Specify a descriptor event listener to be added during bootstrapping.</p> <p>Valid values: qualified class name for a class that implements the <code>eclipselink.sessions.SessionEventListener</code> interface.</p> <p>Example: <code>persistence.xml</code> file</p> <pre><property name="eclipselink.session-event-listener" value="mypackage.MyClass.class" /></pre>	
Include Descriptor Queries	<p>Enable or disable the default copying of all named queries from the descriptors to the session. These queries include the ones defined using EclipseLink API, descriptor amendment methods, and so on.</p>	True
Miscellaneous Options		

Property	Description	Default
Temporal mutable	Specify if all Date and Calendar persistent fields should be handled as mutable objects. Example: persistence.xml file <pre><property name="eclipselink.temporal.mutable" value="true"/></pre>	False
Lock timeout		5
Query timeout		5
Validation mode		Auto
Validate pre-persist group		
Validate pre-update group		
Validate pre-remove group		

Schema Generation

This table lists the properties of the Schema Generation page of the [persistence.xml Editor](#).

Note: This page is not available for projects using the **Generic** platform.

Property	Description	Default
DDL Generation Type	Select the type of DDL generation: <ul style="list-style-type: none"> ▪ None -- Do not generate DDL; no schema is generated. ▪ Create Tables -- Create DDL for non-existent tables; leave existing tables unchanged. ▪ Drop and Create Tables -- Create DDL for all tables; drop all existing tables. 	None
Output Mode	Select the DDL generation target: <ul style="list-style-type: none"> ▪ Both -- Generate SQL files and execute them on the database. ▪ Database -- Execute SQL on the database only (do not generate SQL files). ▪ SQL Script -- Generate SQL files only (do not execute them on the database). 	
DDL Generation Location	Specify where EclipseLink writes DDL output. Specify a file specification to a directory in which you have write access. The file specification may be relative to your current working directory or absolute. If it does not end in a file separator, then EclipseLink appends one that is valid for your operating system.	

Property	Description	Default
Create DDL File Name	Specify the file name of the DDL file that EclipseLink generates that contains SQL statements for creating tables for JPA entities. Specify a file name valid for your operating system.	createDDL.jdbc
Drop DDL File Name	Specify the file name of the DDL file that EclipseLink generates that contains SQL statements for dropping tables for JPA entities.	dropDDL.jdbc

Properties

This page enables you to add or remove the vendor-specific <properties> elements of persistence.xml.

To add a property, click **Add** then enter the property **Name** and **Value**.

Source

Using this page, you can manually edit the persistence.xml file.

See "[Managing the persistence.xml file](#)" on page 3-20 for additional information.

Preferences

This section includes information on the following preference pages:

- [Java Persistence Preferences page – JPA](#)
- [Java Persistence Preferences page – Errors/Warnings](#)

This section also includes information on the following project property pages:

- [Project Properties page – JPA](#)
- [Project Properties page – EclipseLink](#)
- [Project Properties page – Entity Generation](#)
- [Project Properties page – Errors/Warnings](#)
- [Project Properties page – JAXB Options](#)

Java Persistence Preferences page – JPA

Use the JPA options on the Java Persistence Preferences page to select the general settings for JPA development.

Property	Description
Entity generation from tables	
Default package	Specify the default package name for generated entities.
JPQL Editing	
Specify the case ...	Specify the case for JPQL identifiers when editing JPQL with content assist, <ul style="list-style-type: none"> ▪ Lowercase ▪ Uppercase

Property	Description
Match case of first character	If enabled, Dali will match the case with the first character.

Java Persistence Preferences page – Errors/Warnings

Property	Description
Enable project specific settings	<p>Select the severity level for reporting validation problem for each category:</p> <ul style="list-style-type: none"> ▪ Project ▪ Persistence unit ▪ Type ▪ Attribute ▪ Database ▪ Inheritance ▪ Queries and generators <p>You can expand each category to display the possible error and warning messages.</p>

Project Properties page – JPA

Use the JPA options on the Properties page to select the database connection to use with the project.

Note: Connection must be active to get data source specific help and validation.

This table lists the properties available in the JPA options preferences page.

Property	Description
Platform	Select the vendor-specific platform.
JPA Implementation	
Type	<p>Select User Library to select from the available user-defined or downloaded libraries.</p> <p>If you select Disable, you must manually include the JPA implementation library on the project classpath.</p>
Library	<p>Select a specific JPA library configuration.</p> <p>Click Manage libraries to create or update a user library.</p> <p>Click Download libraries to download a specific library configuration.</p>
Include libraries with this application	Specify if the selected libraries are included when deploying the application.

Property	Description
Connection	The database connection used to map the persistent entities. <ul style="list-style-type: none"> ■ To create a new connection, click Add Connections. ■ To reconnect to an existing connection, click Reconnect.
Override default catalog from connection	Select a catalog other than the default one derived from the connection information. Use this option if the default catalog is incorrect or cannot be used.
Override default schema from connection	Select a schema other than the default one derived from the connection information. Use this option if the default schema is incorrect or cannot be used. For example, use this option when the deployment login differs from the design-time login.
Persistent Class Management	Specify if Dali will discover annotated classes automatically , or if the annotated classes must be listed in the persistence.xml file. Note: To insure application portability, you should explicitly list the managed persistence classes that are included in the persistence unit.
Canonical Metamodel	Select the location of the metamodel source.

Project Properties page – EclipseLink

Use the EclipseLink options on the Properties page to select the EclipseLink-specific options to use with the project.

This table lists the properties available in the EclipseLink page.

Property	Description
Static weaving	
Weave classes on build	If enabled, Dali will weave all applicable class files at build time so that you can deliver pre-woven class files.
Source classes	Specifies the location of the Java source files to weave: either a directory or a JAR file.
Target classes	Specifies the output location: either a directory or a JAR file.
Log level	Specifies the amount and detail of log output. See " Logging " on page 4-44 for information on the different logging levels.
Persistence XML root	Specifies the location of the persistence.xml file if it is not in the same location as the source.

Project Properties page – Entity Generation

Use the Entity Generation options on the Properties page to configure the defaults Dali uses when generating entities

This table lists the properties available in the Entity Generation page.

Property	Description
Entity generation from tables	
Default package	Specify the default package name used for generated entities.

Project Properties page – Errors/Warnings

Use the Errors/Warnings options on the Properties page to specify if Dali should report errors and warnings for the project.

This table lists the properties available in the Errors/Warnings page.

Property	Description
Enable project specific settings	Specify if Dali reports errors and warning for the following features: <ul style="list-style-type: none"> ▪ Project ▪ Persistence unit ▪ Type ▪ Attribute ▪ Database ▪ Inheritance ▪ Queries and generators You can expand each category to display the possible error and warning messages.

Project Properties page – JAXB Options

Use the JAXB options on the Properties page to select the specific JAXB implementation use with the JAXB project.

This table lists the properties available in the JAXB project properties page.

Property	Description
Platform	Select the vendor-specific platform.
JAXB Implementation	
Type	Select User Library to select from the available user-defined or downloaded libraries. If you select Disable, you must manually include the JPA implementation library on the project classpath.
Library	Select a specific JPA library configuration. Click Manage libraries to create or update a user library. Click Download libraries to download a specific library configuration.
Include libraries with this application	Specify if the selected libraries are included when deploying the application.

Project Properties page – Schemas

Use the Schemas options on the Properties page to configure the JAXB schemas to use for validation and content assistance.

Click Add to

This table lists the properties available in the Schemas properties page.

Property	Description
Namespace	

Property	Description
Location	

Dialogs

This section includes information on the following dialogs:

- [Edit Join Columns dialog](#)
- [Add Join Column dialog](#)
- [Select Cascade dialog](#)
- [New EclipseLink Mapping File dialog](#)
- [Add Converter dialog](#)
- [Mapping Type Selection dialog](#)
- [JPA Metadata Conversion dialog](#)
- [Make Persistent dialog](#)
- [Add Query dialog](#)
- [Add Primary Key Join Column dialog](#)
- [Add Schema Location dialog](#)
- [Select Schema Location dialog](#)
-

Edit Join Columns dialog

Use the Join Columns dialog to create or modify the join tables and columns in relationship mappings.

This table lists the properties available in the Join Columns dialog.

Property	Description
Name	Name of the joint table column that contains the foreign key column.
Referenced Column Name	Name of the database column that contains the foreign key reference for the entity relationship.

Add Join Column dialog

Use the Join Columns dialog to create or modify the join tables and columns in relationship mappings.

This table lists the properties available in the Add Join Column dialog.

Property	Description
Name	Name of the joint table column that contains the foreign key column.

Property	Description
Referenced Column Name	Name of the database column that contains the foreign key reference for the entity relationship.
Table	
Column definition	
Insertable	
Updatable	
Unique	
Mutable	

Select Cascade dialog

Specify which operations are propagated throughout the association: All, Persist, Merge, Remove, or Refresh.

New EclipseLink Mapping File dialog

Specify the location and properties of the EclipseLink mapping file (`eclipselink-orm.xml`).

Property	Description
Project	Select the project in which to add the mapping file.
Source folder	Click Browse and select the source file in which to add the mapping file. The default is <code>../<PROJECT>/src</code> .
File path	Enter the filename and path of the mapping file. The default is <code>META-INF/eclipselink-orm.xml</code> .
Default access	Select whether the entity's access to instance variables is field -based or property -based, as defined in the JPA specification.
Add to persistence unit	Specify if this mapping file should be added to the persistence unit (<code>persistence.xml</code>).
Persistence Unit	Select the persistence unit in which to add the mapping file.

Add Converter dialog

Use this dialog to create a new EclipseLink converter.

Property	Description
Name	Enter the name for this converter. Converter names must be unique within the persistence unit.
Type	Select the converter type: <ul style="list-style-type: none"> ■ Custom ■ Object type ■ Struct ■ Type

Mapping Type Selection dialog

Use this dialog to select a specific mapping type for the attribute or entity.

Property	Description
Enter mapping type or pattern	Enter the name (or part of a name) of a mapping type. Leave blank to show all available options.
Matched items	Dali displays the mapping types that match your search pattern.

JPA Metadata Conversion dialog

Use this dialog to export your JPA metadata (converters, queries, and generators) to an XML mapping file.

Duplicated or overridden annotations *will not* be included in the generated mapping file.

Property	Description
Mapping file	Name and location of the XML file in which to save the JPA metadata

WARNING: Malformed metadata will result in a non-functional mapping file

Make Persistent dialog

Use this dialog to add persistence to a Java class.

Property	Description
Annotate in Java	Specify if Dali should use annotations. If disabled, Dali will add persistence information in the XML mapping file.
Add to XML mapping file	Specify if Dali should add persistence information in the XML mapping file. If disabled, Dali will use annotations.
Java classes	For each Java class, select the Mapping type: <ul style="list-style-type: none"> ▪ Entity ▪ Embeddable ▪ Mapped superclass
List in persistence.xml	Specify if Dali should add persistence properties to the persistence.xml file.

Add Query dialog

Use this dialog to add a new named query or native named query .

Property	Description
Name	Name of the query
Type	Select the type of query to create: <ul style="list-style-type: none"> ▪ Named query ▪ Native named query

Add Primary Key Join Column dialog

Property	Description
Name	
Referenced column name	
Table	
Column definitions	

Add Schema Location dialog

Use this dialog to configure a new schema namespace and the location where it can be found.

Property	Description
Location	Click Browse and use the Select Schema Location dialog to specify the location of the schema.
Namespace	

Select Schema Location dialog

Property	Description
Select file from Workspace	
Select XML catalog entry	
Workspace files	

Add Virtual Attribute dialog

Use this dialog to add a new virtual attribute to the JPA entity.

Property	Description
Name	Name of the virtual attribute
Map As	Select the mapping for the attribute
Attribute type	Click Browse and select the Java type of the attribute.

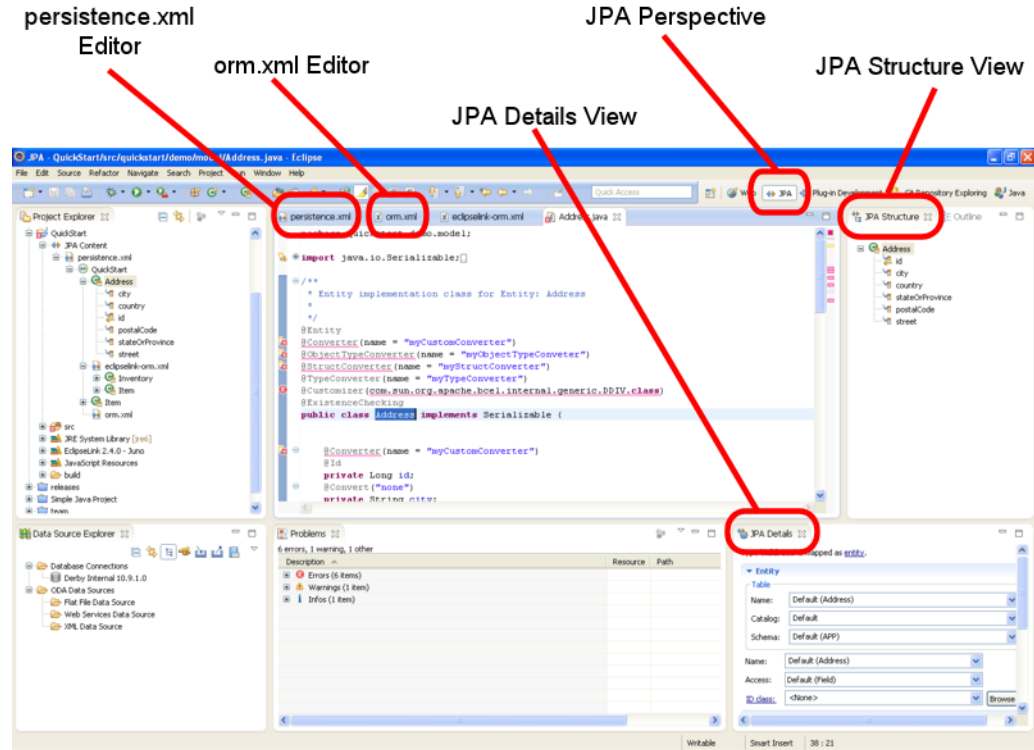
JPA Development perspective

The **JPA Development perspective** defines the initial set and layout of views in the Workbench window when using Dali. By default, the JPA Development perspective includes the following views:

- [JPA Structure view](#)
- [JPA Details view \(for entities\)](#)

- JPA Details view (for attributes)
- JPA Details view (for orm.xml)

Figure 4–2 Sample JPA Development Perspective



Icons and buttons
















This section includes information on each of the icons and buttons used in the Dali OR Mapping Tool.

- [Icons](#)
- [Buttons](#)

Icons


The following icons are used throughout the Dali OR Mapping Tool.

Icon	Description
Entity icons	
	Entity
	Embeddable entity
	Mapped superclass
Mapping icons	
	Array mapping

Icon	Description
	Basic mapping
	Basic collection mapping
	Basic map mapping
	Element collection mapping
	Embedded mapping
	Embedded ID mapping
	ID mapping
	Many-to-many mapping
	Many-to-one mapping
	One-to-many mapping
	One-to-one mapping
	Transformation mappings
	Transient mapping
	Variable one-to-one mappings
	Version mapping

Buttons

The following buttons are used throughout the Dali OR Mapping Tool.

Icon	Description
	JPA Development perspective

Dali developer documentation

Additional Dali documentation is available online at:

http://wiki.eclipse.org/index.php/Dali_Developer_Documentation

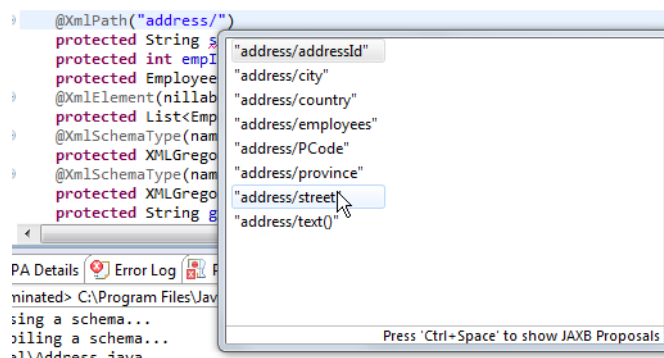
This developer documentation includes information about:

- Dali architecture
- Plugins that comprise the Dali JPA Eclipse feature
- Extension points

Tips and tricks

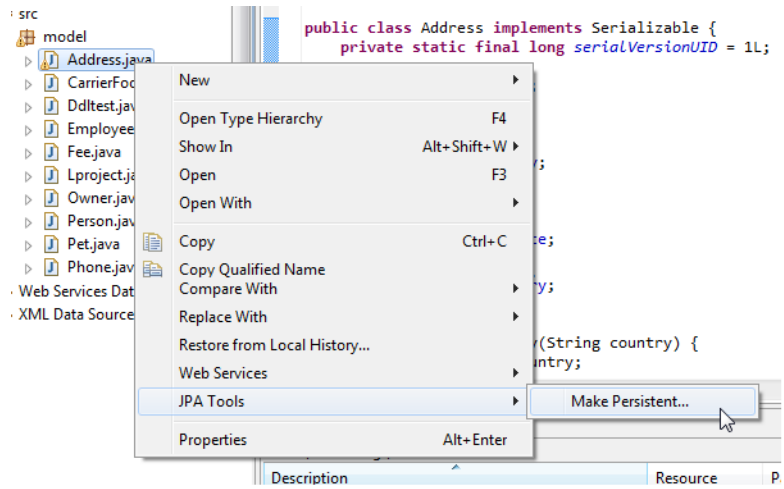
The following tips and tricks give some helpful ideas for increasing your productivity.

Database connections	When starting a new workbench session, be sure to reconnect to your database (if you are working online). This allows Dali to provide database-related mapping assistance and validation.
Schema-based persistence.xml file	If you are behind a firewall, you may need to configure your Eclipse workspace proxy in the Preferences dialog (Preferences > Internet > Proxy Settings) to properly validate a schema-based persistence.xml file.
@XPath content assist and validation	Dali supports @XPath content assist and validation. You can now easily traverse deeply nested XML structures specifying XPath values.



Making persistent entities

With the [Make Persistent dialog](#), you can easily transform Java classes into persistent entities via Java annotation or entry in an XML mapping file. With multi-select, you can quickly create many entities at once.



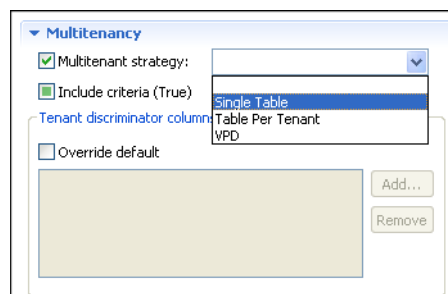
This section contains descriptions of the following new features and significant changes made to the Dali OR Mapping Tool for Release 3.2 (Web Tools Platform 3.4):

- [EclipseLink multitenancy support](#)
- [EclipseLink static weaving support](#)
- [Generating EclipseLink dynamic entities from tables](#)
- [Converting JPA metadata to XML](#)
- [EclipseLink 2.4 support](#)

EclipseLink multitenancy support

Dali Release 3.2 supports configuration of EclipseLink's multitenancy feature. Multitenancy allows multiple application tenants to share the same schema using tenant descriptor columns. Dali supports the following multitenant strategies:

- Single table
- Table per tenant
- VPD

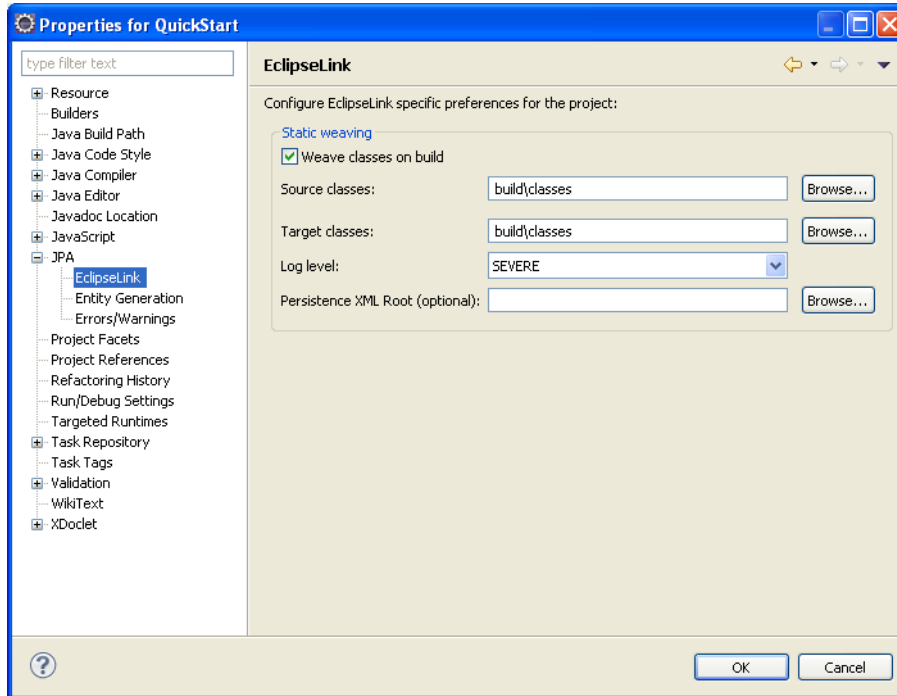


For more information, see:

- ["Multitenancy" on page 4-19](#)
- ["Multitenant" in the *Java Persistence API \(JPA\) Extensions Reference for EclipseLink*](#)
http://www.eclipse.org/eclipselink/documentation/2.4/jpa/extensions/a_multitenant.htm

EclipseLink static weaving support

Dali Release 3.2 allows the configuration of EclipseLink's weaving support at the project properties level. Static weaving allows you to use EclipseLink's weaving support in cases where dynamic weaving is not available or is not an option. Dali configures and executes the byte code weaving of compiled Java classes

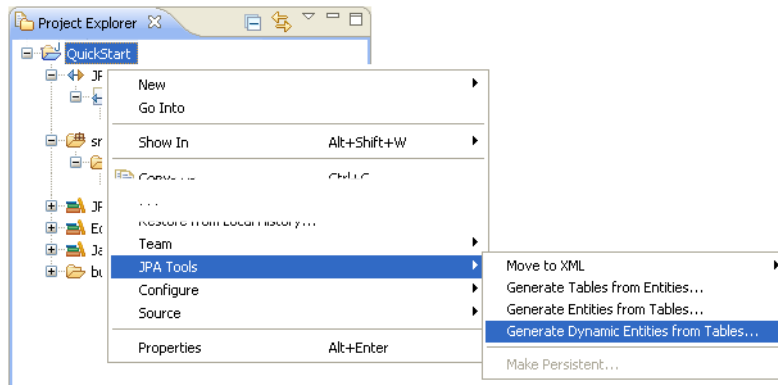


For more information, see:

- "Project Properties page – EclipseLink" on page 4-52
- "Static Weaving" in the EclipseLink documentation
http://wiki.eclipse.org/EclipseLink/UserGuide/JPA/Advanced_JPA_Development/Performance/Weaving/Static_Weaving

Generating EclipseLink dynamic entities from tables

When using EclipseLink JPA, you can create dynamic entities from your database tables. This dynamic persistence provides access to a relational database with all the benefits of JPA *without coding* or maintaining Java classes.

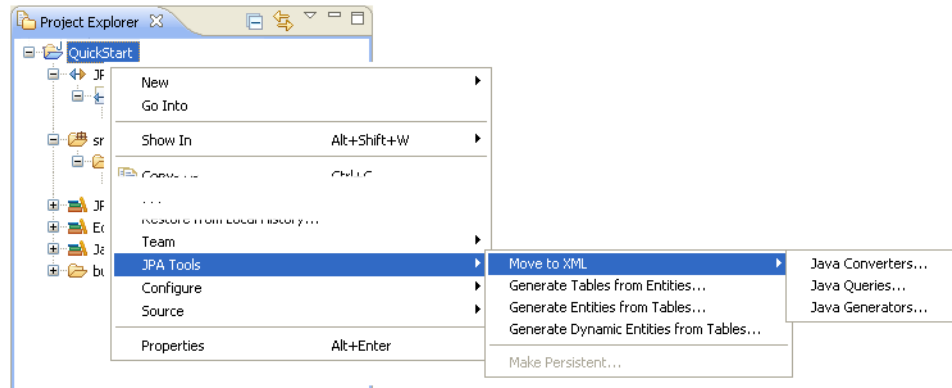


For more information, see:

- ["Generating dynamic entities from tables"](#) on page 3-52

Converting JPA metadata to XML

Dali can convert JPA metadata (such as converters, queries, and generators) into an XML mapping file. This allows you to maintain the global metadata for a persistence unit (such as queries and generators) in an XML mapping file.

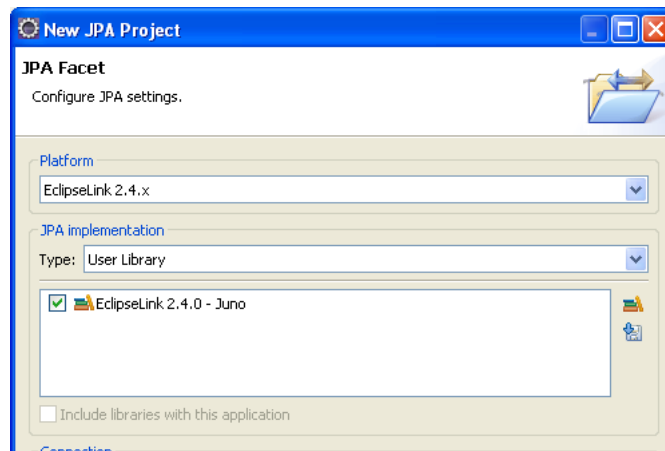


For more information, see:

- ["Converting JPA metadata to XML"](#) on page 3-56

EclipseLink 2.4 support

Release 3.2 provides support for EclipseLink 2.4.x.



EclipseLink (the Eclipse Persistence Services Project) is a complete persistence framework. Refer to <http://www.eclipse.org/eclipselink/> for more information.

Copyright © 2011, 2012, Oracle. All rights reserved.

This program and the accompanying materials are made available under the terms of the Eclipse Public License v1.0 which accompanies this distribution, and is available at:

<http://www.eclipse.org/legal/epl-v10.html>

About this content

August, 2012

License

The Eclipse Foundation makes available all content in this plug-in ("Content"). Unless otherwise indicated below, the Content is provided to you under the terms and conditions of the Eclipse Public License Version 1.0 ("EPL"). A copy of the EPL is available at <http://www.eclipse.org/legal/epl-v10.html>. For purposes of the EPL, "Program" will mean the Content.

If you did not receive this Content directly from the Eclipse Foundation, the Content is being redistributed by another party ("Redistributor") and different terms and conditions may apply to your use of any object code in the Content. Check the Redistributor's license that was provided with the Content. If no such license exists, contact the Redistributor. Unless otherwise indicated below, the terms and conditions of the EPL still apply to any source code in the Content and such source code may be obtained at <http://www.eclipse.org>.

Symbols

@Basic, 3-35, 4-23
@Column, 4-23, 4-25, 4-29
@DiscriminatorColumn, 4-18
@DiscriminatorValue, 4-18
@Embeddable, 3-17
@Embedded, 3-37
@EmbeddedId, 3-38
@Entity, 3-16
@Enumerated, 4-30
@GeneratedValue, 4-33
@Id, 3-39, 4-25
@Inheritance, 3-32
@JoinColumn, 4-32
@Lob, 4-30
@ManyToMany, 3-40, 4-25
@ManyToOne, 3-41, 4-26
@MappedSuperclass, 3-18
@NamedQuery, 3-33
@ObjectTypeConverter, 4-21
@OneToMany, 3-42, 4-27
@OneToOne, 3-43, 4-28
@OrderBy, 4-31
@SecondaryTable, 3-31
@SequenceGenerator, 4-20, 4-21, 4-34
@StructConverter, 4-21
@Temporal, 4-30
@Transient, 3-44
@TypeConverter, 4-21
@Version, 3-44, 4-29
@XMLPath, 5-1

A

Add Converter dialog, 4-55
Add Join Column dialog, 4-54
Add Primary Key Join Column dialog, 4-57
Add Query dialog, 4-56
Add Schema Location dialog, 4-57
Add Virtual Attribute dialog, 4-57
Advanced, in Java Details view, 4-22
annotations. *See specific annotation.*
architecture of Dali feature, 4-59
association tables, 4-12
Attribute Overrides, in Java Details view, 4-19

attributes
 JPA Details view, 4-22
 mapping, 2-1
 overrides, 4-19
 virtual, 3-19
attributes, virtual, 4-57

B

basic mapping
 @Basic, 3-35
 about, 3-35, 4-23
 See also mappings
batch writing, persistence unit, 4-39

C

caching, 4-15, 4-42
Caching, in persistence.xml editor, 4-42
canonical metamodel, 4-52
cardinality, association tables, 4-13
change tracking, 4-22
classes
 adding persistence to, 3-15
 embeddable, 3-17
 entity, 3-15
 managed, 4-38
 managing persistent classes, 4-10
 mapped superclass, 3-18
 synchronizing, 3-28
collection mappings, 3-36
columns
 discriminator, 4-18
 join, 4-32
 mapping to, 4-23, 4-25, 4-29
 value, 4-18
connection pool, 3-23
Connection, in persistence.xml Editor, 4-38
connections, database, 5-1
converters, 4-30
Converters, in JPA Details view, 4-21, 4-30, 4-36
converting
 Java project to JPA, 3-9
 JPA metadata to XML, 3-56
Create a JPA Project Wizard, 3-2, 3-5, 3-8
Create JPA Entity wizard, 4-5

- Create New Association wizard, 4-12
- Create New JAXB Project wizard, 4-10
- Create New JPA Project wizard, 4-8
- Create ORM Mapping File wizard, 4-7
- Customization, in persistence.xml editor, 4-40
- customizer class, for entities, 4-22

D

- database tables, generating entities from, 3-45
- Database Web Services. *see* DBWS
- database, persistence
 - connection, 4-51, 4-53
 - schema, 4-52
- DBWS project, creating, 3-7, 4-11
- DDL script, generating, 3-50
- DDL, generating, 4-12
- Derived Identity, in JPA Details view, 4-32
- developer documentation, Dali, 4-59
- Driver Files page, 3-9
- dynamic entities, from tables, 3-52

E

- eager fetch, 4-23
- EclipseLink XML mapping file, 4-38
- eclipselink-orm.xml file, with dynamic entities, 3-55
- Edit Join Columns dialog, 4-54
- EJB. *see* persistent entities
- element collection mapping
 - about, 4-24
 - creating, 3-36
- embeddable class
 - @Embeddable, 3-17
 - about, 3-17
 - JPA Details view, 4-14
- embedded ID mapping
 - @EmbeddedId, 3-38
 - about, 3-38
- embedded mapping
 - @Embedded, 3-37
 - about, 3-37
- entities
 - @Entity annotation, 3-16
 - about, 2-1
 - creating, 3-11, 4-5
 - creating tables from, 3-50, 4-12
 - customer class, 4-22
 - customizing, 4-3, 4-5
 - embeddable, 3-17
 - from tables, 3-45, 3-52, 4-2, 4-4
 - generating, 4-2, 4-4
 - JPA Details view, 4-13
 - mapped superclass, 3-18
 - mapping, 1-3, 3-34
 - persistent, 1-3, 3-15, 3-16
 - read-only, 4-22
 - secondary tables, 4-21
 - tracking changes, 4-22
- Entity Class page, 3-13

- Entity Properties page, 3-14
- enumerated, 4-30
- error messages, Dali, 3-57, 3-58
- extension points, Dali feature, 4-59

F

- fetch type, 4-23
- firewall, 5-1

G

- General, in persistence.xml Editor, 4-37
- Generate Dynamic Entities from Tables wizard, 4-3
- Generate Entities from Tables dialog, 3-45, 4-2, 4-4
- Generate Entities from Tables wizard, 4-1
- Generate Tables from Entities wizard, 4-12
- generated values
 - entities, preferences, 4-52
 - ID mappings, 4-33
 - sequence, 4-20, 4-21, 4-34
- generating DDL, 4-12
- Generators, in JPA Detail view, 4-35

H

- hints, query, 3-34, 4-36

I

- ID mapping
 - @Id, 3-39
 - about, 3-39, 4-25
- identity, derived, 4-32
- inheritance
 - entity, 3-32, 4-6, 4-18
 - joined tables, 3-33
 - single table, 3-33
- Inheritance, in Java Details view, 4-18
- installation, Dali, 1-1

J

- Java Persistence API (JPA)
 - about, 2-2
- Java project, converting to JPA, 3-9
- JAXB
 - about, 2-2
 - options, 4-53
 - schema, adding, 4-57
- JAXB project
 - creating, 3-5, 4-10
 - implementation, 4-11
- JAXB schema, adding, 4-57
- join columns, 4-13, 4-54
- joined tables, inheritance, 3-33
- joining, 4-32
- Joining Strategy, in JPA Details view, 4-32
- JPA Details view
 - attributes, 4-22
 - entities, 4-13

- Mappings tab, 4-22
- orm.xml file, 4-34
- JPA Development perspective, 4-57
- JPA entity, creating, 3-11
- JPA Facet page, 3-4, 3-7
- JPA Metadata Conversion dialog, 4-56
- JPA project
 - converting from Java, 3-9
 - converting metadata to XML, 3-56
 - creating, 3-1, 4-8
 - implementation, 4-9
 - page, 3-2, 3-6
 - platform, 4-51, 4-53
- JPA Structure view, 4-36

K

- key, primary, 4-20

L

- lazy fetch, 4-23
- library
 - JAXB, 4-11
 - JPA, 4-9
- Logging, in persistence.xml editor, 4-44

M

- Make Persistent dialog, 4-56
- many-to-many mapping
 - @ManyToMany, 3-40
 - about, 3-40, 4-25
- many-to-one mapping
 - @ManyToOne, 3-41
 - about, 3-41, 4-26
- mapped superclass
 - @MappedSuperclass, 3-18
 - about, 3-18
 - JPA Details view, 4-15
- mapping entities, 1-3
- mapping file, 3-29, 4-6, 4-7
- Mapping Type Selection dialog, 4-56
- mappings
 - about, 2-1, 3-34, 4-22
 - basic, 3-35, 4-23
 - element collection, 3-36, 4-24
 - embedded, 3-37
 - embedded ID, 3-38
 - ID, 3-39, 4-25
 - many-to-many, 3-40, 4-25
 - many-to-one, 3-41, 4-26
 - one-to-many, 3-42, 4-27
 - one-to-one, 3-43, 4-28
 - problems, 3-57
 - transient, 3-44
 - version, 3-44, 4-29
- metadata, converting to XML, 3-56
- metamodel, canonical, 4-52
- multitenancy, 4-19

N

- named queries, 4-36
 - entity, 3-33
 - hints, 3-34
 - JPA Detail view, 4-17
- native queries, 3-34, 4-17, 4-36
- New Database Web services from Builder XML wizard, 4-11
- New EclipseLink Mapping File dialog, 4-55
- nonpersistent
 - classes, 3-15
 - fields. *See* transient

O

- object type converter, 4-21
- ObjectChangePolicy, 4-22
- one-to-many mapping
 - @OneToMany, 3-42
 - about, 3-42, 4-27
- one-to-one mapping
 - @OneToOne, 3-43
 - about, 3-43, 4-28
- Options, in persistence.xml editor, 4-47
- OR (object-relational) mappings. *See* mappings
- ordering, 4-31
- Ordering, in JPA Details view, 4-31
- orm.xml file
 - about, 2-2
 - creating, 4-7, 4-10
 - generators, 4-35
 - JPA Details view, 4-34
 - managing, 3-28
 - queries, 4-36
 - sample, 3-28
- outline, persistence. *See* JPA Structure view
- overrides, JPA attributes, 4-19

P

- persistence
 - about, 2-1
 - database connection, 4-51, 4-53
 - database schema, 4-52
 - entity class, 3-15
 - options, 4-51
 - provider, 4-37
 - unit, 4-35
- Persistence Unit, in JPA Details, 4-35
- persistence.xml Editor, 3-20 to 3-27, 4-37
- persistence.xml file
 - about, 2-2
 - managing, 3-20 to 3-27, 3-30
 - sample, 3-20
 - schema based, 5-1
 - synchronizing with classes, 3-28
- persistent entity, 3-15
- perspective, JPA Development, 4-57
- platform, JPA, 4-51, 4-53
- preferences

- Dali, 4-50
- errors and warnings, 4-51
- JPA, 4-50
- project, EclipseLink, 4-52
- project, Entity Generation, 4-52
- project, errors and warnings, 4-53
- project, JAXB, 4-53
- project, JPA, 4-51
- Primary Key Generation, in JPA Details view, 4-20, 4-33
- problems, 3-57
- projects
 - DBWS, 4-11
- projects, DBWS
 - creating new, 3-7
- projects, JAXB
 - creating new, 3-5, 4-10
 - options, 4-53
- projects, JPA
 - creating new, 1-2, 3-1, 4-8
 - options, 4-51
 - properties, 3-56
- properties
 - project, 3-56
 - <properties> element, 4-50
- Properties, in persistence.xml editor, 4-50
- property pages, 4-13

Q

- queries, 4-17, 4-36
- query hints, 3-34, 4-36
- Query, in JPA Details view, 4-36
- quick start, Dali, 1-2

R

- read-only entities, 4-22
- requirements
 - Dali Java Persistence Tools, 1-1
 - persistent entities, 3-15

S

- schema
 - database, 4-52
 - JAXB, 4-57
- Schema Generation, in persistence.xml editor, 4-49
- schema, JAXB, 4-57
- secondary tables, 3-31, 4-21
- Secondary Tables, in Java Details view, 4-21
- Select Cascade dialog, 4-55
- Select Schema Location dialog, 4-57
- single table inheritance, 3-33
- Source, in persistence.xml editor, 4-50
- Specify, 4-55
- struct converter, 4-21
- superclass, 3-18
- synchronizing classes with persistence.xml file, 3-28

T

- tables
 - associations, 4-2, 4-4, 4-12
 - creating dynamic entities from, 3-52
 - creating entities from, 3-45, 4-2, 4-4
 - creating from entities, 4-12
 - from entities, 3-50
 - inheritance, 3-33
 - secondary, 3-31, 4-21
- temporal, 4-30
- thread identifier, in logging, 4-45
- timestamp, in logging, 4-45
- tracking changes, 4-22
- transaction type, persistence unit, 4-38
- transient mapping
 - @Transient, 3-44
 - about, 3-44
- type converter, 4-21
- Type Information, in JPA Details view, 4-30

V

- vendor-specific properties, 4-50
- version mapping
 - @Version, 3-44
 - about, 3-44, 4-29
- views
 - JPA Details view, 4-13, 4-22
 - JPA Structure view, 4-36
- virtual access type, 3-52
- virtual attributes, adding, 3-19, 4-57

W

- warning messages, Dali, 3-57
- weaving
 - EclipseLink, 4-52
- Web Dynamic Project page, 3-8
- Web Tools Platform (WTP), 1-1
- wizards, 4-1

X

- XML editor, 3-30
- XML, project metadata, 3-56
- XMLPath, 5-1