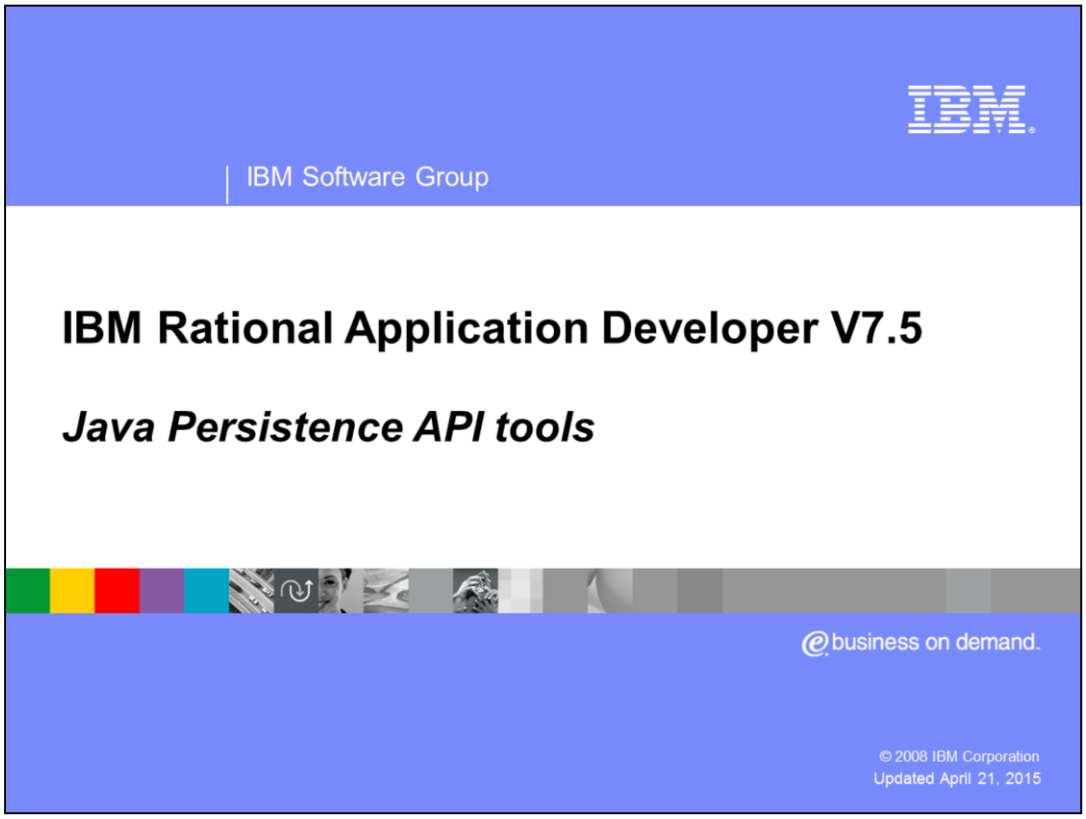
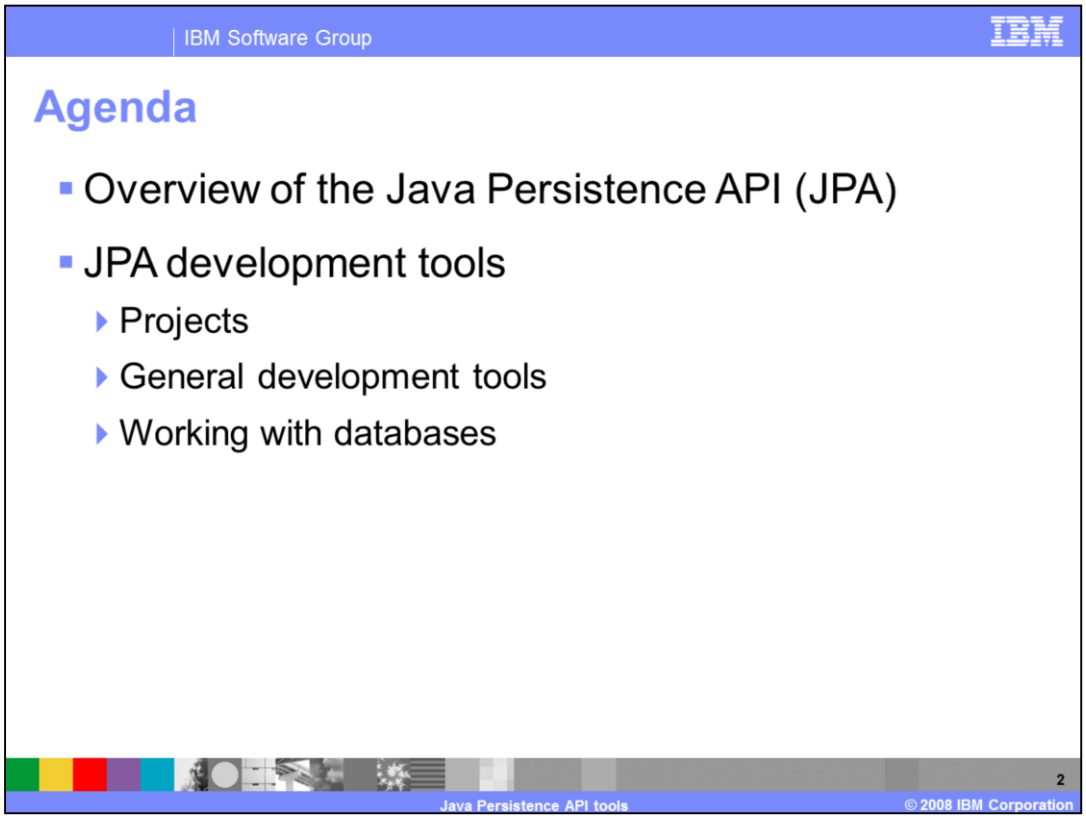
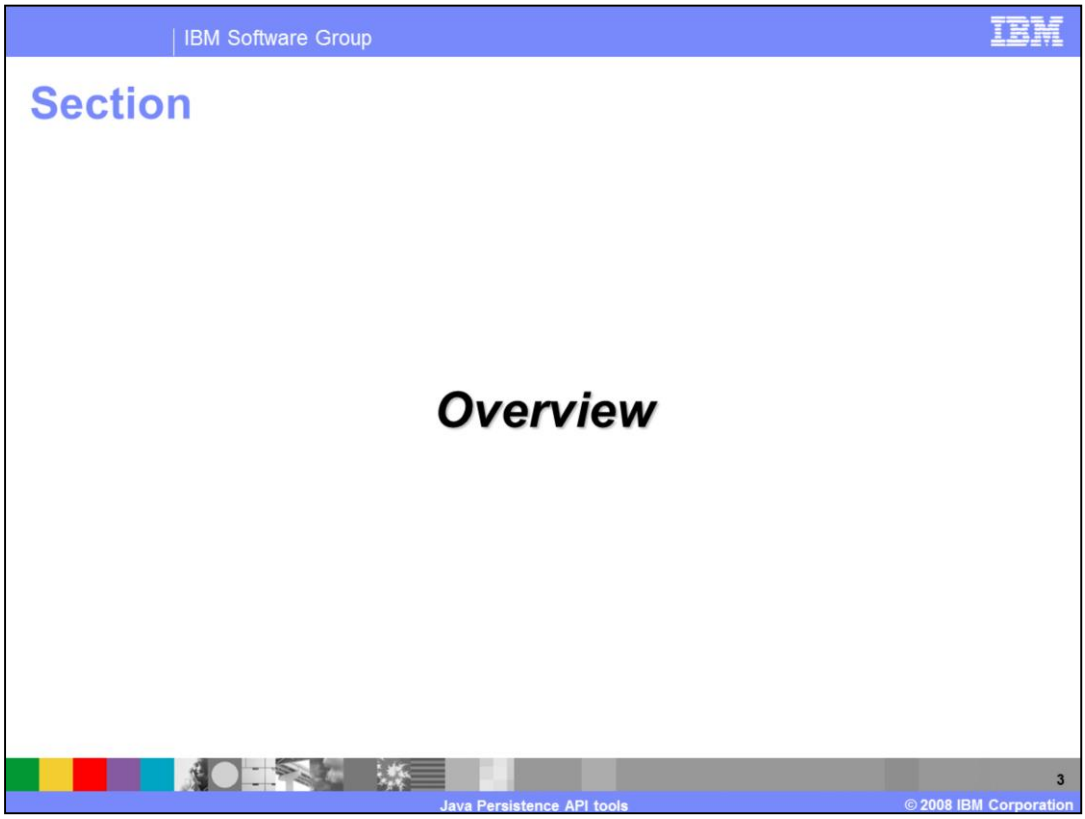


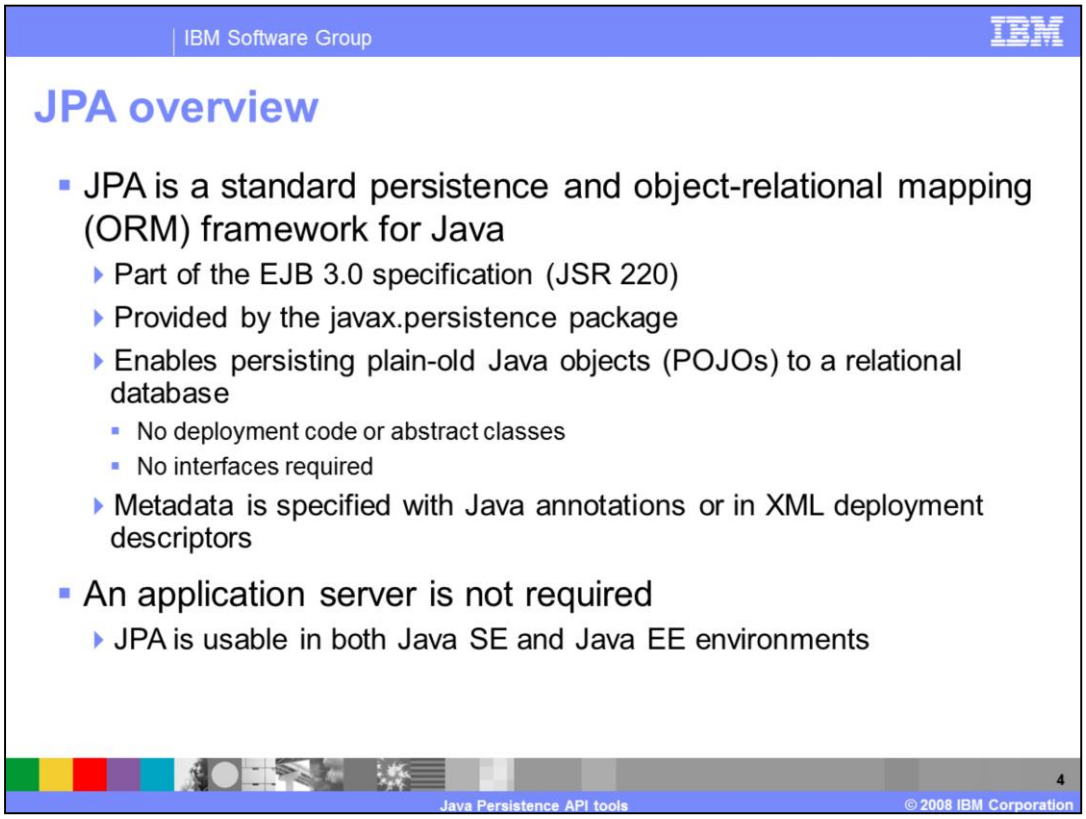

This presentation describes the Java™ Persistence API (JPA) and the tools available in Rational® Application Developer V7.5 to do JPA development.



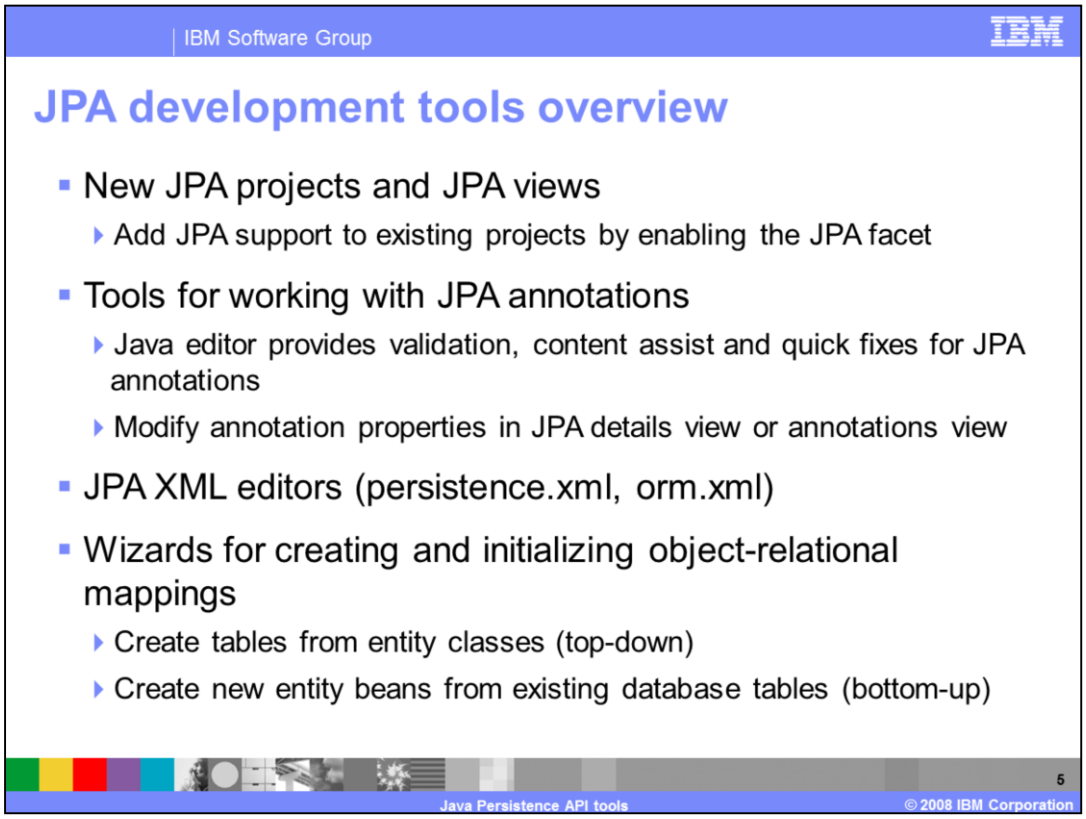

The first section of this presentation provides an overview of the Java Persistence API and the JPA tools that are available in the Rational Application Developer workbench. The rest of the presentation covers these development tools in more detail, including how to work with JPA projects, general JPA development tools; like how to create entities from tables and using the new specialized views in the workbench. Finally, how to configure and connect to databases for application testing.



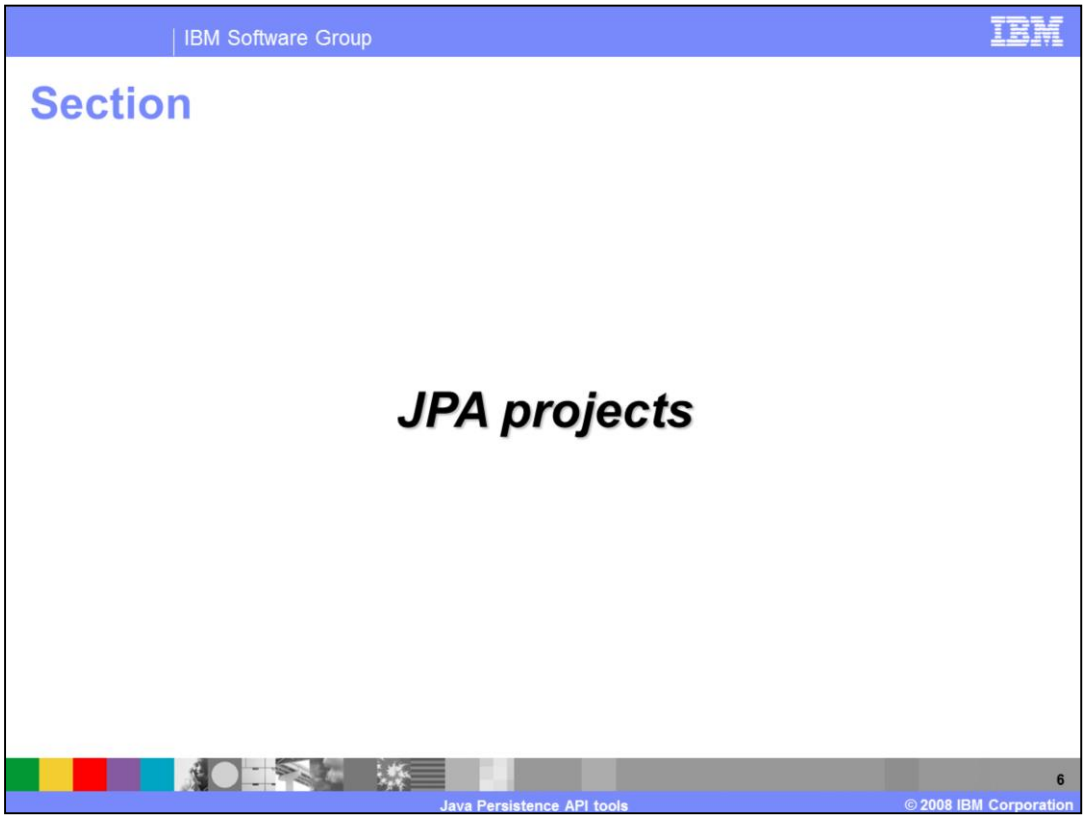

This section includes an overview of the Java Persistence API and JPA development tools in Rational Application Developer.




The Java Persistence API is a standard framework that provides persistence and object-relational mapping as a part of the EJB 3.0 specification. It is a Java standard that provides many of the benefits of alternative persistence frameworks, with the added benefit of portability to any EJB 3.0 compliant container, without having to install any extra libraries. It allows you to persist plain-old Java objects to a relational database – a much simpler approach than using container-managed persistence (CMP) in EJB 2.1. In JPA, Entities are plain-old Java objects, just like other components in EJB 3.0. An entity typically represents a table in a relational database, and each instance of an entity is a row. Entities are concrete classes, not abstract classes like Entity Beans in EJB 2.1. You do not need to generate deployment code, which speeds deployment, and you do not have to implement any particular interfaces. Another benefit is that all object-relational mapping information is specified in a standard fashion, using Java annotations or XML files. In earlier versions of the EJB specification, there was not a standard way to provide this information, which meant each vendor's implementation was different, and led to a reliance on vendor-specific tools. JPA can also be used without an EJB container, that is, in a Java Standard Edition (SE) environment.

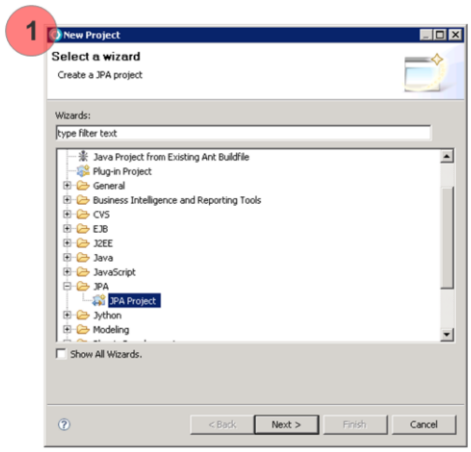



With extensible frameworks and tools, you can perform many tasks that are associated with accessing data for a multi-tier enterprise application. You can work with JPA properties in either the JPA Details view or the Annotations view, so that you don't need to keep both views open at once. For clarity, the Annotations view distinguishes between implied and specified annotation attributes. Like in Java EE 5 application development, deployment descriptors in JPA applications are optional – the programming model for the JPA relies on using annotations to specify information about how the Java object will map to a database table. However, deployment descriptors can still be used in a JPA application, and the workbench provides deployment descriptor editors to work with those files. In order to minimize the complexity of mapping between JPA entities and tables, you can use wizards to create and automate initial mappings. You can also receive programming assistance from the tools through dynamic problem identification.

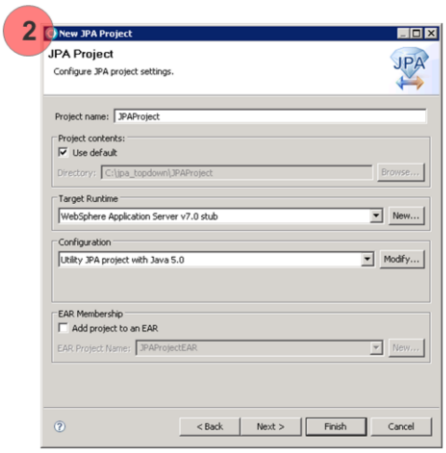



This section describes how to create and work with JPA projects.

# JPA projects – new project wizard




Start the JPA project wizard by selecting **File > New > Project > JPA > JPA Project**

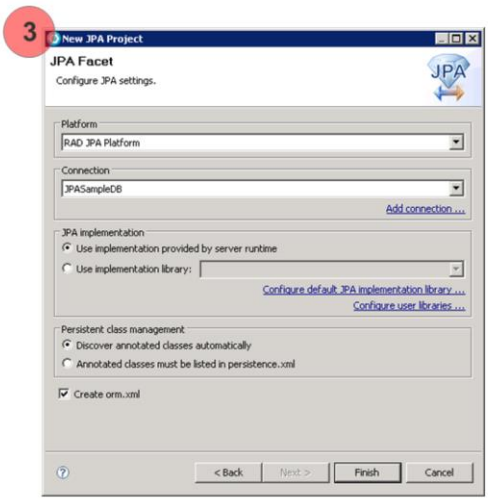



Name the project and choose the **Target Runtime**




While the JPA is a part of the Java EE 5 specification, you can also create stand-alone JPA projects. You can do this either using the JPA project creation wizard or by modifying an existing Java project. In the project creation wizard, you need to choose a project name and specify which target runtime to use for testing.

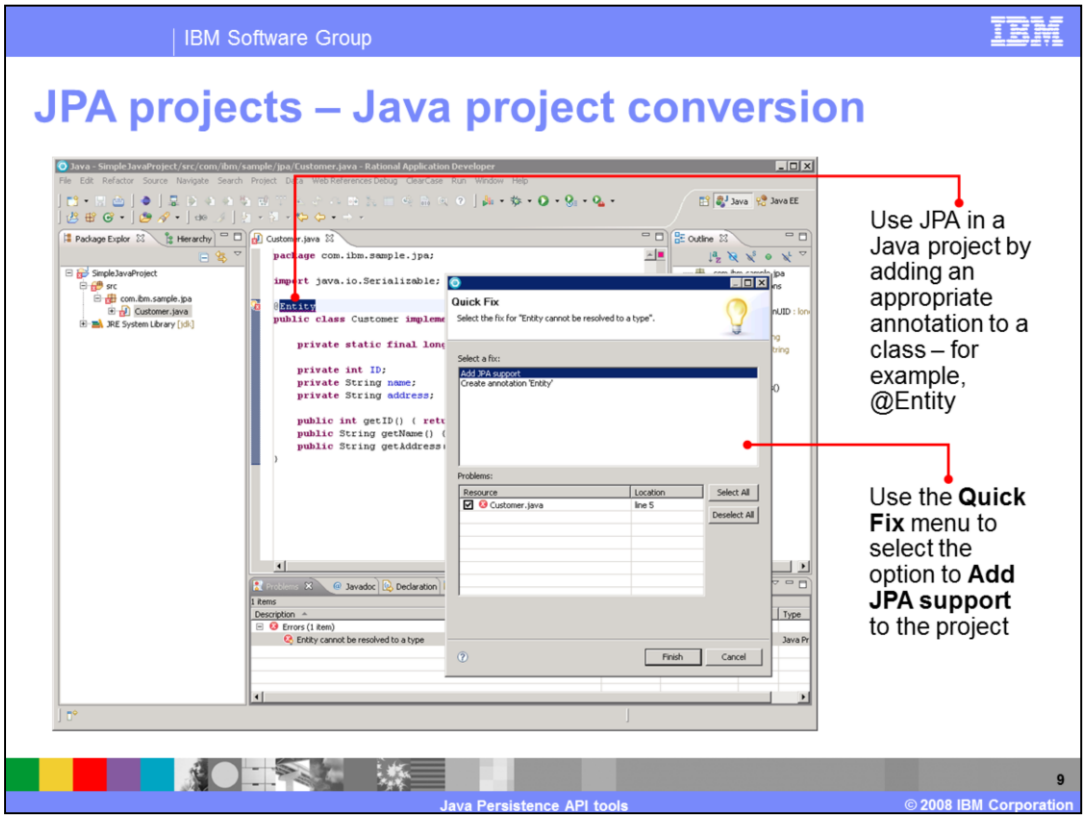
# JPA projects – new project wizard (continued)



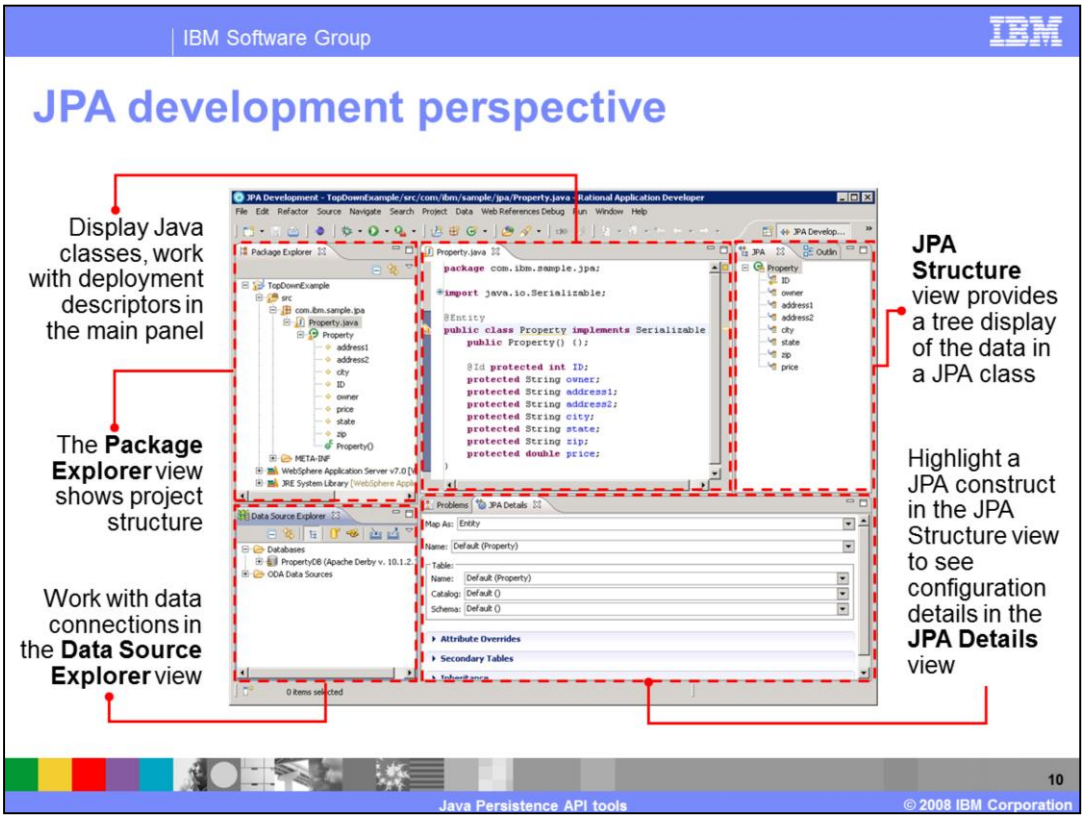

- Configure additional JPA settings on the JPA facet panel
- Add a connection to a database
- Choose which implementation library to use



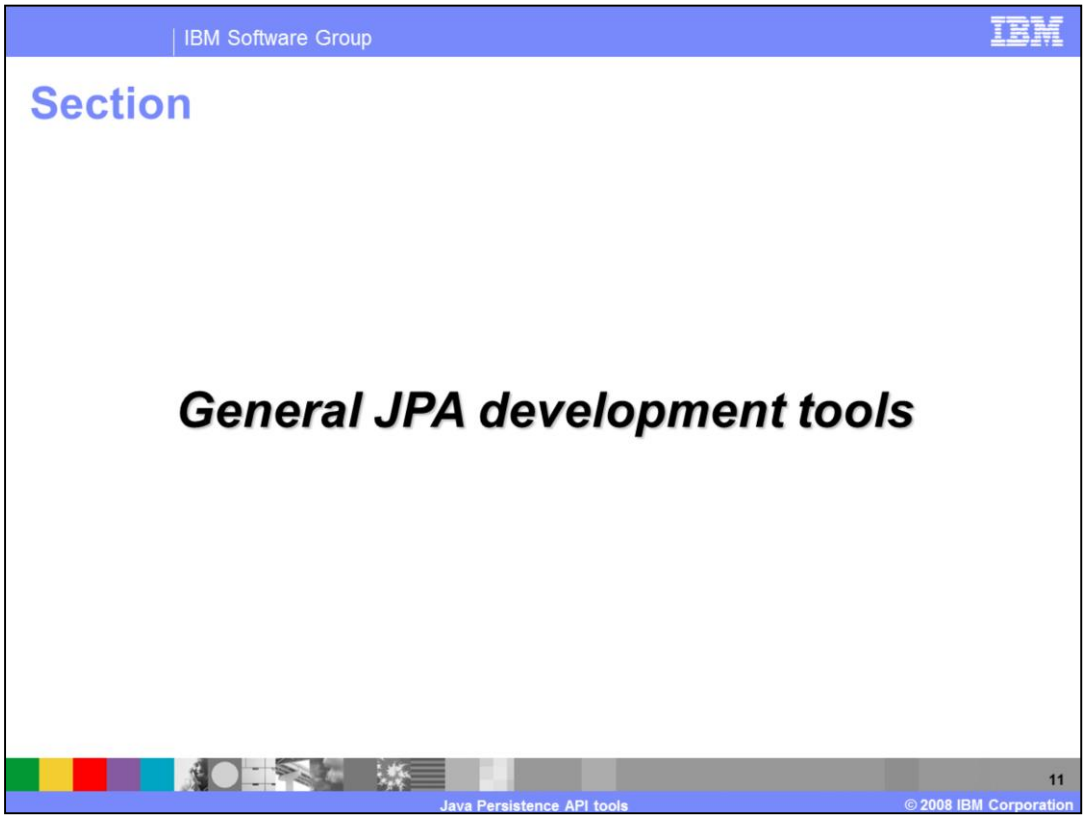

After choosing the project name and runtime, you can configure additional options associated with the JPA facet. This is the same panel that you see if you go in and edit the project properties later. From here, choose which JPA platform to use. You also have the option of adding a connection to a database. If you do not create a database connection at project creation time, you need to do so later, before you can test your application. You have the option of choosing a specific JPA implementation library to use with your project. If you are using a WebSphere® Application Server V7 runtime and choose to use the implementation provided by the server, then you are using an implementation based on Apache OpenJPA. Deployment descriptors are optional in a JPA project, but you can choose to create the persistence.xml and orm.xml files at project creation time, if needed.




You can also convert a Java project into a JPA project, using Quick Fixes that are available in the Java editor. When you type a JPA annotation into a plain old Java object (POJO), the resulting compilation errors have Quick Fixes for adding the required JPA facet and an appropriate runtime environment (if available) to your project. For example, adding the @Entity annotation to the beginning of a Java class indicates that you want to store that class's fields to a database. The development environment automatically recognizes this as a JPA annotation and includes the appropriate fixes in the Quick Fix menus.



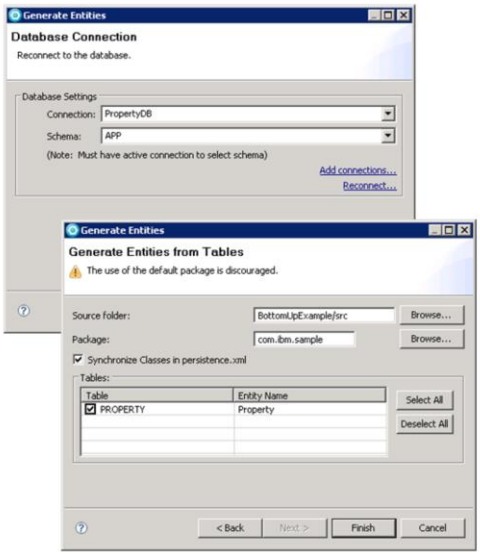

The JPA Development perspective in the Rational Application Developer workbench includes database-related tools and specialized JPA views to assist in JPA application development and testing. The JPA Structure view provides a tree display of the data in a JPA class. Highlight a JPA construct in the JPA structure view to see the configuration details for that item in the JPA Details view. This view provides detailed information on the attributes of a JPA entity, including the name of the table it's being mapped to and all of the attributes of the mapping. Many of the JPA attributes do not need to be specified in your application and have context-appropriate default values; you can see these default values in the JPA Details view. The JPA Development perspective also includes the Data Source Explorer, which you can use to work with data source connections for application testing.




This section describes general JPA development tools available in Rational Application Developer V7.5.

# Creating entities from database tables

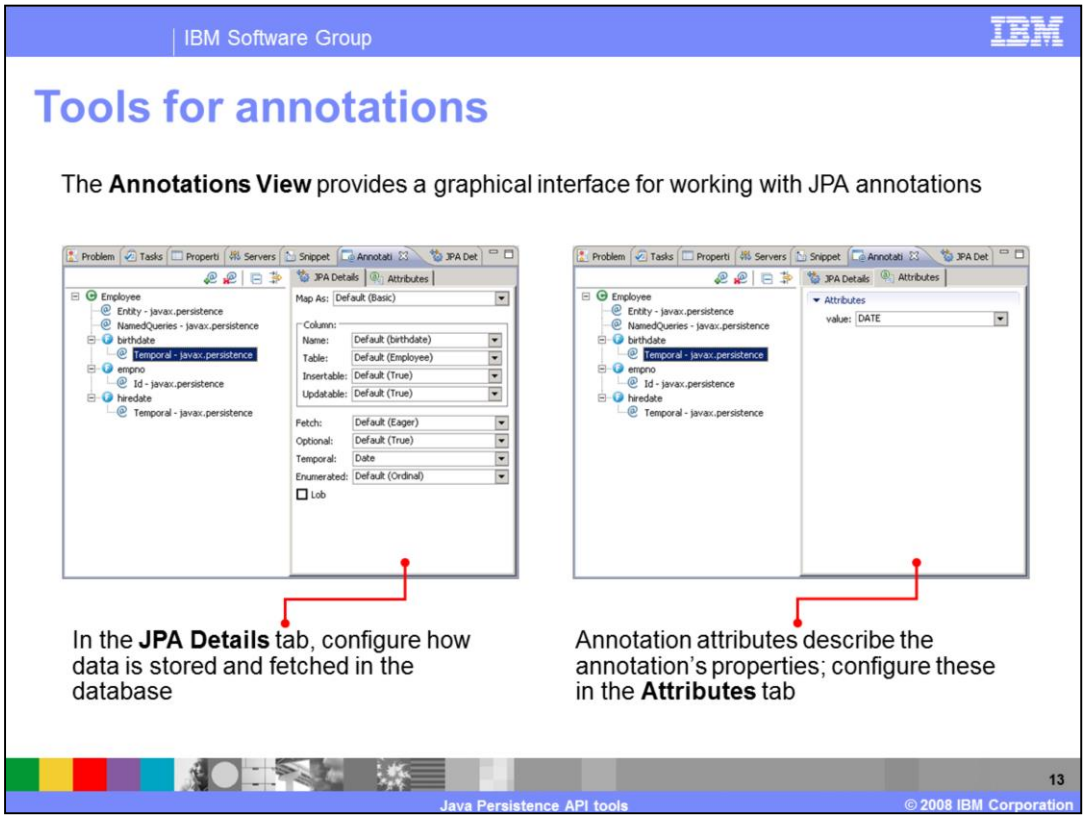
- Configure a database connection in the project
  - Database needs to contain a table to use for generating an entity
- Right-click the project in the Project Explorer and select **JPA Tools > Generate Entities**
- Choose the appropriate connection and schema in the **Database Connection** panel
- Select the table to use to create an entity
- Tools automatically generate a class based on the table structure
  - Includes data members and accessor methods







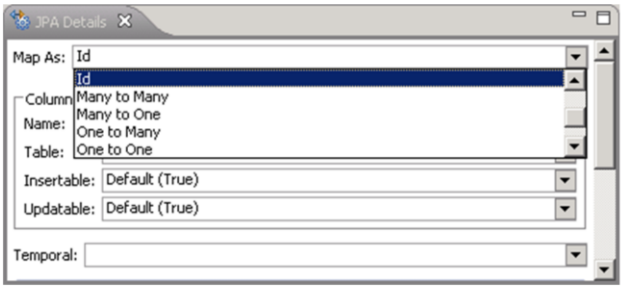
You can generate JPA entity beans from existing database tables; this is called bottom-up mapping. In order to create an entity, you need to configure a database connection for your project and that database needs to contain a table to use to generate the entity. If you have not configured a database connection for your project, you can do so on the Database Connection panel after launching the Generate Entities process for your project. From there, either choose the appropriate existing connection or configure a new connection. Then, select the table to use to create the entity. The development tools will automatically generate a class based on the table structure, and the class will include the appropriate data members and accessor methods. After generating the entity, create a primary key element in the class by using either the Configure JPA Entities tools or by adding the @ID annotation manually to the primary key field in the class.




In the JPA Development perspective, the JPA Details view is integrated with the Annotations view. From this view, you can browse through all of the JPA annotations in your application and configure their attributes. As with other Java EE 5 constructs, in the JPA, annotations take the place of information that was previously provided only in deployment descriptors, and the programming model makes assumptions about what the default attributes should be. For example, if you do not specify a table name to store your JPA entity, the programming model assumes that the class name is the same as the table name. You can see and override these annotation attributes from these views.

# JPA details display – mappings

- The **JPA Details** display is available as a separate view and as a part of the **Annotations** view
- Configure how data is persisted
  - Type of mapping and mapping parameters
    - **Examples:** transient, one-to-many, embedded, many-to-one
    - See documentation for details on available mapping types
  - Column and table in the database

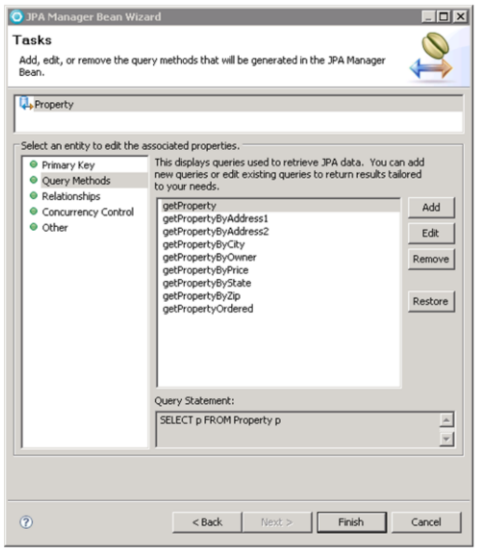






The JPA details view also provides a dropdown menu to configure JPA mapping types and parameters. You can specify the column and table in the database that holds the information and the way that the mapping will happen. For example, you can choose which element is the primary key, if a mapping is one-to-one, one-to-many, or a variety of other mapping types. All of the available types are listed in the mapping menu, and you can read more about them in the product documentation.

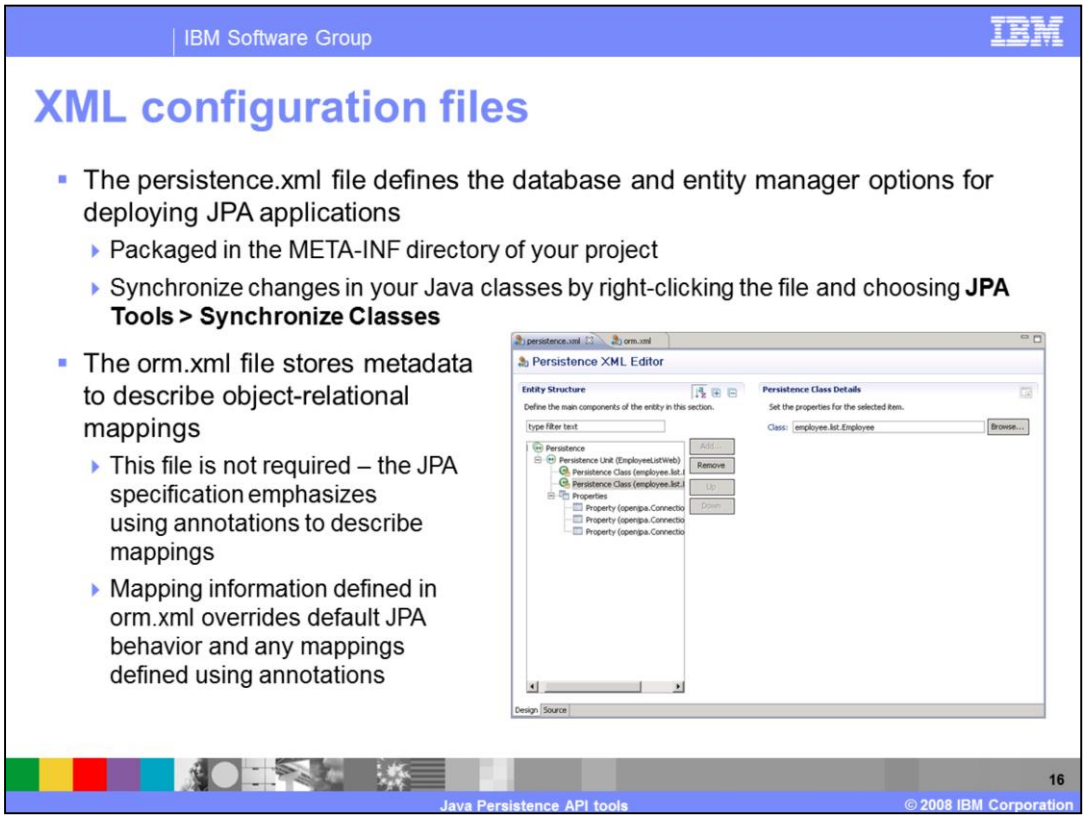
# Creating a JPA manager




- Right-click a JPA project and select **JPA Tools > Add JPA Manager Beans** to open the wizard
  - Choose the entities to associate with the manager
  - Configure tasks for the manager
  - Configure deployment options



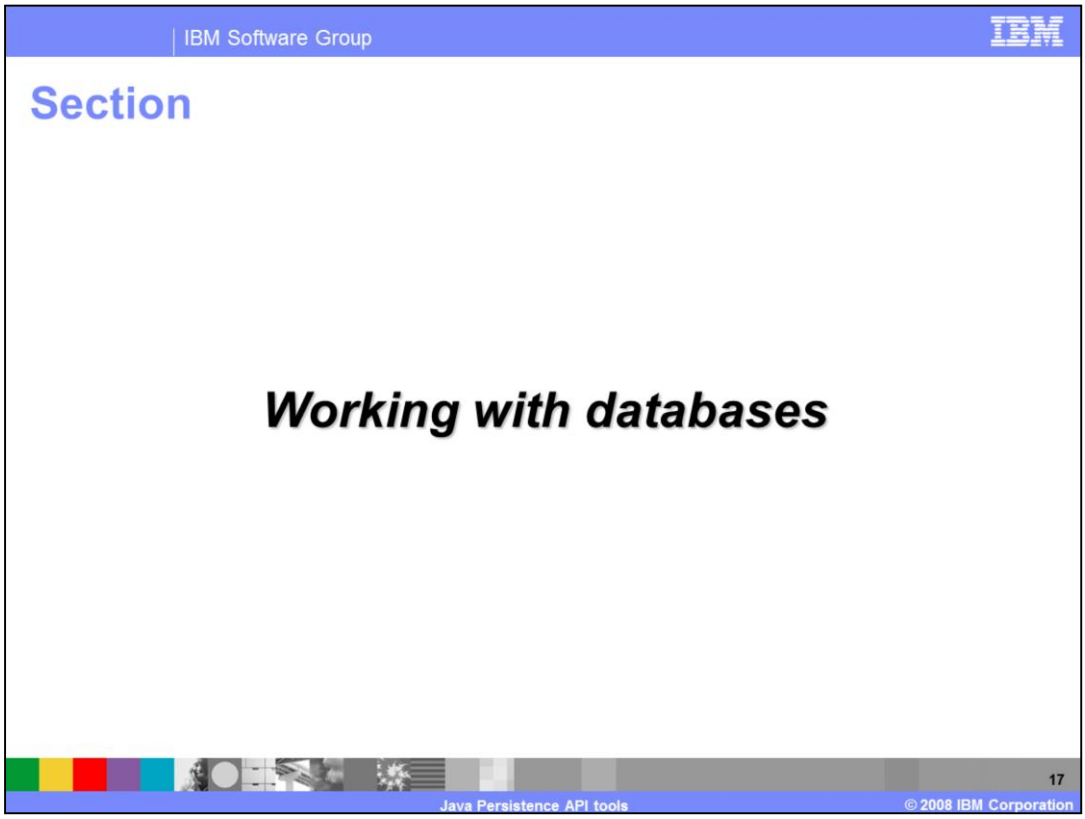

JPA Manager Beans are service beans that act as façades or controllers over a particular JPA entity. They encapsulate and abstract all of the data access code for creating, updating, deleting, and displaying information from your database using JPA entities. JPA Manager Beans are an ideal programming model for use in two-tier Web environments. They fill the role that would normally be filled by a session bean in an EJB environment; all of the business logic related to an entity is performed by the JPA Manager Bean. The use of JPA Manager Beans is not limited to Web applications. They can be used anywhere that you want to take advantage of their data abstraction capabilities, such as an EJB project, a JPA Utility project, or even a plain Java project. Note that if you want to use JPA inside of an EJB project, you might instead want to create an EJB Session Bean to contain all of your JPA logic in order to get the benefits that the EJB container provides. You can automatically generate Java Persistence API (JPA) manager bean classes for JPA entities using the Add JPA Manager Beans wizard. This wizard creates the manager bean and lets you configure which entities to associate with it and what tasks you want the manager to perform.



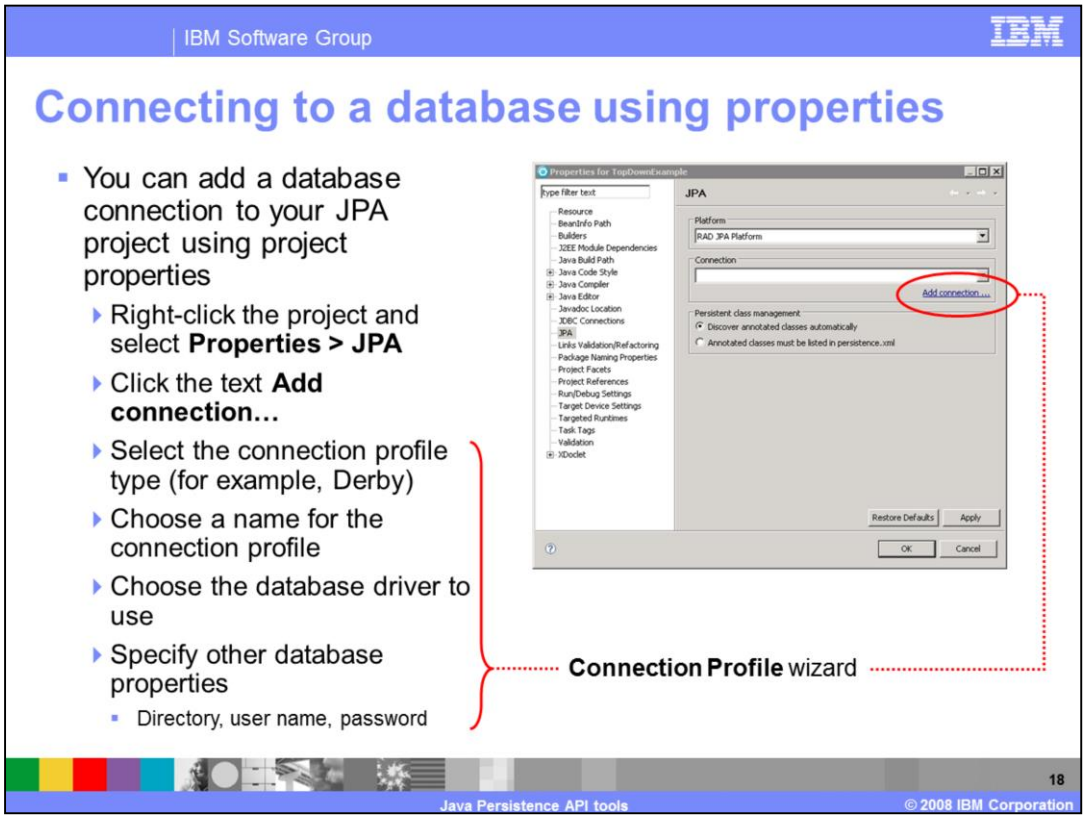

The persistence.xml file describes the details of the persistence units in your JPA project. A persistence unit contains a list of entity beans. You can edit the persistence.xml file with the Persistence XML Editor. The Persistence XML editor simplifies making changes to the persistence.xml file like adding or removing persistence units, or modifying the properties of a persistence unit. After you create persistent classes, you can automatically add them to list of classes in their persistence unit in the persistence.xml file. To automatically add all persistent classes in your JPA project to the persistence.xml file, right-click one of the Java classes and choose the Synchronize Classes option. The persistent classes in your JPA project are automatically discovered and added to the persistence unit in the persistence.xml file.

When you specify object-relational mappings using the JPA tools, the mapping information is contained in the Java class files in the form of Java annotations. However, you can also choose to define the object-relational mappings in XML in a file called orm.xml. Mapping information defined in the orm.xml file automatically overrides both default JPA behavior and any mappings defined using annotations. Therefore, you can, for example, to adapt existing JPA entity beans to a different set of database tables without needing to modify the entity class files. You can use the Object Relational Mapping XML Editor to define object-relational mappings for JPA entity beans in the orm.xml file.

The persistence.xml file and orm.xml file are located in the META-INF directory of the JPA project.



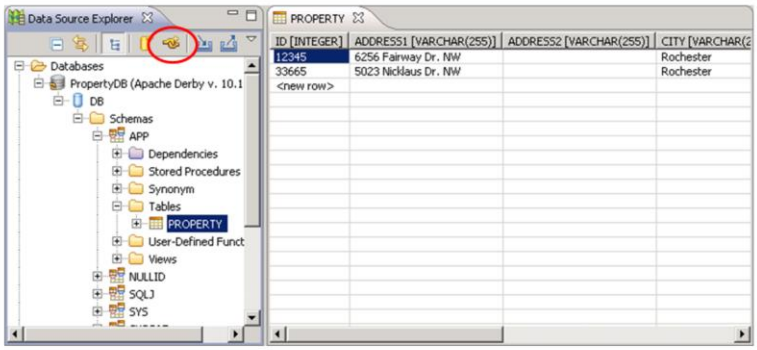

This section describes some additional tools for working with databases that can be useful in developing Java Persistence API applications.




There are multiple ways to access the database connection wizard from the workbench. One way is to create a database connection while you are creating your JPA project. Another way, described here, is to use the JPA project properties to start the wizard. A connection profile contains the connection property information needed to connect to a database runtime instance. You need to choose the type of database being used (for example, Derby or DB2®), a name for the profile, the database driver you are using, and any other required database parameters to create a connection – like the location of the database and user name and password information if the database is secured.

# Using the data source explorer

- Data source explorer is a part of the JPA Development perspective
  - Open the explorer under **Window > Show View > Other > Connectivity > Data Source Explorer**
- Launch the connection profile wizard using **New Connection Profile** icon
- Edit a data table by right-clicking on the table in the **Data Source Explorer** view and choosing **Data > Edit**

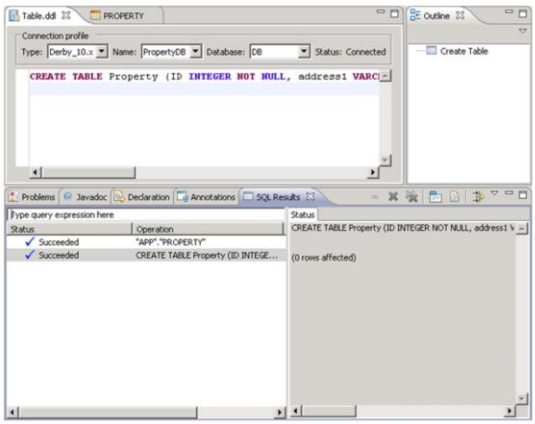






The data source explorer is integrated into the JPA Development perspective, and you can also open it manually under the Window > Show View menu. From the data source explorer, you can create new database connections (using the wizard described on the previous page) and define and edit database tables. To edit a table, right-click it and choose the option to edit the table. The database table entries appear in the table on the right, and you can manually edit them. You can use this editor to populate test data for working with your applications.

# Creating a database table from an entity




- Use JPA entities to create database tables top-down
  - In the Package Explorer view, right-click the JPA project and select **JPA Tools > Generate DDL...**
  - The JPA tools create a file named Table.ddl in the META-INF directory of your JPA project
  - Double-click Table.ddl to open it
  - Configure the connection properties using the dropdown menus at the top of the editor
  - Right-click in the editor and select **Execute All**
  - Check the SQL Results view to see if the table generation was successful

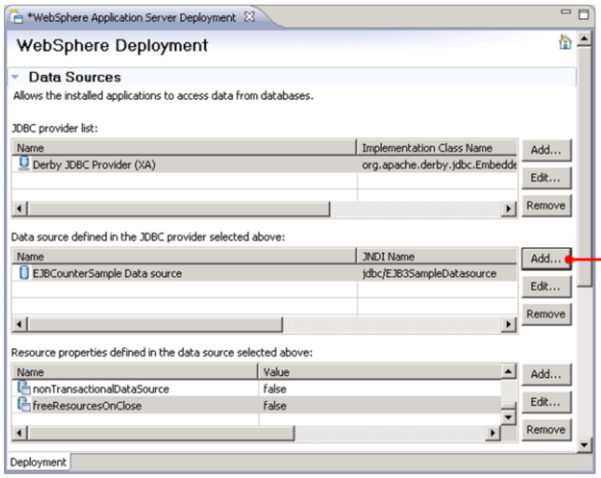



Using the JPA tools, you can generate data definition language files for creating database tables from entity beans that you create. In top-down mapping, you start with entity beans and use them to create your database tables. You start from scratch with the entity definitions and the object-relational mappings, and then you derive database schemas from that data. If you use this approach, you are most likely concerned with creating the architecture of your object model and then writing your entity classes. These entity classes eventually drive the creation of your database model. If you are using a top-down mapping of the object model to the relational model, develop the entity classes, and then use the JPA tools DDL generation capability to create the database tables that are based on the entity classes. The JPA Tools in your project can automatically generate a database definition language, or DDL, file based on one of the entities in your project. Once you have the DDL file, run it from the workbench to create the table.

# Defining an application data source

In the Project Explorer, right-click an EAR project and select **Open WebSphere Application Server Deployment**

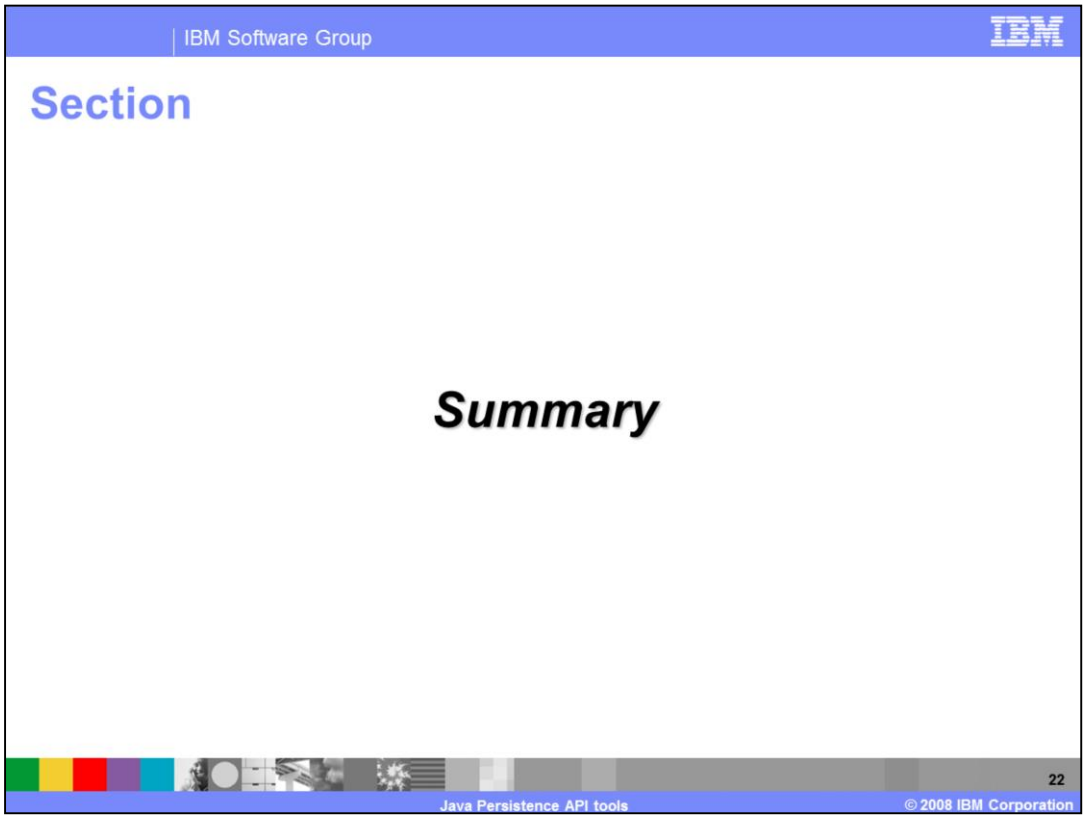
You can configure other server deployment options further down in the display: J2C options, applications to deploy, shared libraries, and more



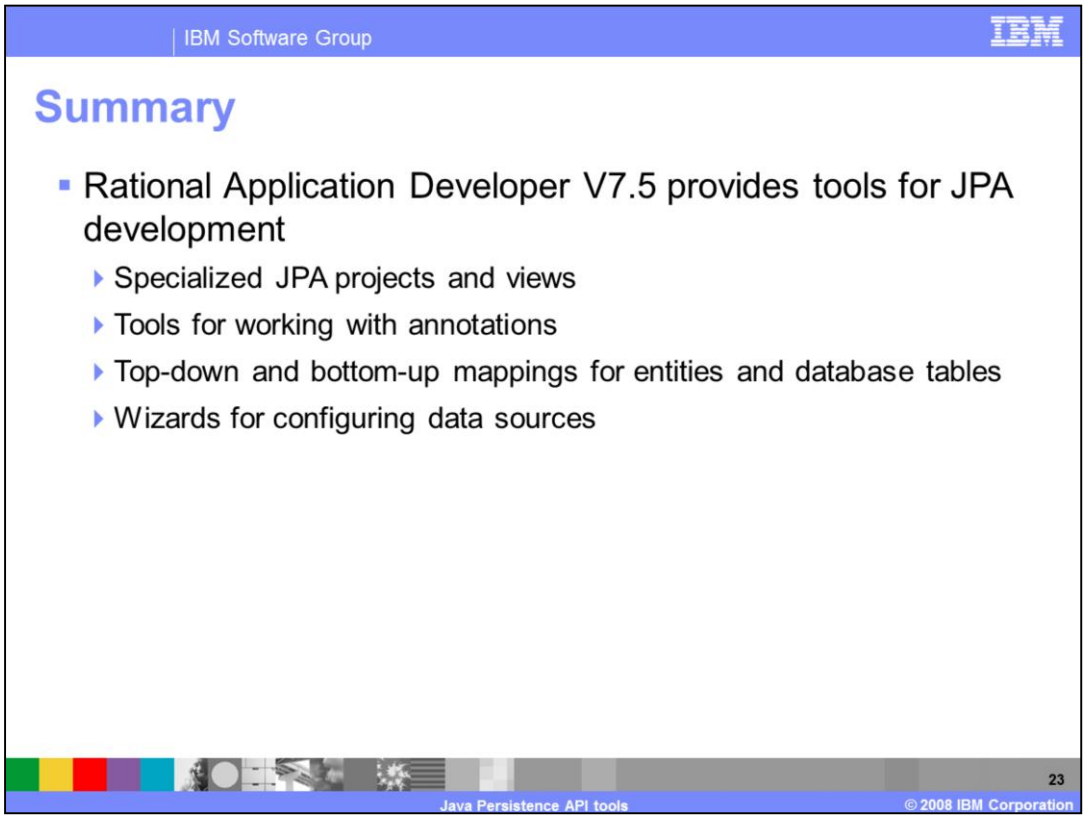

Use wizards to define the data source




If you want to test your application on an application server, like WebSphere Application Server, then you need to define a data source on that server. This assumes that you are working in an EAR project and have a server connection configured for WebSphere Application Server. Right-click the EAR project and choose the Open WebSphere Application Server Deployment option to bring up the WebSphere Deployment editor. From here, you can configure several server deployment options, including data source definitions.




This section contains a summary of the presentation.




The Java Persistence API is a new programming model for persisting Java objects to database; it is part of the Java EE 5 specification. The Rational Application Developer V7.5 workbench provides tools for doing JPA development, including specialized JPA projects and views, tools for working with annotations, the ability to do top-down and bottom-up mapping of entities and database tables, and wizards for configuring data sources for application testing and deployment.

# Trademarks, copyrights, and disclaimers