How-To: Generate a JavaServer Faces 2.x CRUD Application Netbeans 12 and MySQL Database

In this tutorial, you will use the NetBeans IDE to create a web application that interacts with a back-end database. The application provides you with the ability to view and modify data contained in the database - otherwise referred to as *CRUD* (Create, Read, Update, Delete) functionality. The application that you will develop relies on the following technologies.

- **JavaServer Faces (JSF) 2.x** for front-end web pages, validation handling, and management of the request-response cycle.
- **Java Persistence API (JPA) 2.0** using EclipseLink to generate entity classes from the database, and manage transactions. (EclipseLink is the reference implementation for JPA, and is the default persistence provider for the GlassFish server.)
- **Enterprise JavaBeans (EJB) 3.1**, which provides you with stateless EJBs that access the entity classes, and contain the business logic for the application.

The IDE provides two wizards which generate all of the code for the application. The first is the Entity Classes from Database wizard which enables you to generate entity classes from the provided database. After you create entity classes, you use the JSF Pages from Entity Classes wizard to create JSF managed beans and EJBs for the entity classes, as well as a set of Facelets pages to handle the views for entity class data. The final section of the tutorial, Exploring the Application, is optional, and provides numerous exercises to help you to better understand the application and become more familiar with the IDE.

To complete this tutorial, you need the following software and resources.

Software or Resource	Version Required
NetBeans IDE	7.2, 7.3, 7.4, 8.0, Java EE bundle
Java Development Kit (JDK)	7 or 8
GlassFish Server Open Source Edition	3.x, 4.x
mysql-consult.zip (MySQL) *or* javadb-consult.zip (JavaDB)	n/a

Notes:

- The NetBeans IDE Java EE bundle also includes the GlassFish server, a Java EE-compliant server, which you require for this tutorial.
- For the solution project to this tutorial, download ConsultingAgencyJSF20.zip.

Creating the Database

This tutorial uses a consulting agency database called `consult`. The database is not included when you install the IDE so you need to first create the database to follow this tutorial.

The `consult` database was designed to demonstrate the scope of IDE support for handling a variety of database structures. The database is thus not intended as an example of recommended database design or best-practice. Instead, it attempts to incorporate many of the relevant features that are potentially found in a database design. For example, the `consult` database contains all possible relationship types, composite primary keys, and many different data types. See the tables below for a more detailed overview of the database structure.

Notes:

- This tutorial uses the MySQL database server but you can also complete the tutorial using the JavaDB database server. To create the database in JavaDB, download and extract the javadb-consult.zip archive. The archive contains SQL scripts for creating, dropping, and populating the `consult` database.
- For more information on configuring the IDE to work with MySQL, see the Connecting to a MySQL Database tutorial.
- For more information on configuring the IDE to work with JavaDB, see the Working with the Java DB (Derby) Database tutorial.

MySQL with GlassFish Combination:

If you are using MySQL, and are using GlassFish v3 or Open Source Edition 3.0.1, you must ensure that your database is password-protected. (For more information, see GlassFish Issue 12221.) If you are using the default MySQL `root` account with an empty password, you can set the password from a command-line prompt.

For example, to set your password to *`nbuser`*, in a command-line prompt enter the following commands.

```
shell> mysql -u root

mysql> UPDATE mysql.user SET Password = PASSWORD('_nbuser_') WHERE User = 'root';

mysql> FLUSH PRIVILEGES;
```

If you receive a 'mysql: command not found' error, then the `mysql` command has not been added to your `PATH` environment variable. You can instead call the command by entering the full path to your MySQL installation's `bin` directory. For example, if the `mysql` command is located on your computer at `/usr/local/mysql/bin`, enter the following:

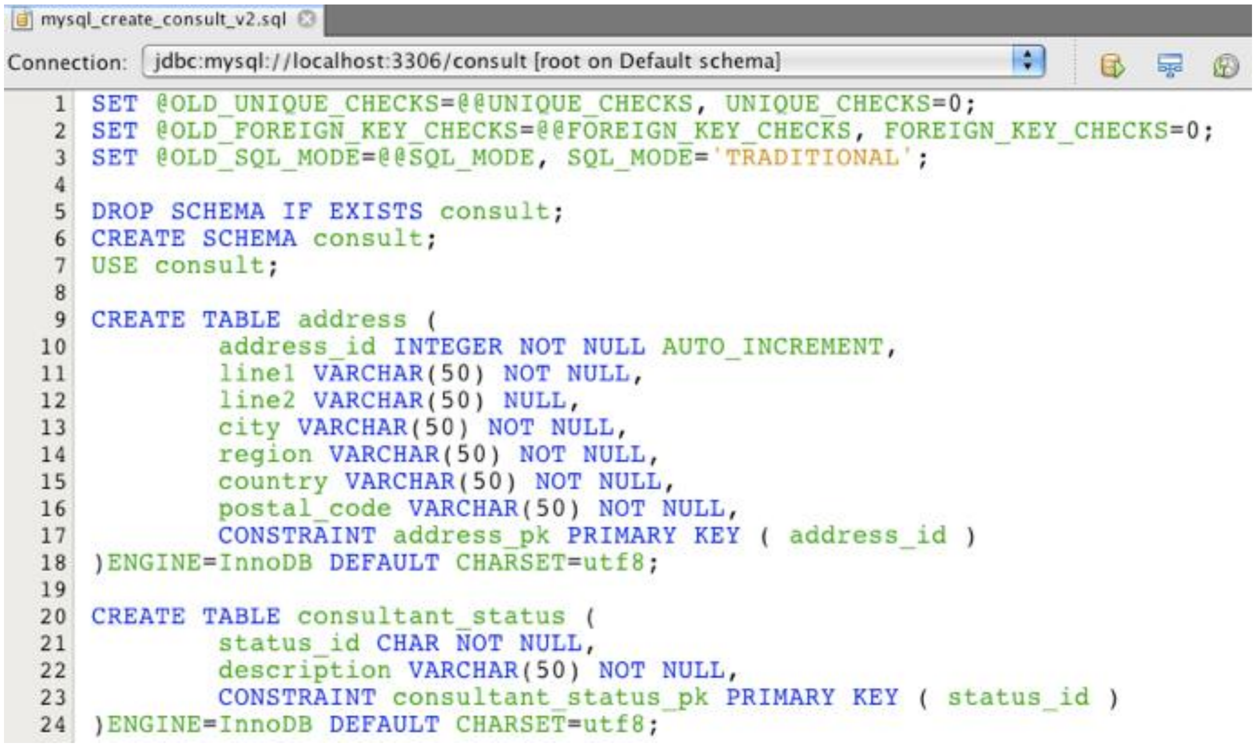
```
shell> /usr/local/mysql/bin/mysql -u root
```

For more information, see the offical MySQL Reference Manual:

- Securing the Initial MySQL Accounts
- 4.2.1. Invoking MySQL Programs
- 4.2.4. Setting Environment Variables

Perform the following steps to create a database and connect to it from the IDE.

1. Download mysql-consult.zip and extract the archive to your local system. When you extract the archive you will see the SQL scripts for creating and populating the database. The archive also has scripts for dropping tables.
2. In the Services window, expand the Databases node, right-click the MySQL node and choose Start Server.
3. Right-click the MySQL Server node and choose Create Database.
4. Type **consult** as the Database Name in the Create MySQL Database dialog. Click OK. A new node appears under the Databases node (`jdbc:mysql://localhost:3306/consult [root on Default schema]`).
5. Right-click the new node and choose Connect.
6. Choose File > Open File from the main menu and navigate to the extracted file `mysql_create_consult.sql`. Click Open. The file automatically opens in the SQL editor.

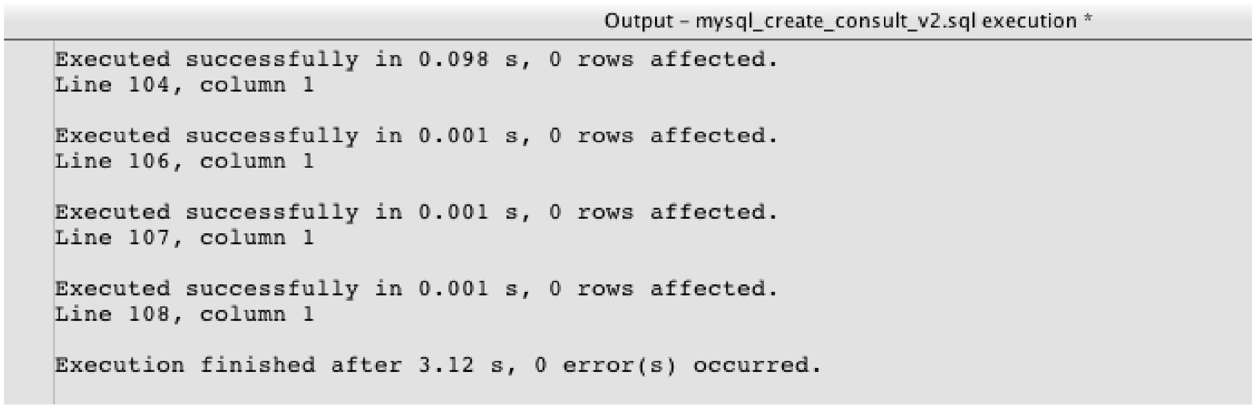

mysql_create_consult_v2.sql
Connection: jdbc:mysql://localhost:3306/consult [root on Default schema]

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

DROP SCHEMA IF EXISTS consult;
CREATE SCHEMA consult;
USE consult;

CREATE TABLE address (
        address_id INTEGER NOT NULL AUTO_INCREMENT,
        line1 VARCHAR(50) NOT NULL,
        line2 VARCHAR(50) NULL,
        city VARCHAR(50) NOT NULL,
        region VARCHAR(50) NOT NULL,
        country VARCHAR(50) NOT NULL,
        postal_code VARCHAR(50) NOT NULL,
        CONSTRAINT address_pk PRIMARY KEY ( address_id )
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE consultant_status (
        status_id CHAR NOT NULL,
        description VARCHAR(50) NOT NULL,
        CONSTRAINT consultant_status_pk PRIMARY KEY ( status_id )
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Figure 1. Open SQL files in the IDE's editor

1. Make sure that the `consult` database is selected in the Connection drop-down list in the SQL editor toolbar, then click the Run SQL ( ) button.

When you click Run SQL, the following output appears in the Output window.

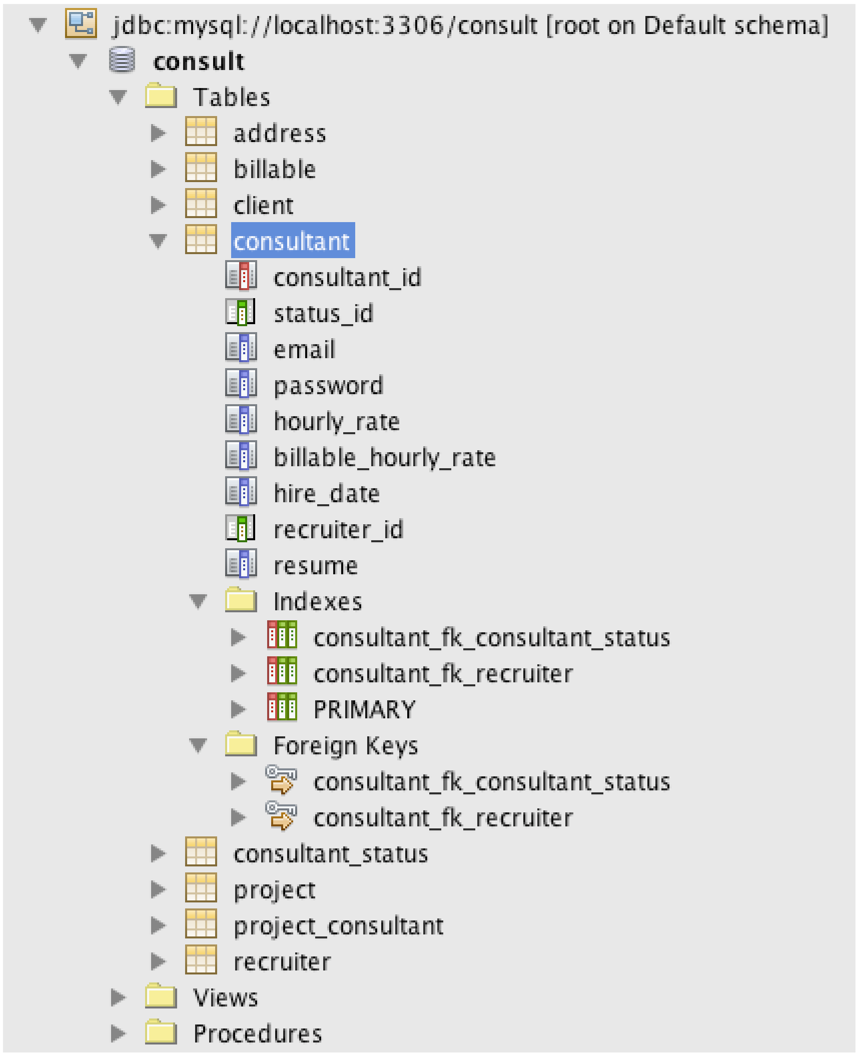


```
Executed successfully in 0.098 s, 0 rows affected.
Line 104, column 1

Executed successfully in 0.001 s, 0 rows affected.
Line 106, column 1

Executed successfully in 0.001 s, 0 rows affected.
Line 107, column 1

Executed successfully in 0.001 s, 0 rows affected.
Line 108, column 1

Execution finished after 3.12 s, 0 error(s) occurred.
```

Figure 2. Output window provides information on SQL execution

Examining the Database Structure

To confirm that the tables were created correctly, expand the Tables node under the database connection node. You can expand a table node to see the table columns, indexes and any foreign keys. You can right-click a column and choose Properties to view additional information about the column.



```
▼ jdbc:mysql://localhost:3306/consult [root on Default schema]
    ▼ consult
        ▼ Tables
            ▶ address
            ▶ billable
            ▶ client
            ▼ consultant
                  consultant_id
                  status_id
                  email
                  password
                  hourly_rate
                  billable_hourly_rate
                  hire_date
                  recruiter_id
                  resume
                ▼ Indexes
                    ▶ consultant_fk_consultant_status
                    ▶ consultant_fk_recruiter
                    ▶ PRIMARY
                ▼ Foreign Keys
                    ▶ consultant_fk_consultant_status
                    ▶ consultant_fk_recruiter
            ▶ consultant_status
            ▶ project
            ▶ project_consultant
            ▶ recruiter
        ▶ Views
        ▶ Procedures
```

Figure 3. Services window displays database connections, tables, table columns, indexes, and foreign keys

Note: If you do not see any tables under the Tables node, right-click the Tables node and choose Refresh.

Looking at the structure of the `consult` database, you can see that the database contains tables that have a variety of relationships and various field types. When creating entity classes from a database, the IDE automatically generates the appropriate code for the various field types.

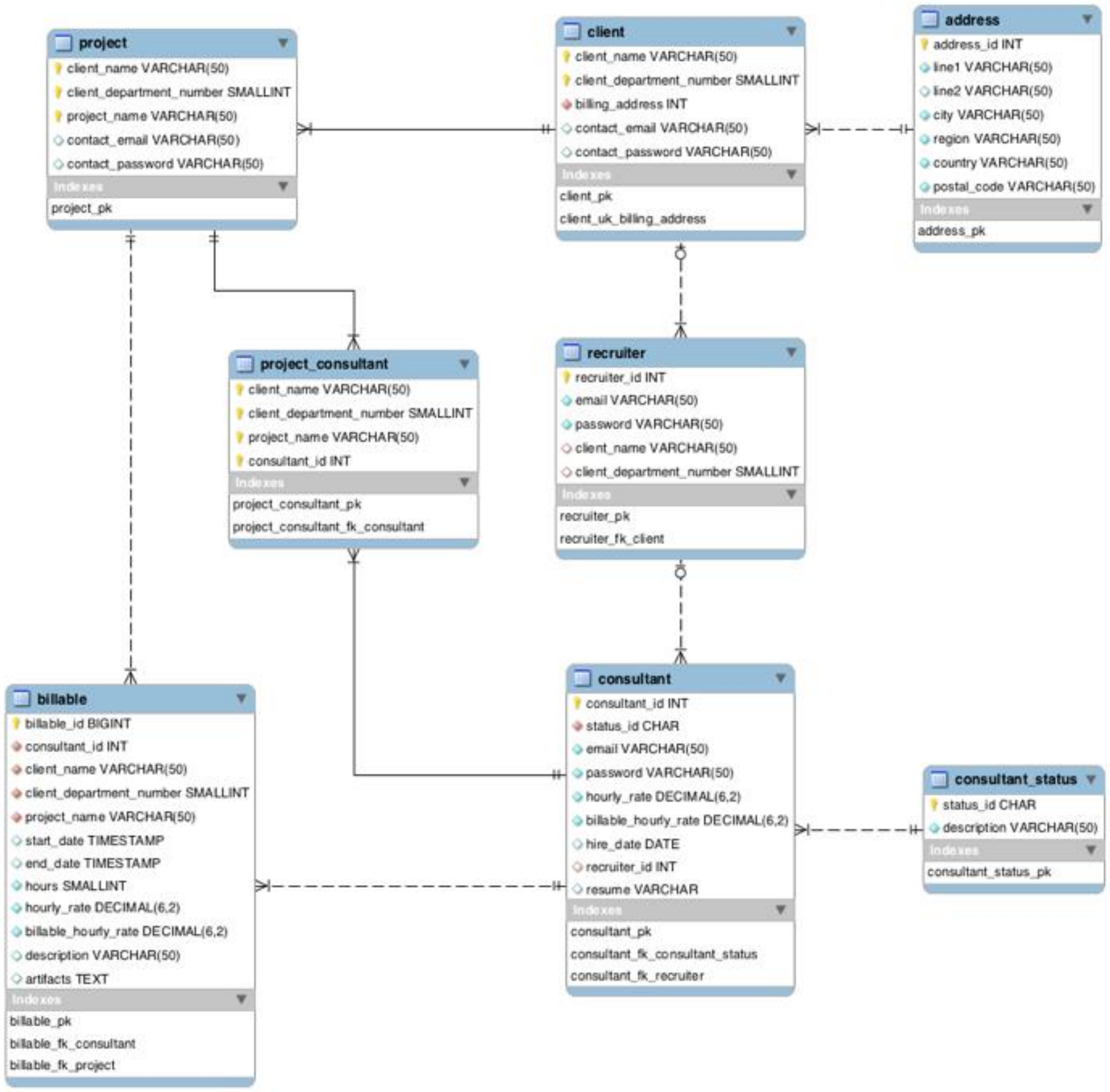


Figure 4. Entity-relationship diagram of consult database

The following table describes the tables found in the `consult` database.

Database Table	**Description**	**Design Features**
CLIENT	A client of the consulting agency	Non-generated, composite primary key (whose fields do not constitute a foreign key)
CONSULTANT	An employee of the consulting agency whom clients can hire on a contract basis	Includes a resume field of type LONG VARCHAR

Database Table	Description	Design Features
CONSULTANT_STATUS	A consultant's status with the consulting agency (for example, Active and Inactive are possible statuses)	Non-generated primary key of type CHAR
RECRUITER	An employee of the consulting agency responsible for connecting clients and consultants	
PROJECT	A project that a client staffs with consultants of the consulting agency	Non-generated, composite primary key that includes two fields constituting a foreign key to the CLIENT table
BILLABLE	A set of hours worked by a consultant on a project, for which the consulting agency bills the relevant client	Includes an artifact field of type CLOB
ADDRESS	A client's billing address	
PROJECT_CONSULTANT	Join table indicating which consultants are currently assigned to which projects	Cross-references PROJECT and CONSULTANT, the former having a composite primary key

The `consult` database includes a variety of relationships. When creating entity classes from a database, the IDE automatically generates the properties of the appropriate Java type based on the SQL type of the columns. The following table describes the entity relationships for the `consult` database. (Inverse relationships are not shown.)

Entity	Related Entity	Relationship Information	Description
CLIENT	RECRUITER	nullable one-to-one with manual editing; nullable one-to-many if not edited	CLIENT has many RECRUITERs and RECRUITER has zero or one CLIENT (if not manually edited)
CLIENT	ADDRESS	non-nullable one-to-one	CLIENT has one ADDRESS and ADDRESS has zero or one CLIENT
CLIENT	PROJECT	non-nullable one-to-many; in a Project entity, the value of the client field is part of the	CLIENT has many PROJECTs and PROJECT has one CLIENT

Entity	**Related Entity**	**Relationship Information**	**Description**
		Project's primary key	
CONSULTANT	PROJECT	many-to-many	CONSULTANT has many PROJECTs and PROJECT has many CONSULTANTs
CONSULTANT	BILLABLE	non-nullable one-to-many	CONSULTANT has many BILLABLEs and BILLABLE has one CONSULTANT
CONSULTANT_STATUS	CONSULTANT	non-nullable one-to-many	CONSULTANT_STATUS has many CONSULTANTs and CONSULTANT has one CONSULTANT_STATUS
CONSULTANT	RECRUITER	nullable one-to-many	CONSULTANT has zero or one RECRUITER and RECRUITER has many CONSULTANTs
BILLABLE	PROJECT	non-nullable one-to-many	BILLABLE has one PROJECT and PROJECT has many BILLABLEs

Now that the database is created, you can create the web application and use the Entity Classes from Database wizard to generate entity classes based on the database tables.

Creating the Web Application Project

In this exercise you create a web project and add the JavaServer Faces framework to the project. When you create the project, you will select JavaServer Faces in the Frameworks panel of the New Project wizard.

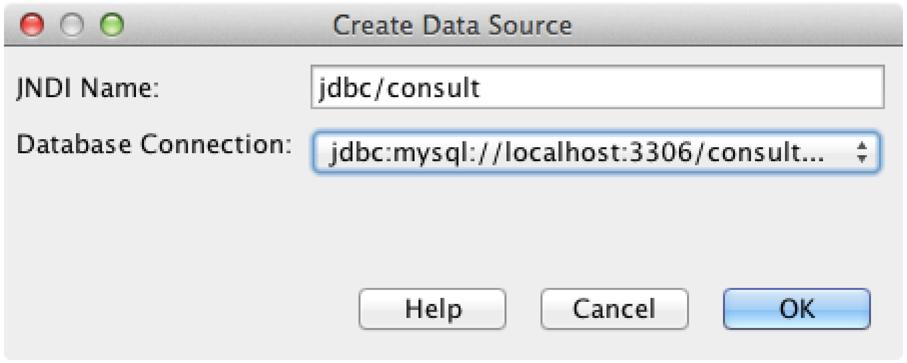
1. Choose File > New Project (Ctrl-Shift-N; ⌘-Shift-N on Mac) from the main menu.
2. Select Web Application from the Java Web category. Click Next.
3. Type `ConsultingAgency` for the project name and set the project location. Click Next.
4. Set the server to GlassFish and set the Java EE Version to Java EE 6 Web or Java EE 7 Web. Click Next.
5. In the Frameworks panel, select the JavaServer Faces option. Click Finish.

When you click Finish, the IDE generates the web application project and opens `index.xhtml` in the editor.

Generating the Entity Classes from the Database

After connecting to a database in the IDE, you can use the Entity Classes from Database wizard to quickly generate entity classes based on the tables in the database. The IDE can generate entity classes for each table that you select, and can also generate any necessary entity classes for related tables.

1. In the Projects window, right-click the `ConsultingAgency` project node, and choose New > Entity Classes from Database. (If this option is not listed, choose Other. Then, in the File wizard, select the Persistence category, then Entity Classes from Database.)
2. Select New Data Source from the Data Source drop-down list to open the Create Data Source dialog.
3. Type `jdbc/consult` as the JNDI Name and select the `jdbc:mysql://localhost:3306/consult` connection as the Database Connection.

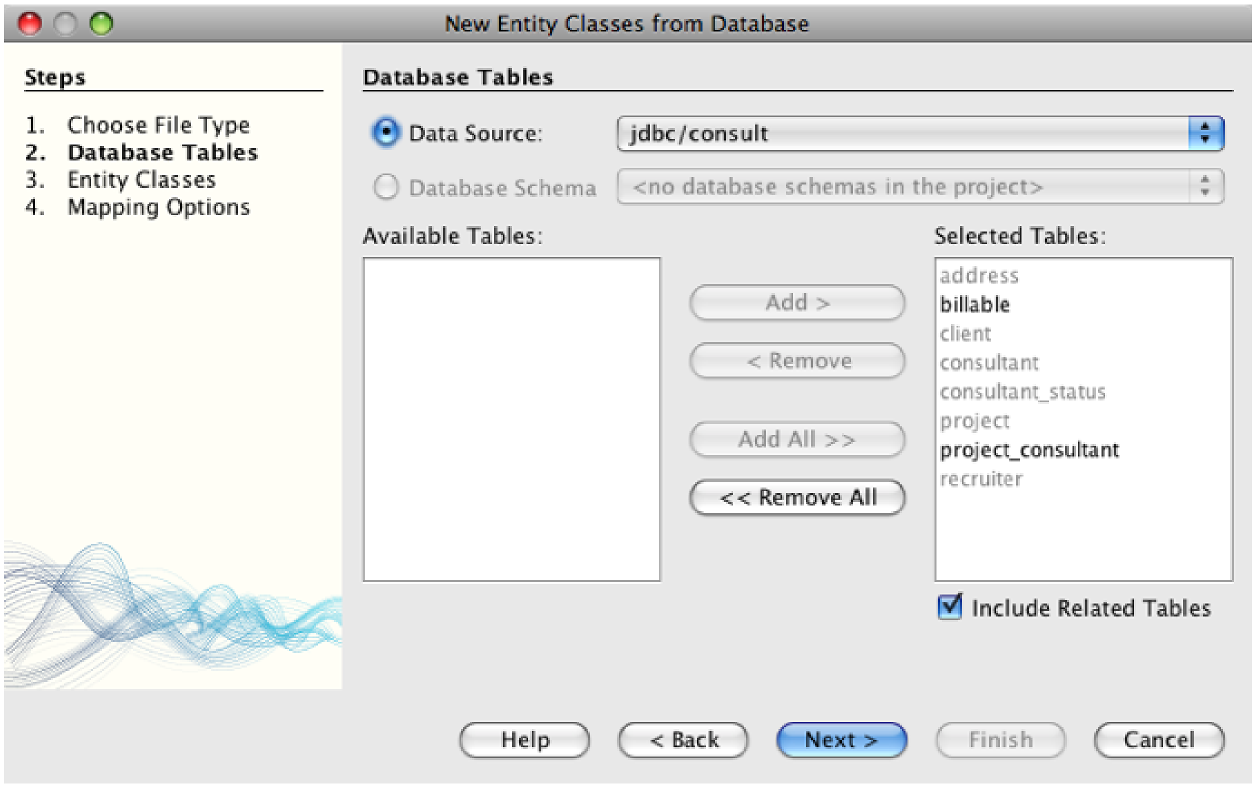

Create Data Source

JNDI Name: jdbc/consult

Database Connection: jdbc:mysql://localhost:3306/consult...

Figure 5. Specify a JNDI name and database connection to create a datasource

1. Click OK to close the dialog box and return to the wizard. The tables in the `consult` database appear in the Available Tables listbox.
2. Click the Add All button to select all tables contained in the database. Click Next.


New Entity Classes from Database

Steps

1. Choose File Type
2. **Database Tables**
3. Entity Classes
4. Mapping Options

Database Tables

Data Source: jdbc/consult

Database Schema <no database schemas in the project>

Available Tables:

Selected Tables:
address
billable
client
consultant
consultant_status
project
project_consultant
recruiter

☑ Include Related Tables

1. Type `jpa.entities` as the Package name.
2. Confirm that the checkboxes to generate named queries and create a persistence unit are selected. Click Finish.

When you click Finish, the IDE generates the entity classes in the `jpa.entities` package of the project.

When using the wizard to create entity classes from a database, the IDE examines the relationships between database tables. In the Projects window, if you expand the `jpa.entities` package node, you can see that the IDE generated an entity class for each table except for the `PROJECT_CONSULTANT` table. The IDE did not create an entity class for `PROJECT_CONSULTANT` because the table is a join table.

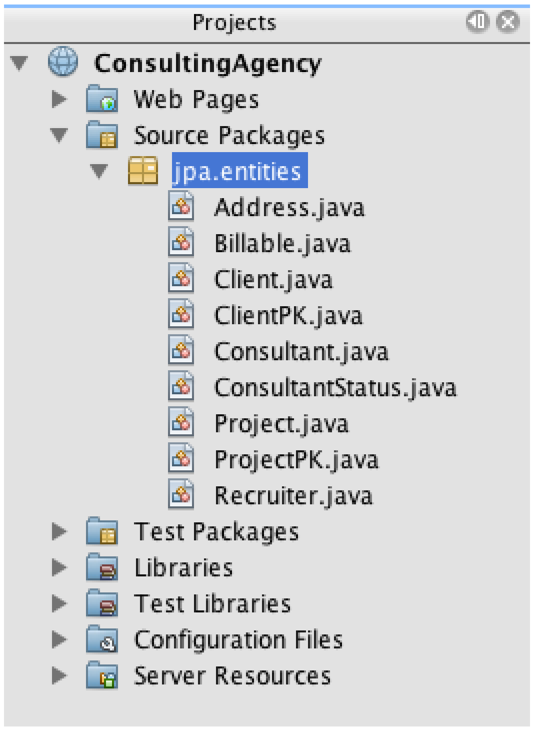


Figure 6. screenshot of Projects window showing generated entity classes

The IDE also generated two additional classes for the tables with composite primary keys: `CLIENT` and `PROJECT`. The primary key classes for these tables (`ClientPK.java` and `ProjectPK.java`) have `PK` appended to the name.

If you look at the generated code for the entity classes you can see that the wizard added `@GeneratedValue` annotations to the auto-generated ID fields and `@Basic(optional = "false")` annotations to some of the fields in the entity classes. Based on the `@Basic(optional = "false")` annotations, the JSF Pages from Entity Classes wizard can generate code that includes checks to prevent non-nullable column violations for those fields.

Generating JSF Pages From Entity Classes

Now that the entity classes are created, you can create the web interface for displaying and modifying the data. You will use the JSF Pages from Entity Classes wizard to generate JavaServer Faces pages. The code generated by the wizard is based on persistence annotations contained in the entity classes.

For each entity class the wizard generates the following files.

- a stateless session bean that extends `AbstractFacade.java`

- a JSF session-scoped, managed bean
- a directory containing four Facelets files for CRUD capabilities (`Create.xhtml`, `Edit.xhtml`, `List.xhtml`, and `View.xhtml`)

The wizard also generates the following files.

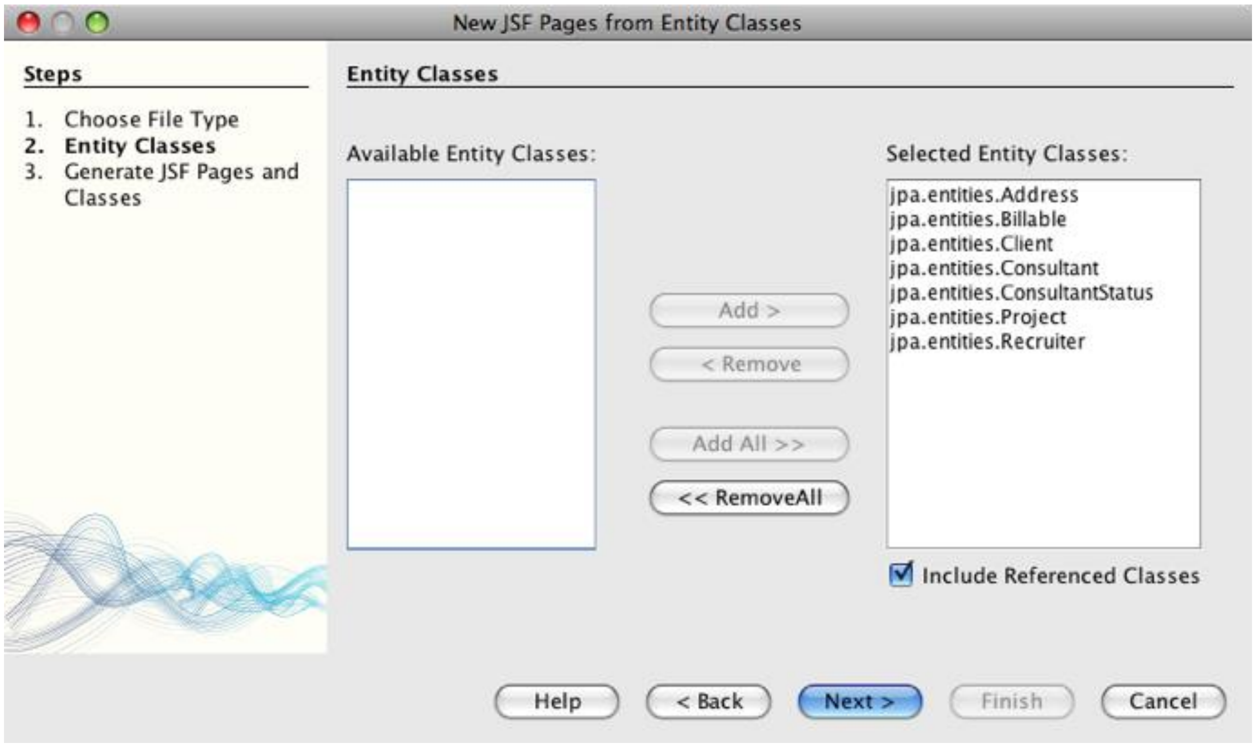
- the `AbstractFacade.java` class that contains the business logic for creation, retrieval, modification and removal of entity instances
- utility classes used by the JSF managed beans (`JsfUtil`, `PaginationHelper`)
- a properties bundle for localized messages, and a corresponding entry in the project's Faces configuration file (A `faces-config.xml` file is created if one does not already exist.)
- auxiliary web files, including a default stylesheet for rendered components, and a Facelets template file

To generate the JSF pages:

1. In the Projects window, right-click the project node and choose New > JSF Pages from Entity Classes to open the wizard. (If this option is not listed, choose Other. Then, in the File wizard, select the JavaServer Faces category, then JSF Pages from Entity Classes.)

The Available Entity Classes box lists the seven entity classes contained in the project. The box does not list the embeddable classes (`ClientPK.java` and `ProjectPK.java`).

1. Click Add All to move all the classes to the Selected Entity Classes box.



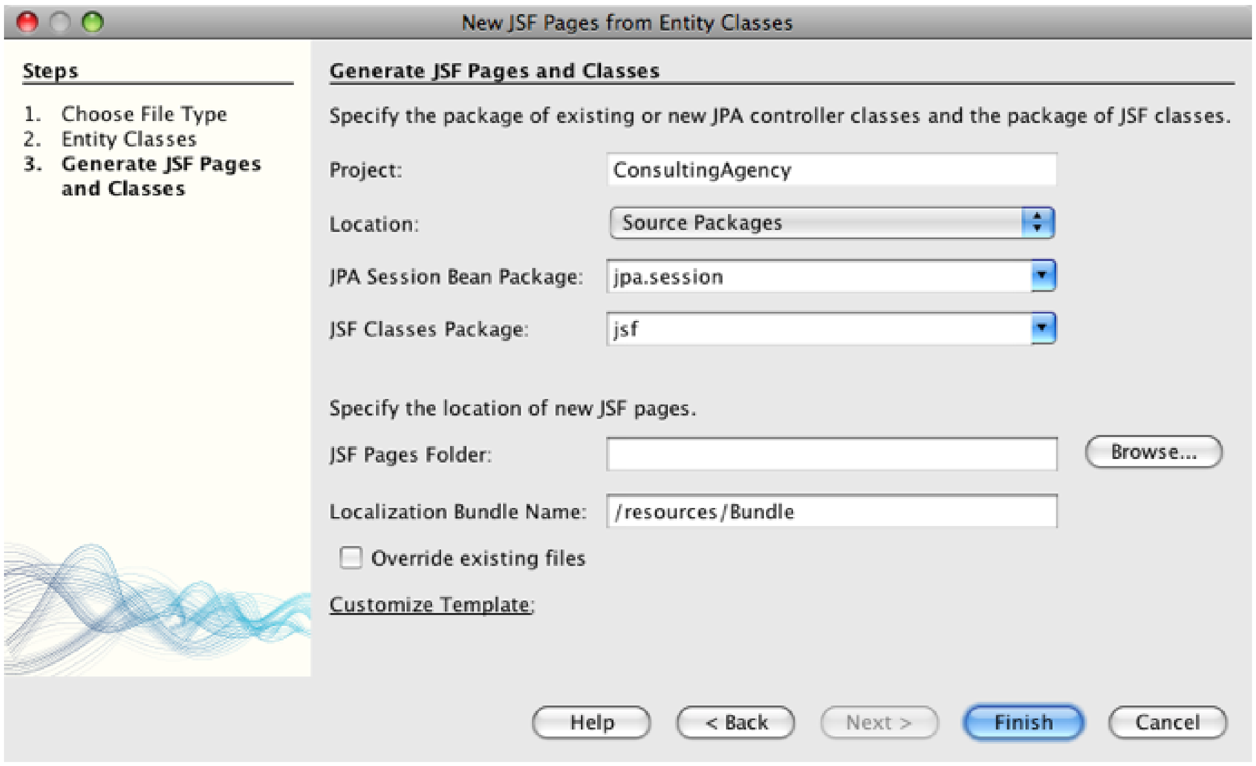
jpa.entities.Address
jpa.entities.Billable
jpa.entities.Client
jpa.entities.Consultant
jpa.entities.ConsultantStatus
jpa.entities.Project
jpa.entities.Recruiter

☑ Include Referenced Classes

Figure 7. New JSF Pages from Entity Classes wizard displays all entity classes contained in project

Click Next.

1. In Step 3 of the wizard, Generate JSF Pages and Classes, type `jpa.session` for the JPA Session Bean Package.
2. Type `jsf` for the JSF Classes Package.
3. Enter '/resources/Bundle' into the Localization Bundle Name field. This will generate a package named `resources` which the `Bundle.properties` file will reside in. (If you leave this blank, the properties bundle will be created in the project's default package.)


New JSF Pages from Entity Classes

Steps

1. Choose File Type
2. Entity Classes
3. **Generate JSF Pages and Classes**

Generate JSF Pages and Classes

Specify the package of existing or new JPA controller classes and the package of JSF classes.

Project: ConsultingAgency

Location: Source Packages

JPA Session Bean Package: jpa.session

JSF Classes Package: jsf

Specify the location of new JSF pages.

JSF Pages Folder:

Localization Bundle Name: /resources/Bundle

☐ Override existing files

Customize Template:

Figure 8. Specify package and folder names for generated files

To let the IDE better accommodate your project conventions, you can customize any files generated by the wizard. Click the Customize Template link to modify the file templates used by the wizard.

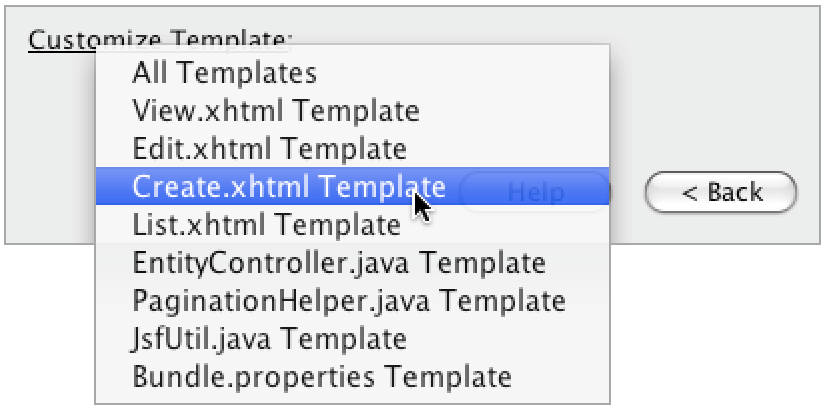


Figure 9. Customize templates for files generated by the wizard

In general, you can access and make changes to all templates maintained by the IDE using the Template Manager (Tools > Templates).

1. Click Finish. The IDE generates the stateless session beans in the `jpa.session` package, and the JSF session-scoped, managed beans in

the `jsf` package. Each stateless session bean handles the operations for the corresponding entity class, including creating, editing, and destroying instances of the entity class via the Java Persistence API. Each JSF managed bean implements the `javax.faces.convert.Converter` interface and performs the work of converting instances of the corresponding entity class to `String` objects and vice versa.

If you expand the Web Pages node, you can see that the IDE generated a folder for each of the entity classes. Each folder contains the files `Create.xhtml`, `Edit.xhtml`, `List.xhtml` and `View.xhtml`. The IDE also modified the `index.xhtml` file by inserting links to each of the `List.xhtml` pages.

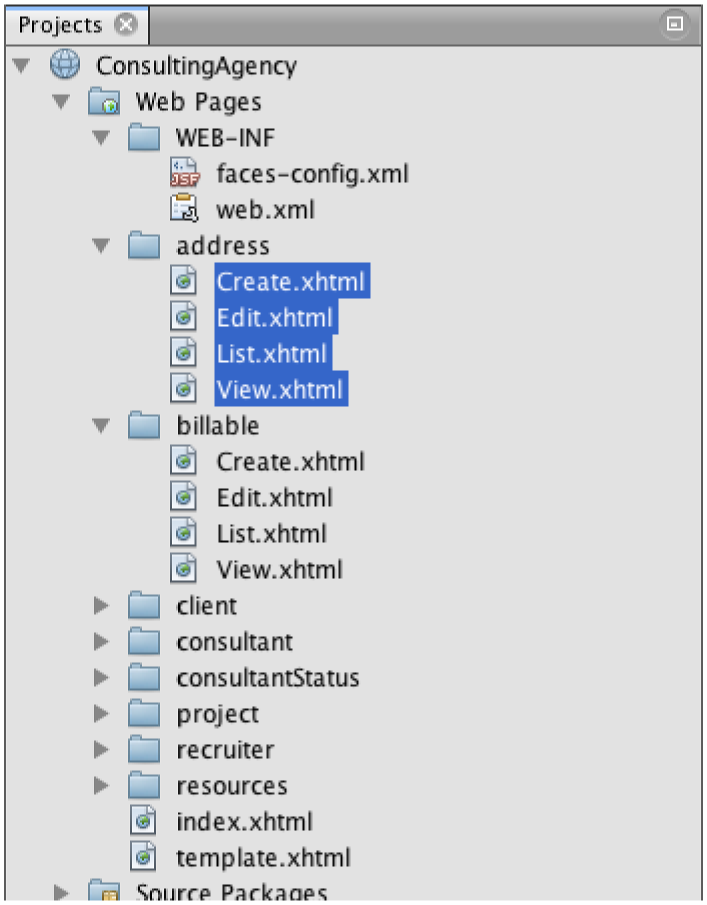


Figure 10. Facelets pages for each entity class are generated by the wizard

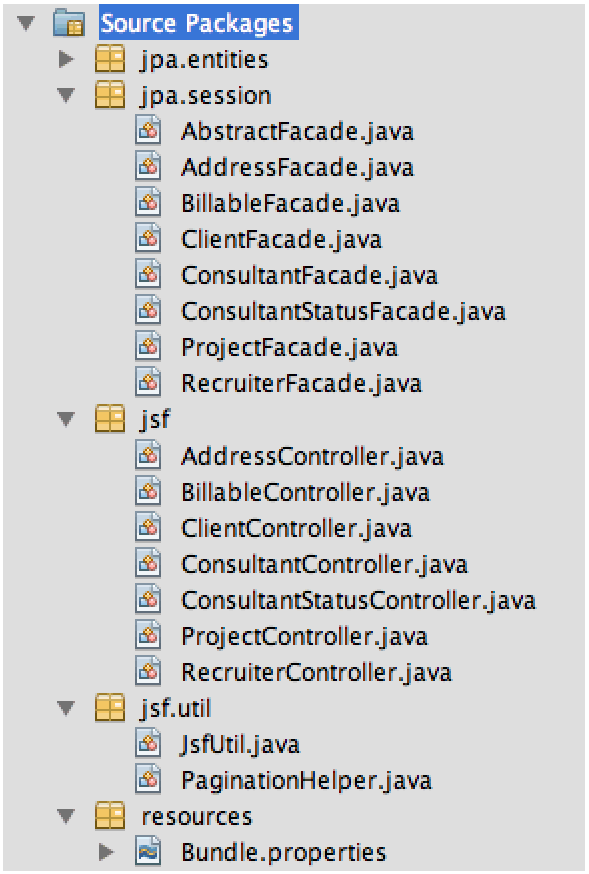
Each JSF managed bean is specific to the four corresponding Facelets files and includes code that invokes methods in the appropriate session bean.

Expand the `resources` folder node to locate the default `jsfcrud.css` stylesheet that was generated by the wizard. If you open the application welcome page (`index.xhtml`) or the Facelets template file (`template.xhtml`) in the editor, you will see that it contains a reference to the stylesheet.

```
<h:outputStylesheet name="css/jsfcrud.css"/>
```

The Facelets template file is used by each of the four Facelets files for each entity class.

If you expand the Source Packages node you can see the session beans, JSF managed beans, utility classes, and properties bundle that the wizard generated.



```
Source Packages
  jpa.entities
  jpa.session
    AbstractFacade.java
    AddressFacade.java
    BillableFacade.java
    ClientFacade.java
    ConsultantFacade.java
    ConsultantStatusFacade.java
    ProjectFacade.java
    RecruiterFacade.java
  jsf
    AddressController.java
    BillableController.java
    ClientController.java
    ConsultantController.java
    ConsultantStatusController.java
    ProjectController.java
    RecruiterController.java
  jsf.util
    JsfUtil.java
    PaginationHelper.java
  resources
    Bundle.properties
```

Figure 11. screenshot of Source Packages directory in Projects window showing classes generated by wizard

The wizard also generated a Faces Configuration file (`faces-config.xml`) in order to register the location of the properties bundle. If you expand the Configuration Files node and open `faces-config.xml` in the XML editor, you can see that the following entry is included.

```
<application>

    <resource-bundle>

        <base-name>/resources/Bundle</base-name>

        <var>bundle</var>

    </resource-bundle>

</application>
```

Also, if you expand the new `resources` package, you'll find the `Bundle.properties` file that contains messages for the client's default language. The messages have been derived from the entity class properties.

To add a new property bundle, right-click the `Bundle.properties` file and choose Customize. The Customizer dialog enables you to add new locales to your application.

Exploring the Application

Now that your project contains entity classes, EJB session beans to control the entity classes, and a JSF-powered front-end to display and modify database, try running the project to see the results.

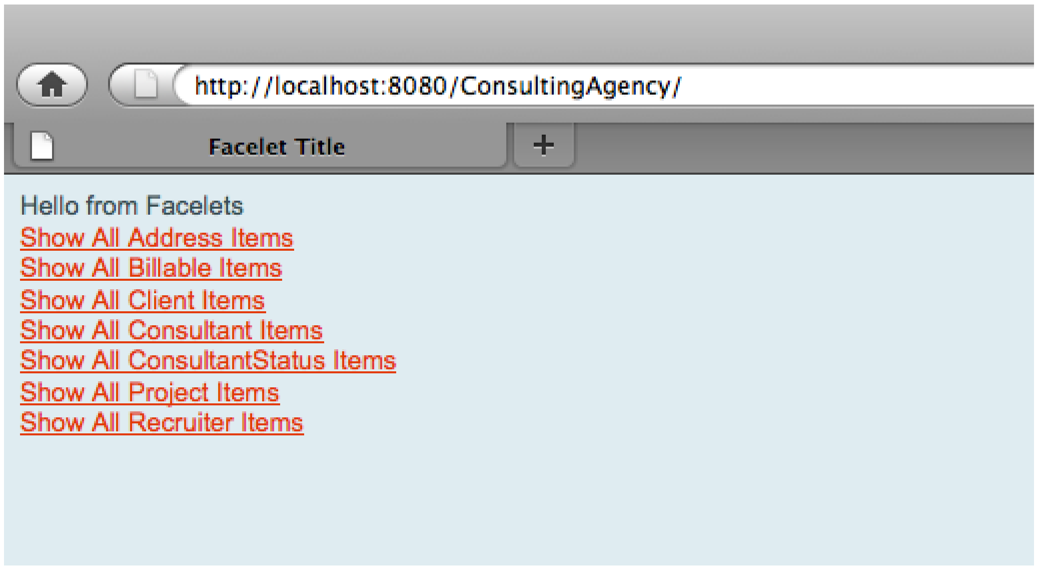
The following is a series of short, optional exercises that help you to become familiar with the application, as well as the features and functionality offered to you by the IDE.

- Examining the Completed Project
- Populating the Database with an SQL Script
- Exploring Editor Support in Facelets Pages
- Exploring Database Integrity with Field Validation
- Editing Entity Classes

Examining the Completed Project

1. To run the project, either right-click the project node in the Projects window and choose Run, or click the Run Project (▷) button in the main toolbar.

When the application's welcome page displays, you are provided with a list of links enabling you to view entries contained in each database table.



Hello from Facelets
Show All Address Items
Show All Billable Items
Show All Client Items
Show All Consultant Items
Show All ConsultantStatus Items
Show All Project Items
Show All Recruiter Items

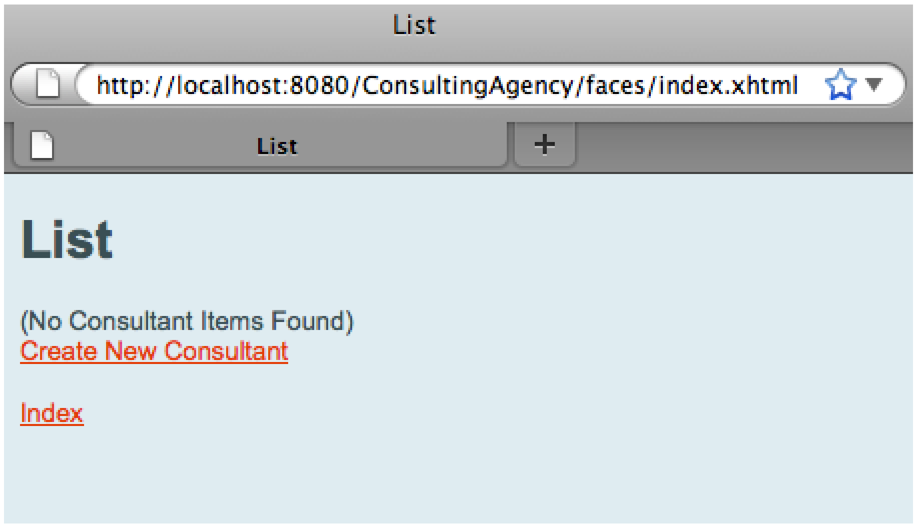
Figure 12. Links to display database contents for each table

The links were added to the welcome page (`index.xhtml`) when you completed the JSF Pages from Entity Classes wizard. They are provided as entry points into the Facelets pages that provide CRUD functionality on the Consulting Agency database.

```
<h:body>

    Hello from Facelets

    <h:form>

        <h:commandLink action="/address/List" value="Show All Address Items"/>

    </h:form>

    <h:form>

        <h:commandLink action="/billable/List" value="Show All Billable Items"/>

    </h:form>

    <h:form>

        <h:commandLink action="/client/List" value="Show All Client Items"/>

    </h:form>

    <h:form>
```

```
        <h:commandLink action="/consultant/List" value="Show All Consultant Items"/>

    </h:form>

    <h:form>

        <h:commandLink action="/consultantStatus/List" value="Show All
ConsultantStatus Items"/>

    </h:form>

    <h:form>

        <h:commandLink action="/project/List" value="Show All Project Items"/>

    </h:form>

    <h:form>

        <h:commandLink action="/recruiter/List" value="Show All Recruiter Items"/>

    </h:form>

</h:body>
```

1. Click the ‘Show All Consultant Items’ link. Looking at the code above, you can see that the target page is `/consultant/List.xhtml`. (In JSF 2.x, the file extension is inferred due to implicit navigation.)



List

(No Consultant Items Found)
Create New Consultant

Index

Figure 13. Consultants table is currently empty

The database currently doesn’t contain any sample data. You can add data manually by clicking the ‘Create New Consultant’ link and using the provided web form. This triggers the `/consultant/Create.xhtml` page to display. You can also run an

SQL script in the IDE to populate tables with sample data. The following sub-sections explore both options.

You can click the index link to return to the links listed in the welcome page. The links provide you with a view of the data held in each database table and trigger the `List.xhtml` file for each entity folder to display. As is later demonstrated, after you add data to the tables, other links will display for each entry enabling you to view (`View.xhtml`), edit (`Edit.xhmtl`), and destroy data for a single table record.

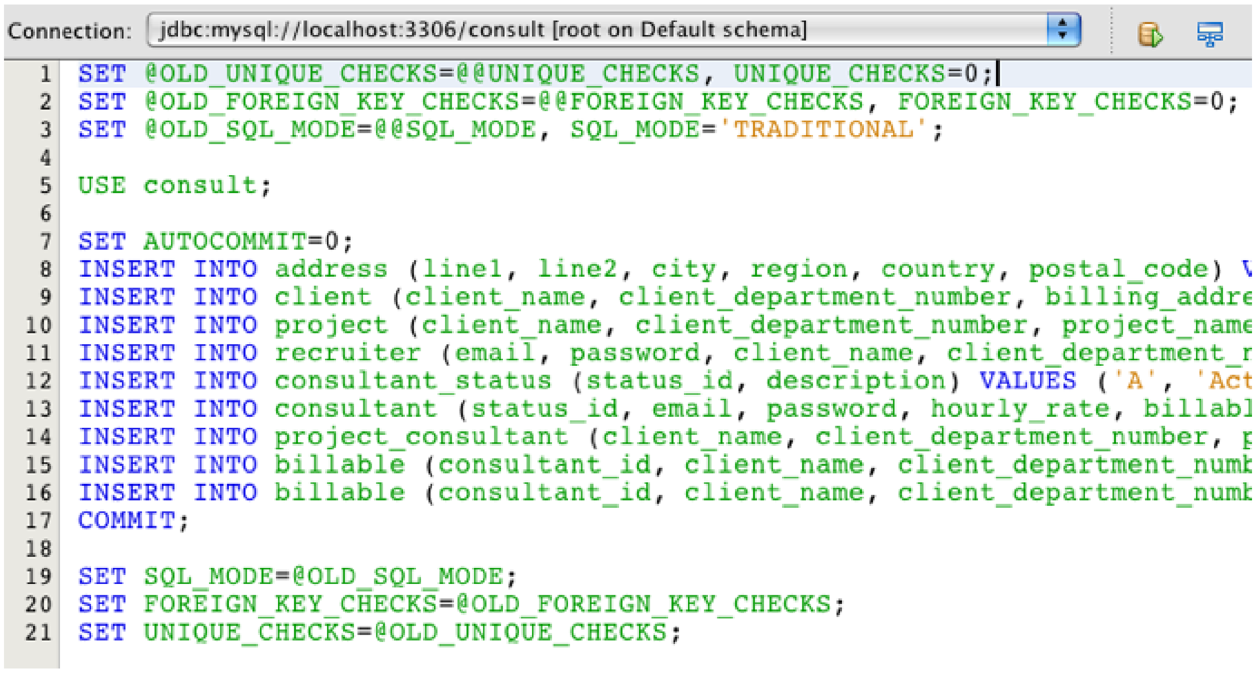
Note. If the application fails to deploy, see the troubleshooting section below. (Also see the troubleshooting section of Creating a Simple Web Application Using a MySQL Database.)

Populating the Database with an SQL Script

Run the provided script, which generates sample data for the database tables. The script (`mysql_insert_data_consult.sql`) is included in the Consulting Agency Database zip file which you can download from the required software table.

Depending on the database server you are working with (MySQL or JavaDB), you can run the provided script, which generates sample data for the database tables. For MySQL, this is the `mysql_insert_data_consult.sql` script. For JavaDB, this is the `javadb_insert_data_consult.sql` script. Both scripts are included in their respective archives, which can be downloaded from the required software table.

1. Choose File > Open File from the main menu, then navigate to the location of the script on your computer. Click Open. The file automatically opens in the IDE's SQL editor.
2. Make sure that the `consult` database is selected in the Connection drop-down list in the SQL editor toolbar.



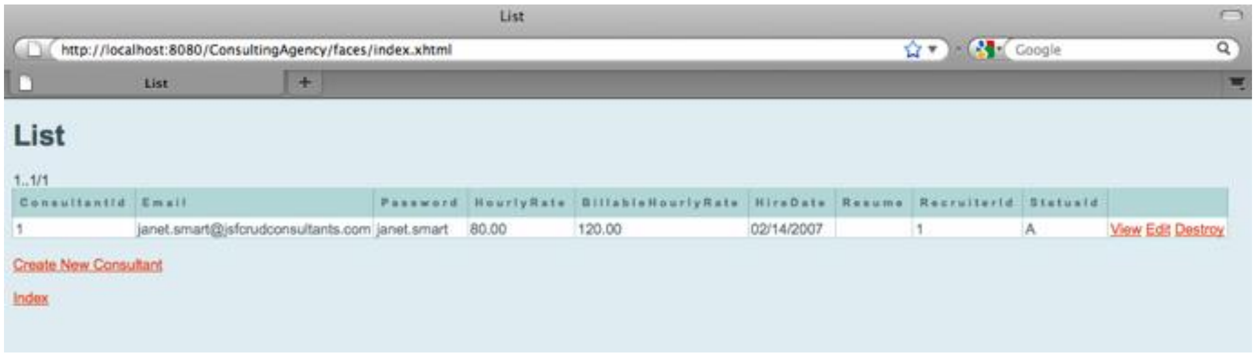
Connection: jdbc:mysql://localhost:3306/consult [root on Default schema]

```sql
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

USE consult;

SET AUTOCOMMIT=0;
INSERT INTO address (line1, line2, city, region, country, postal_code) V
INSERT INTO client (client_name, client_department_number, billing_addre
INSERT INTO project (client_name, client_department_number, project_name
INSERT INTO recruiter (email, password, client_name, client_department_r
INSERT INTO consultant_status (status_id, description) VALUES ('A', 'Act
INSERT INTO consultant (status_id, email, password, hourly_rate, billabl
INSERT INTO project_consultant (client_name, client_department_number, p
INSERT INTO billable (consultant_id, client_name, client_department_numb
INSERT INTO billable (consultant_id, client_name, client_department_numb
COMMIT;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Figure 14. Open the script in the IDE's SQL editor

Either right-click in the editor and choose Run Statement, or click the Run SQL ( ) button. You can see the result of the script execution in the Output window.

1. Restart the GlassFish server. This is a necessary step to enable the server to reload and cache the new data contained in the `consult` database. To do so, click the GlassFish server tab in the Output window (The GlassFish server tab displays the server log.), then click the Restart Server ( ) button in the left margin. The server stops, then restarts.
2. Run the project again and click the 'Show All Consultant Items' link. You will see that the list is no longer empty.



List

http://localhost:8080/ConsultingAgency/faces/index.xhtml

List

1..1/1

ConsultantId	Email	Password	HourlyRate	BillableHourlyRate	HireDate	Resume	RecruiterId	StatusId	
1	janet.smart@jsfcrudconsultants.com	janet.smart	80.00	120.00	02/14/2007		1	A	View Edit Destroy

Create New Consultant

Index

NetBeans Database Support

You can use the IDE's database table viewer to display and modify table data maintained directly in the database. For example, right-click the `consultant` table in the Services window, and choose View Data.

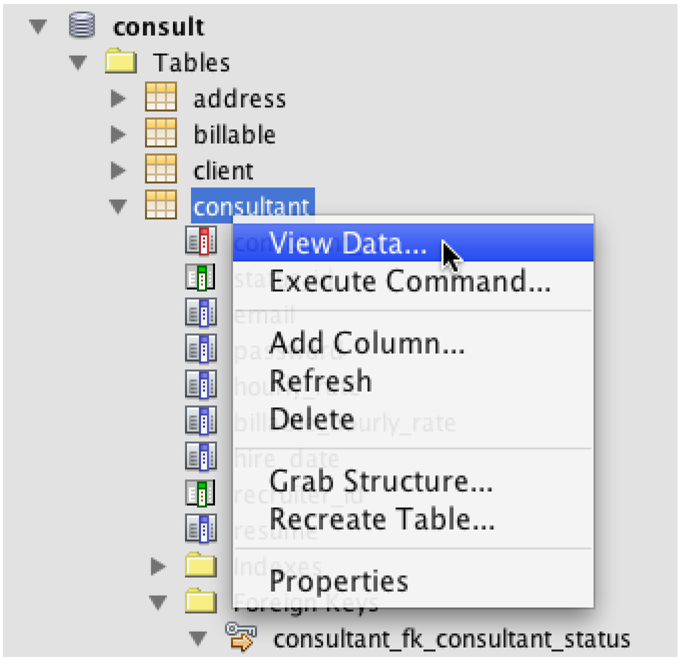
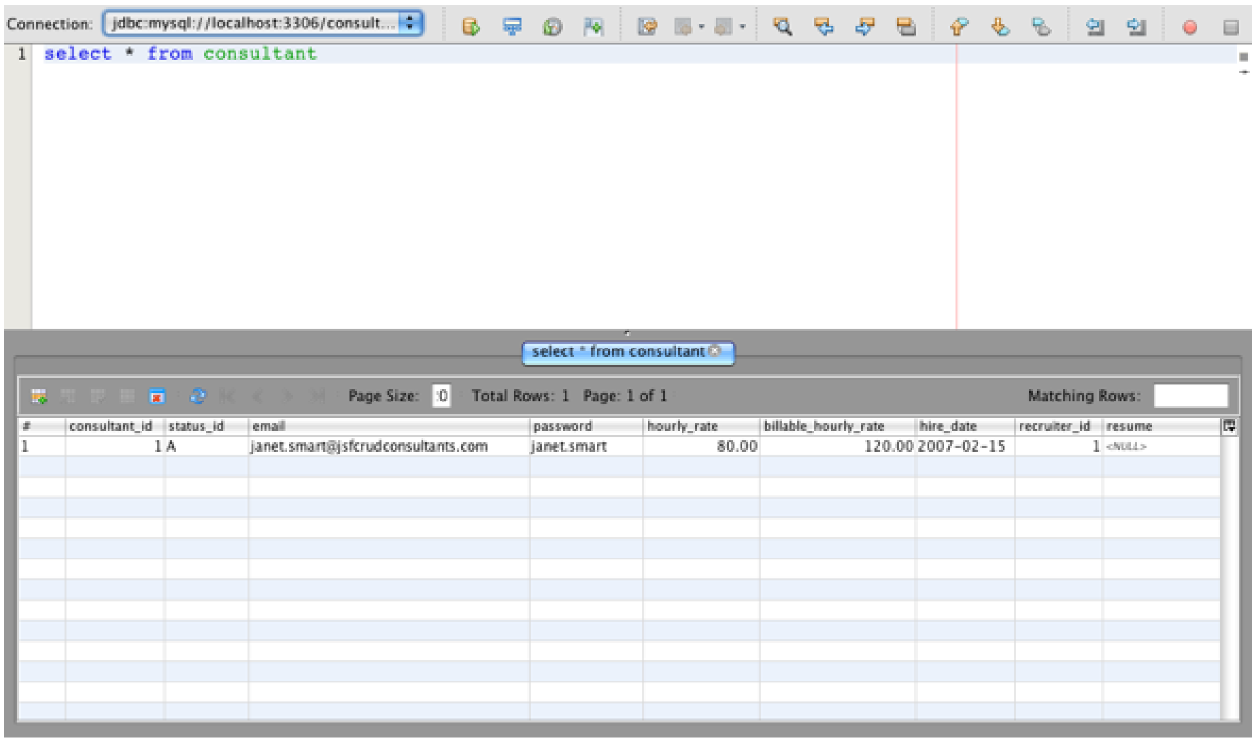


Figure 15. Choose View Data from the right-click menu of database tables

The SQL query used to perform the action displays in the upper portion of the editor, and a graphical view of the table displays beneath.



Connection: jdbc:mysql://localhost:3306/consult...

```sql
select * from consultant
```

select * from consultant

Page Size: 0 Total Rows: 1 Page: 1 of 1 Matching Rows:

#	consultant_id	status_id	email	password	hourly_rate	billable_hourly_rate	hire_date	recruiter_id	resume
1	1	A	janet.smart@jsfcrudconsultants.com	janet.smart	80.00	120.00	2007-02-15	1	<NULL>

Double-click inside table cells to perform inline modifications to data. Click the Commit Records ( ) icon to commit changes to the database.

The graphical view provides much more functionality. See Database Support in NetBeans IDE for more information.

Exploring Editor Support in Facelets Pages

1. Open the `/consultant/List.xhtml` page in the editor. Line 8 indicates that the page relies on the Facelets `template.xhtml` file to render.

```
<ui:composition template="/template.xhtml">
```

To display line numbers, right-click in the editor's left margin and choose Show Line Numbers.

1. Use the IDE's Go to File dialog to open `template.xhtml`. Press Alt-Shift-O (Ctrl-Shift-O on Mac), then begin typing `template`.

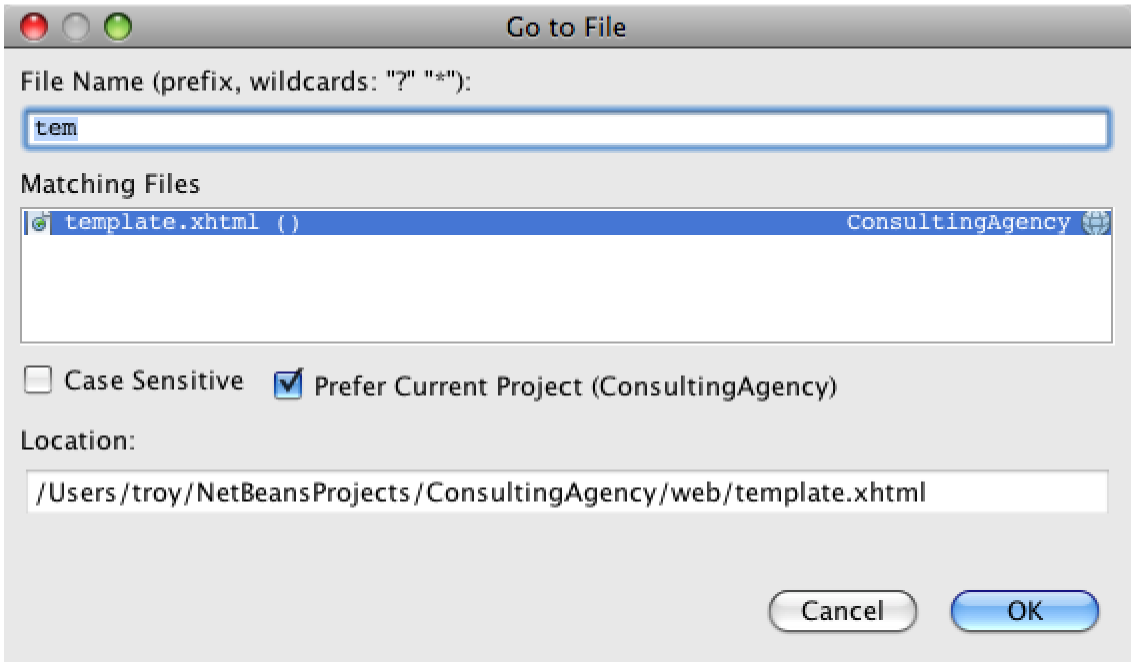
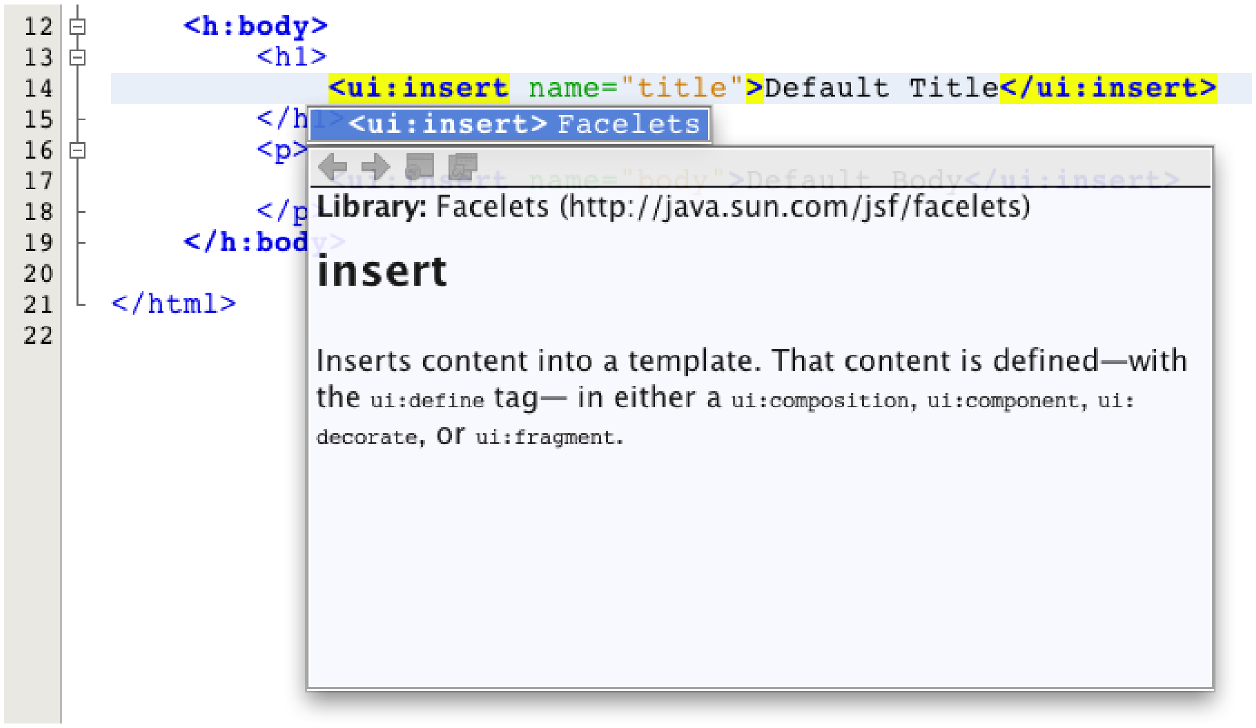

Go to File
File Name (prefix, wildcards: "?" "*"):
tem
Matching Files
template.xhtml () ConsultingAgency
Case Sensitive
Prefer Current Project (ConsultingAgency)
Location:
/Users/troy/NetBeansProjects/ConsultingAgency/web/template.xhtml
Cancel
OK

Figure 16. Use the Go to File dialog to quickly open project files

Click OK (or press Enter).

1. The template applies the `<ui:insert>` tags to insert content from other files into its title and body. Place your cursor on the `<ui:insert>` tag, then press Ctrl-Space to invoke a documentation popup window.



```
    <h:body>
        <h1>
            <ui:insert name="title">Default Title</ui:insert>
        </h
        <p>

        </p
    </h:bod
</html>
```

Library: Facelets (http://java.sun.com/jsf/facelets)

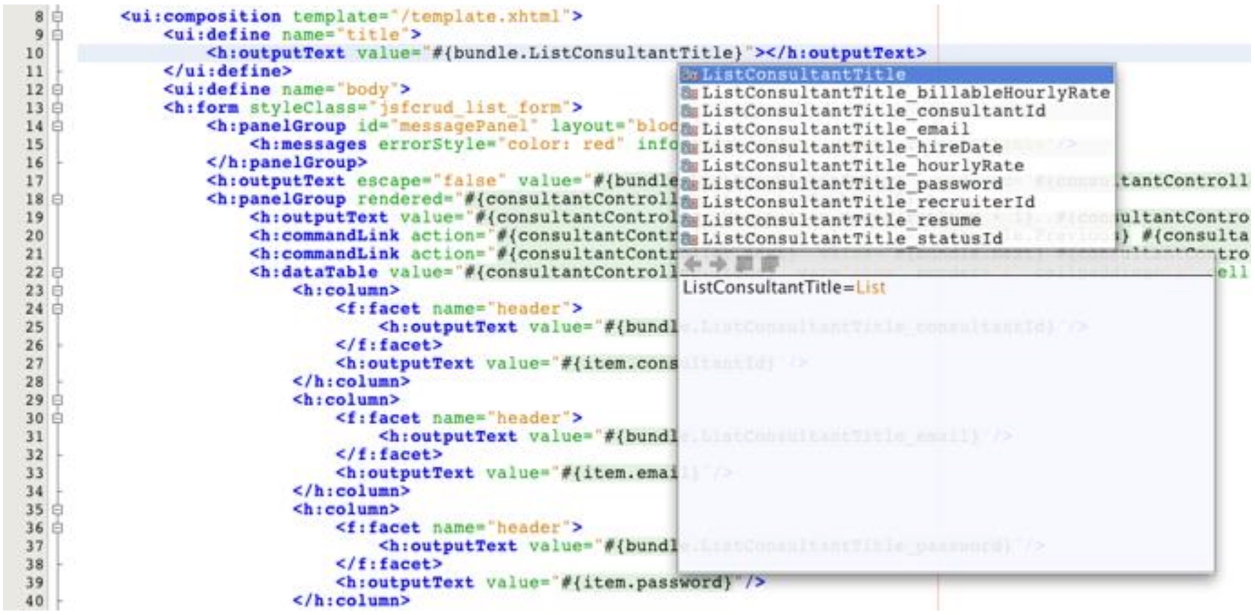
insert

Inserts content into a template. That content is defined—with the `ui:define` tag— in either a `ui:composition`, `ui:component`, `ui:decorate`, or `ui:fragment`.

Figure 17. Press Ctrl-Space to invoke a documentation popup on Facelets tags

You can press Ctrl-Space on JSF tags and their attributes to invoke a documentation pop-up. The documentation you see is taken from the descriptions provided in the official JSF Tag Library Documentation.

1. Switch back to the `List.xhtml` file (press Ctrl-Tab). The `<ui:define>` tags are used to define the content that will be applied to the template's title and body. This pattern is used for all four Facelets files (`Create.xhtml`, `Edit.xhtml`, `List.xhtml`, and `View.xhtml`) generated for each entity class.
2. Place your cursor on any of the EL expressions used for localized messages contained in the `Bundle.properties` file. Press Ctrl-Space to view the localized message.



```
<ui:composition template="/template.xhtml">
    <ui:define name="title">
        <h:outputText value="#{bundle.ListConsultantTitle}"></h:outputText>
    </ui:define>
    <ui:define name="body">
    <h:form styleClass="jsfcrud_list_form">
        <h:panelGroup id="messagePanel" layout="bloc
            <h:messages errorStyle="color: red" info
        </h:panelGroup>
        <h:outputText escape="false" value="#{bundle
        <h:panelGroup rendered="#{consultantControll
            <h:outputText value="#{consultantControl
            <h:commandLink action="#{consultantContr
            <h:commandLink action="#{consultantContr
            <h:dataTable value="#{consultantControll
                <h:column>
                    <f:facet name="header">
                        <h:outputText value="#{bundl
                    </f:facet>
                    <h:outputText value="#{item.cons
                </h:column>
                <h:column>
                    <f:facet name="header">
                        <h:outputText value="#{bundl
                    </f:facet>
                    <h:outputText value="#{item.emai
                </h:column>
                <h:column>
                    <f:facet name="header">
                        <h:outputText value="#{bundl
                    </f:facet>
                    <h:outputText value="#{item.password}"/>
                </h:column>
```

```
ListConsultantTitle
ListConsultantTitle_billableHourlyRate
ListConsultantTitle_consultantId
ListConsultantTitle_email
ListConsultantTitle_hireDate
ListConsultantTitle_hourlyRate
ListConsultantTitle_password
ListConsultantTitle_recruiterId
ListConsultantTitle_resume
ListConsultantTitle_statusId
```

ListConsultantTitle=List

In the above image, you can see that the EL expression resolves to 'List', which is applied to the template title and can be verified from the page rendered in the browser.

1. Scroll to the bottom of the file and locate the code for the `Create New Consultant` link (Line 92). This is as follows:

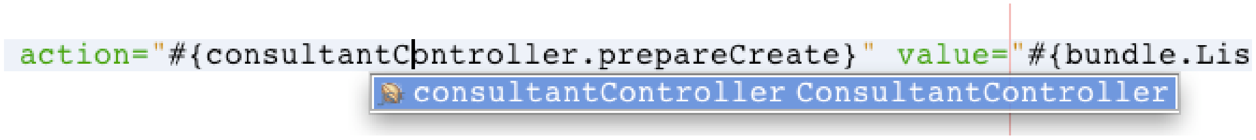
```
<h:commandLink action="#{consultantController.prepareCreate}"
value="#{bundle.ListConsultantCreateLink}"/>
```

1. Press Ctrl-Space on the `commandLink's `action` attribute to invoke the documentation pop-up.

The `action` attribute indicates the method that handles the request when the link is clicked in the browser. The following documentation is provided:

MethodExpression representing the application action to invoke when this component is activated by the user. The expression must evaluate to a public method that takes no parameters, and returns an Object (the toString() of which is called to derive the logical outcome) which is passed to the NavigationHandler for this application. In other words, the `action` value typically refers to a method in a JSF managed bean that evaluates to a `String`. The string is then used by JSF's `NavigationHandler` to forward the request to the appropriate view. You verify this in the following steps.

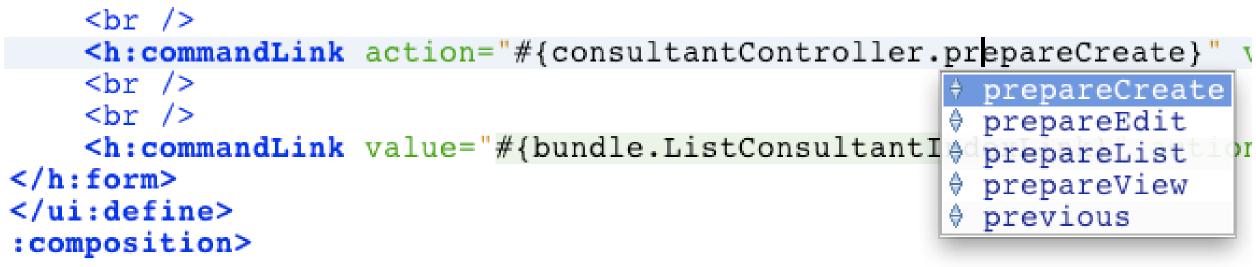
1. Place your cursor on `consultantController` and press Ctrl-Space. The editor's code completion indicates that `consultantController` is a JSF managed bean.



```
action="#{consultantController.prepareCreate}" value="#{bundle.Lis
                          consultantController ConsultantController
```

Figure 18. Code completion is provided for JSF managed beans

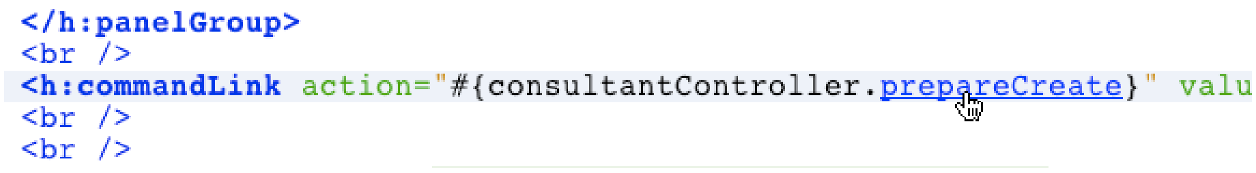
1. Move your cursor to `prepareCreate` and press Ctrl-Space. Code completion lists methods contained in the `ConsultantController` managed bean.



```
    <br />
    <h:commandLink action="#{consultantController.prepareCreate}" v
    <br />                                          prepareCreate
    <br />                                          prepareEdit
    <h:commandLink value="#{bundle.ListConsultantI  prepareList
</h:form>                                           prepareView
</ui:define>                                        previous
:composition>
```

Figure 19. Code completion is provided for class methods

1. Press Ctrl (⌘ on Mac), then hover your mouse over `prepareCreate`. A link is formed, enabling you to navigate directly to the `prepareCreate()` method in the `ConsultantController` managed bean.



```
</h:panelGroup>
<br />
<h:commandLink action="#{consultantController.prepareCreate}" valu
<br />
<br />
```

Figure 20. Use editor navigation to quickly navigate source code

1. Click the link and view the `prepareCreate()` method (displayed below).

```
public String prepareCreate() {

    current = new Consultant();

    selectedItemIndex = -1;
    return "Create";

}
```

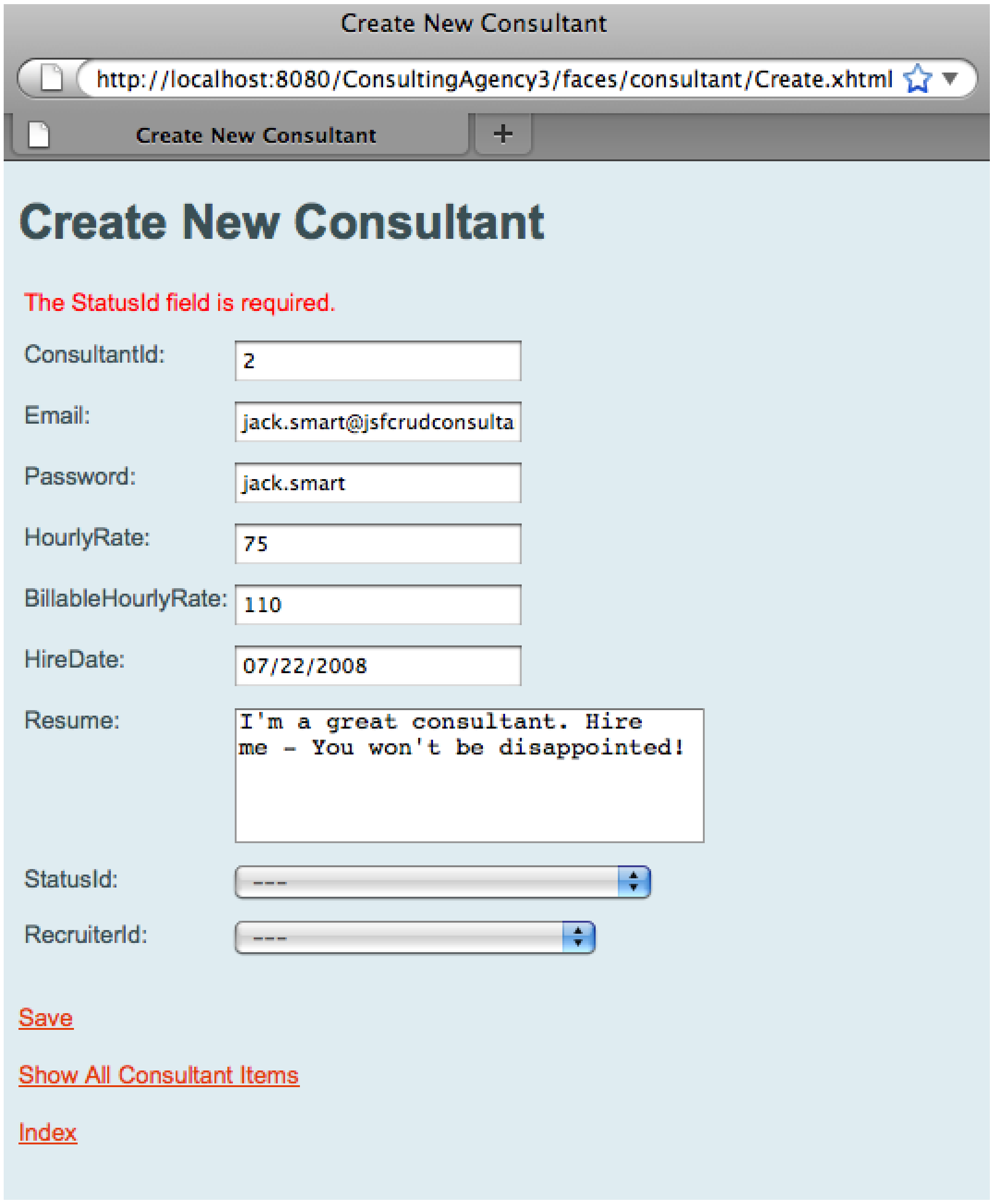
The method returns `Create`. The `NavigationHandler` gathers information behind the scenes, and applies the `Create` string to the path which targets the view sent in response to the request: `/consultant/Create.xhtml`. (In JSF 2.x, the file extension is inferred due to implicit navigation.)

Exploring Database Integrity with Field Validation

1. From the Consultants List page in the browser, click the 'Create New Consultant' link. As demonstrated in the previous sub-section, this triggers the `/consultant/Create.xhtml` page to render.
2. Enter the following details into the form. For the time being, leave both `RecruiterId` and `StatusId` fields blank.

Field	Value
ConsultantId	2
Email	jack.smart@jsfcrudconsultants.com
Password	jack.smart
HourlyRate	75
BillableHourlyRate	110
HireDate	07/22/2008
Resume	I'm a great consultant. Hire me - You won't be disappointed!
RecruiterId	---
StatusId	---

1. Click Save. When you do so, a validation error is flagged for the `StatusId` field.


Create New Consultant

http://localhost:8080/ConsultingAgency3/faces/consultant/Create.xhtml

Create New Consultant

Create New Consultant

The StatusId field is required.

ConsultantId:	2
Email:	jack.smart@jsfcrudconsulta
Password:	jack.smart
HourlyRate:	75
BillableHourlyRate:	110
HireDate:	07/22/2008
Resume:	I'm a great consultant. Hire me - You won't be disappointed!
StatusId:	---
RecruiterId:	---

Save

Show All Consultant Items

Index

Figure 21. Enter sample data into the form

Why did this happen? Reexamine the entity-relationship diagram for the Consulting Agency database. As stated in the relationships table above, the `CONSULTANT` and `CONSULTANT_STATUS` tables share a non-nullable, one-to-many relationship. Therefore, every entry in the `CONSULTANT` table must contain a reference to an entry in the `CONSULTANT_STATUS` table. This is denoted by the `consultant_fk_consultant_status` foreign key that links the two tables.

You can view foreign keys held by tables by expanding a table’s Foreign Keys node in the Services window (Ctrl-5; ⌘-5 on Mac).

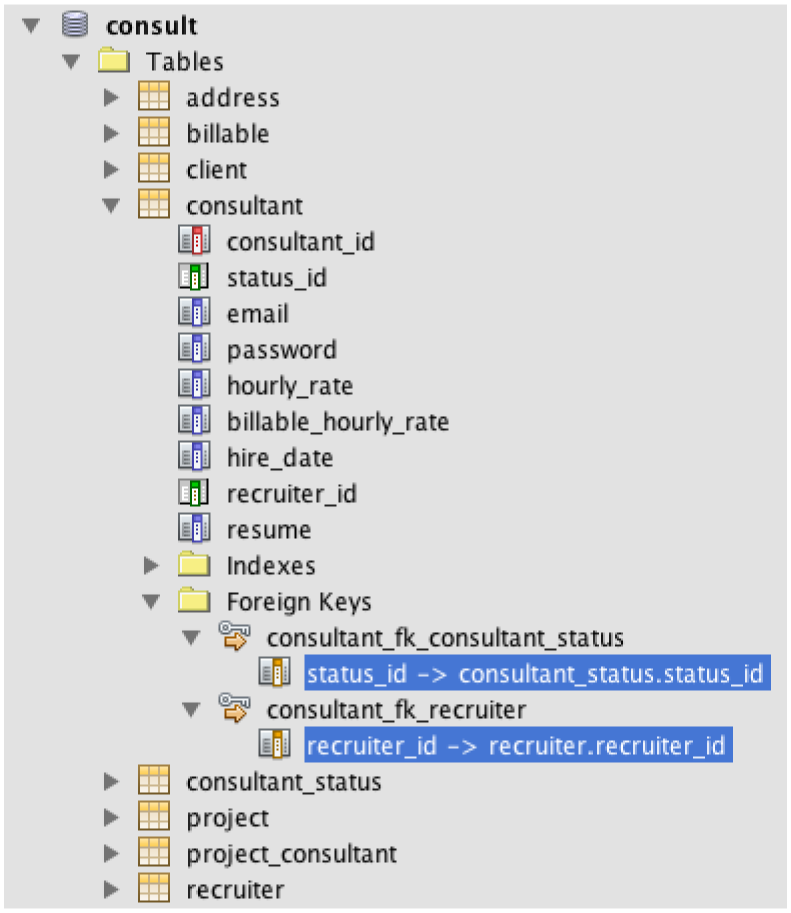


Figure 22. Examine foreign key attributes in the Services window

1. To overcome the validation error, select `entity.ConsultantStatus[statusId=A]` from the `StatusId` drop-down list.

*Note: *You can leave the `RecruiterId` field blank. As indicated in the database entity-relationship diagram, there is a nullable, one-to-many relationship between the `CONSULTANT` and `RECRUITER` tables, meaning that entries in `CONSULTANT` do not need to be associated with a `RECRUITER` entry.

1. Click Save. A message displays, indicating that the consultant entry was successfully saved. If you click `Show All Consultant Items`, you’ll see the new entry listed in the table.

In general, the generated Facelets pages provide errors for user input that introduces:

- empty fields for non-nullable table cells.
- modifications to data that cannot be altered (e.g., primary keys).
- insertion of data that is not of the correct type.
- modifications to data when a user's view is no longer synchronized with the database.

Editing Entity Classes

In the previous sub-section, you saw how the `StatusId` drop-down list provided you with the not-so-user-friendly `entity.ConsultantStatus[statusId=A]` option. You may already be aware that the text displayed for each item in this drop-down is a string representation for each `ConsultantStatus` entity encountered (i.e., The entity class' `toString()` method is called).

This sub-section demonstrates how you can use the editor's code completion, documentation, and navigation support to make this conclusion. It also has you prepare a more user-friendly message for the drop-down list.

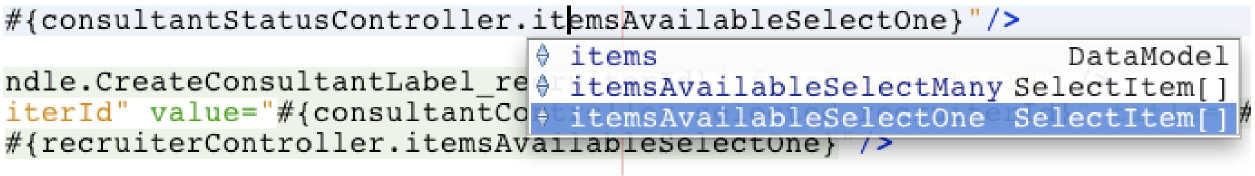
1. Open the `/consultant/Create.xhtml` file in the editor. This is the Create New Consultant form which you just viewed in the browser. Scroll down to the code for the `StatusId` drop-down (shown in **bold** below).

```
    <h:outputLabel value="#{bundle.CreateConsultantLabel_resume}" for="resume" />

    <h:inputTextarea rows="4" cols="30" id="resume"
value="#{consultantController.selected.resume}"
title="#{bundle.CreateConsultantTitle_resume}" />

    *<h:outputLabel value="#{bundle.CreateConsultantLabel_statusId}" for="statusId"
/>

    <h:selectOneMenu id="statusId" value="#{consultantController.selected.statusId}"
title="#{bundle.CreateConsultantTitle_statusId}" required="true"
requiredMessage="#{bundle.CreateConsultantRequiredMessage_statusId}">

        <f:selectItems
value="#{consultantStatusController.itemsAvailableSelectOne}"/>

    </h:selectOneMenu>*

    <h:outputLabel value="#{bundle.CreateConsultantLabel_recruiterId}"
for="recruiterId" />

    <h:selectOneMenu id="recruiterId"
value="#{consultantController.selected.recruiterId}"
title="#{bundle.CreateConsultantTitle_recruiterId}" >
```

```
            <f:selectItems value="#{recruiterController.itemsAvailableSelectOne}"/>

        </h:selectOneMenu>

    </h:panelGrid>
```

1. Examine the `value` applied to the `<f:selectItems>` tag. The `value` attribute determines the text that displays for each item in the drop-down list.

Press Ctrl-Space on `itemsAvailableSelectOne`. The editor's code completion indicates that the `ConsultantStatusController's` ``getItemsAvailableSelectOne()` method returns an array of `SelectItem` objects.

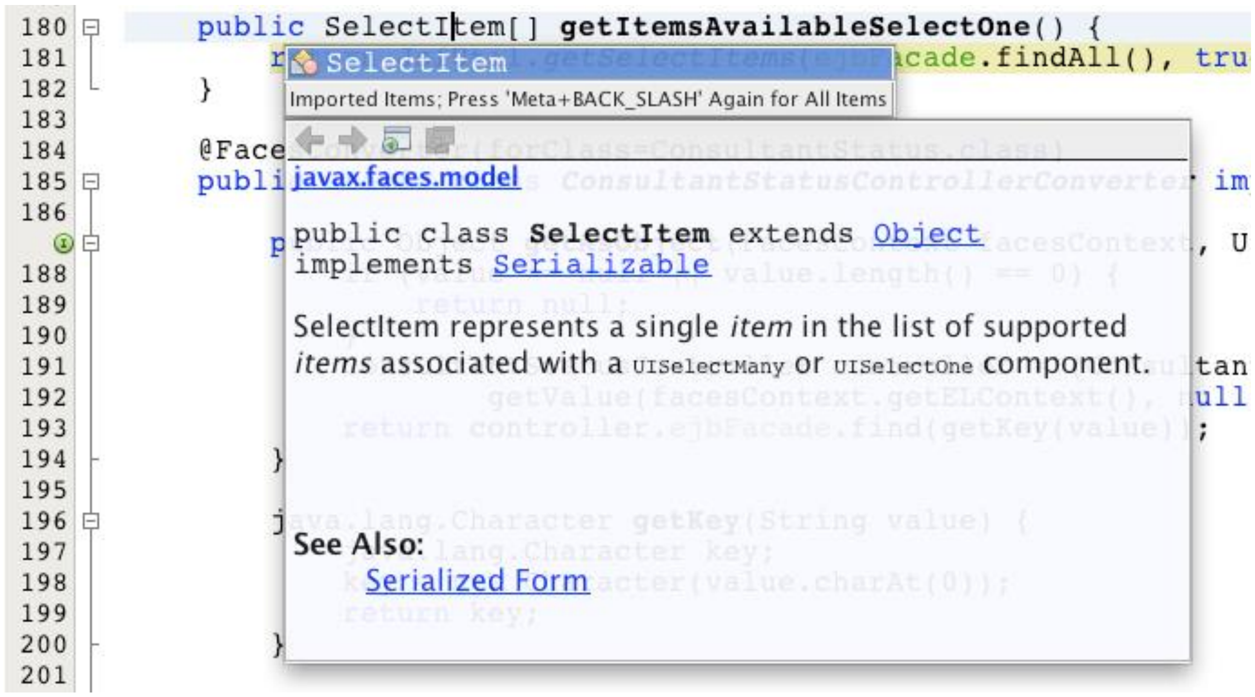


```
#{consultantStatusController.itemsAvailableSelectOne}"/>
ndle.CreateConsultantLabel_re
iterId" value="#{consultantCo
#{recruiterController.itemsAvailableSelectOne}"/>
```

items DataModel
itemsAvailableSelectMany SelectItem[]
itemsAvailableSelectOne SelectItem[]

Figure 23. Code completion displays returned classes for methods

1. Press Ctrl (⌘ on Mac), then hover your mouse over `itemsAvailableSelectOne`. A link is formed, enabling you to navigate directly to the `getItemsAvailableSelectOne()` method in the `ConsultantStatus` entity's source code. Click this link.
2. Place your cursor on the `SelectItem[]` return value in the method signature, and press Ctrl-Space to invoke the documentation pop-up.



```
    public SelectItem[] getItemsAvailableSelectOne() {
        r                                         acade.findAll(), tru
    }

    @Face
    publi
```

public class **SelectItem** extends Object
implements Serializable

SelectItem represents a single *item* in the list of supported *items* associated with a `UISelectMany` or `UISelectOne` component.

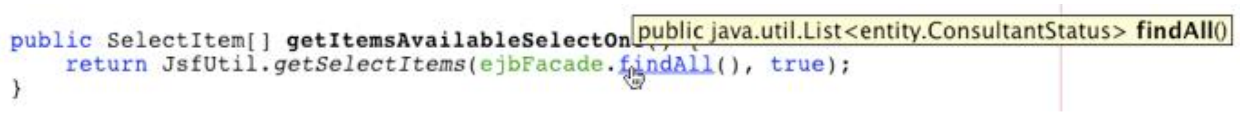
See Also:
Serialized Form

Figure 24. Press Ctrl-Space to invoke documentation support

Click the web browser ( ) icon in the documentation window to open the Javadoc in an external web browser.

As you can see, the `SelectItem` class belongs to the JSF framework.
The `UISelectOne` component, as mentioned in the documentation, is represented by the `<h:selectOneMenu>` tag from the markup which you examined in Step 1 above.

1. Press Ctrl (⌘ on Mac), then hover your mouse over `findAll()`. A pop-up appears, displaying the method signature.



```
public SelectItem[] getItemsAvailableSelectOn
    return JsfUtil.getSelectItems(ejbFacade.findAll(), true);
}
```

public java.util.List<entity.ConsultantStatus> **findAll()**

Figure 25. View pop-ups of method signatures in the editor

You can see that here `ejbFacade.findAll()` returns a `List` of `ConsultantStatus` objects.

1. Navigate to `JsfUtil.getSelectItems`. Hover your mouse over `getSelectItems` and press Ctrl (⌘ on Mac), then click the link that displays.

*Note: *Recall that `JsfUtil` is one of the utility classes that was generated when you completed the JSF Pages from Entity Classes wizard.

The method loops through the list of entities (i.e, the `List` of `ConsultantStatus` objects), creating a `SelectItem` for each. As indicated in **bold** below, each `SelectItem` is created using the entity object and a *label* for the object.

```
public static SelectItem[] getSelectItems(List<?> entities, boolean selectOne) {

    int size = selectOne ? entities.size() + 1 : entities.size();

    SelectItem[] items = new SelectItem[size];

    int i = 0;

    if (selectOne) {

        items[0] = new SelectItem("", "---");

        i++;

    }

    *for (Object x : entities) {
        items[i++] = new SelectItem(x, x.toString());

    }*

    return items;

}
```

The label is created using the entity's `toString()` method, and is the representation of the object when rendered in the response. (See the Javadoc definition for the `SelectItem(java.lang.Object value, java.lang.String label)` constructor.)

Now that you have verified that the entity `toString()` method is what is rendered in the browser when you view items in a drop-down list, modify the `ConsultantStatus` `toString()` method.

1. Open the `ConsultantStatus` entity class in the editor. Modify the `toString` method to return the `statusId` and `description`. These are entity properties which correspond to the two columns of the `CONSULTANT_STATUS` table.

```
public String toString() {

    return *statusId + ", " + description;*

}
```

1. Run the project again. When the browser displays the welcome page, click the `Show All Consultant Items` link, then click `Create New Consultant`.

Inspect the `statusId` drop-down. You'll see that it now displays the status ID and description for the one record contained in the database's `CONSULTANT_STATUS` table.

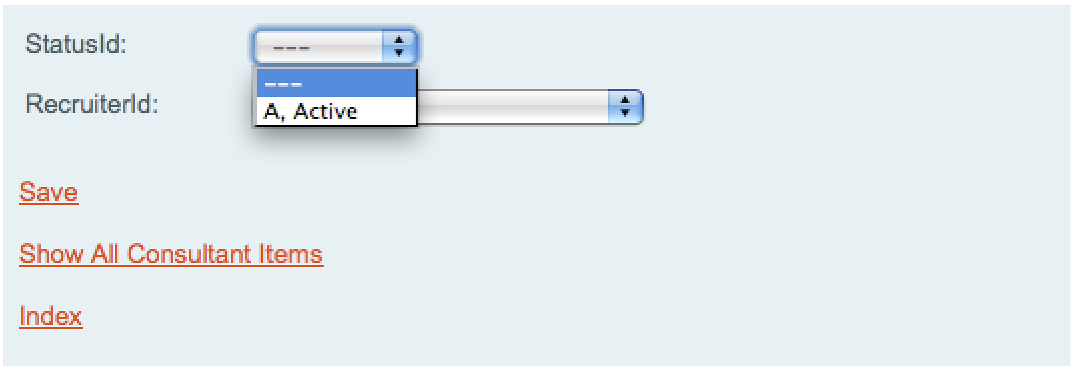


Figure 26. StatusId drop-down displays items according to ConsultantStatus entity's toString() method

Troubleshooting

Depending on your configuration, deploying the application to the server can fail and you might see the following message in the Output window.

```
GlassFish Server 4 is running.

In-place deployment at /MyDocuments/ConsultingAgency/build/web

GlassFish Server 4, deploy, null, false

/MyDocuments/ConsultingAgency/nbproject/build-impl.xml:1045: The module has not been
deployed.

See the server log for details.
```

The most common cause for the failure is a problem when generating the JDBC resources on the server. If this is the case, you will probably see a message similar to the following in the server log tab in the Output window.

```
Severe:   Exception while preparing the app : Invalid resource : jdbc/consult__pm

com.sun.appserv.connectors.internal.api.ConnectorRuntimeException: Invalid resource :
jdbc/consult__pm
```

If the server log tab is not open you can open the tab by right-clicking the GlassFish Server node in the Services window and choosing View Domain Server Log.

This application requires two JDBC resources:

- JDBC Resource or Datasource. The application uses JNDI lookup to locate the JDBC resource. If you look in the persistence unit (`persistence.xml`) you can see that the JNDI name for the JTA data source for this application is `jdbc/consult`.

The JDBC resource identifies the connection pool that is currently used by the application.

- JDBC Connection Pool. The connection pool specifies the connection details for the database, including the location, user name, password. The connection pool that is used for this application is `consultPool`.

The JDBC resource and connection pool are specified in the `glassfish-resources.xml` file. You can open `glassfish-resources.xml` in the editor by expanding the Server Resources node in the Projects window and double-clicking the file. The file will look similar to the following.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE resources PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1
Resource Definitions//EN" "http://glassfish.org/dtds/glassfish-resources_1_5.dtd">

<resources>

  <jdbc-connection-pool allow-non-component-callers="false" associate-with-
thread="false" connection-creation-retry-attempts="0" connection-creation-retry-
interval-in-seconds="10" connection-leak-reclaim="false" connection-leak-timeout-in-
seconds="0" connection-validation-method="auto-commit" datasource-
classname="com.mysql.jdbc.jdbc2.optional.MysqlDataSource" fail-all-
connections="false" idle-timeout-in-seconds="300" is-connection-validation-
required="false" is-isolation-level-guaranteed="true" lazy-connection-
association="false" lazy-connection-enlistment="false" match-connections="false" max-
connection-usage-count="0" max-pool-size="32" max-wait-time-in-millis="60000"
name="consultPool" non-transactional-connections="false" ping="false" pool-resize-
quantity="2" pooling="true" res-type="javax.sql.DataSource" statement-cache-size="0"
statement-leak-reclaim="false" statement-leak-timeout-in-seconds="0" statement-
timeout-in-seconds="-1" steady-pool-size="8" validate-atmost-once-period-in-
seconds="0" wrap-jdbc-objects="false">

    <property name="serverName" value="localhost"/>

    <property name="portNumber" value="3306"/>

    <property name="databaseName" value="consult"/>

    <property name="User" value="root"/>

    <property name="Password" value="nb"/>

    <property name="URL"
value="jdbc:mysql://localhost:3306/consult?zeroDateTimeBehavior=convertToNull"/>

    <property name="driverClass" value="com.mysql.jdbc.Driver"/>

  /<jdbc-connection-pool>

  <jdbc-resource enabled="true" jndi-name="jdbc/consult" object-type="user" pool-
name="consultPool"/>

/<resources>
```

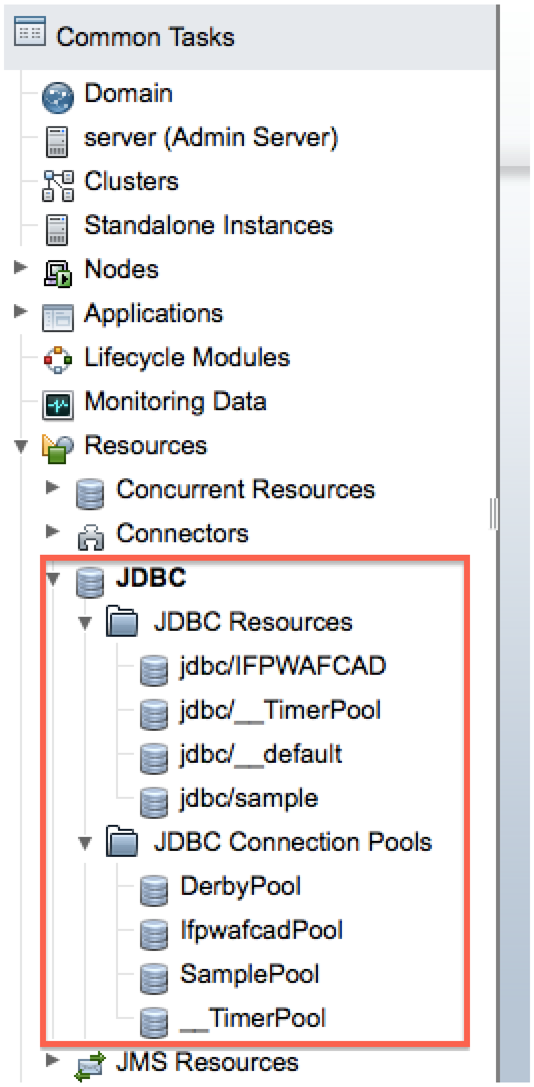
In `glassfish-resources.xml` you can see that the JDBC resource `jdbc/consult` identifies `consultPool` as the name of the connection pool. You can also see the properties for `consultPool`. In this application only one datasource and one connection pool are defined in `glassfish-resources.xml`. In some cases you might want to specify additional resources, for example, to identify a temporary data store that is used only for development or for testing.

If the JDBC resource and connection pool were not generated automatically on the server when you tried to the application, you can perform the following steps to manually create the resources in the GlassFish Admin Console.

1. Open `glassfish-resources.xml` in the editor if it is not already open.

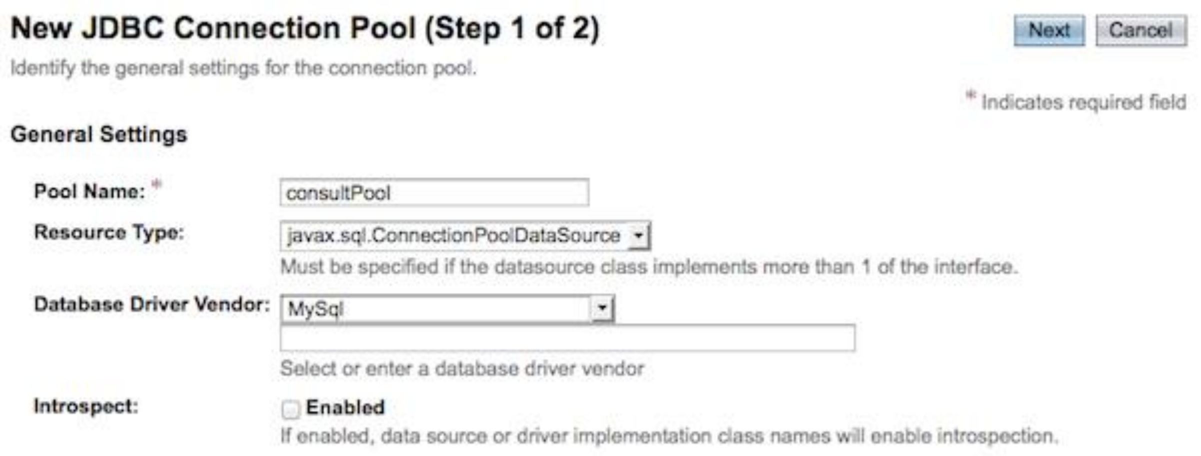
You will use the property values that are specified in `glassfish-resources.xml` when you create the JDBC resource and connection pool.

1. Right-click the GlassFish Server node in the Services window and choose Open Domain Admin Console in the popup menu to open the GlassFish Console in your browser.
2. In the Common Tasks navigation panel of the GlassFish Console, expand the **JDBC** node and the **JDBC Resources** and **JDBC Connection Pools** nodes.



You can see the JDBC resources that are currently registered with the server. You will need to create `jdbc/consult` and `consultPool` if they are not listed under the JDBC node in the Common Tasks navigation panel. Some JDBC resources were created by default when you installed the server and are displayed as sub-nodes.

1. Click the **JDBC Connection Pools** node and click New in the New JDBC Connection Pool pane.



New JDBC Connection Pool (Step 1 of 2)

Identify the general settings for the connection pool.

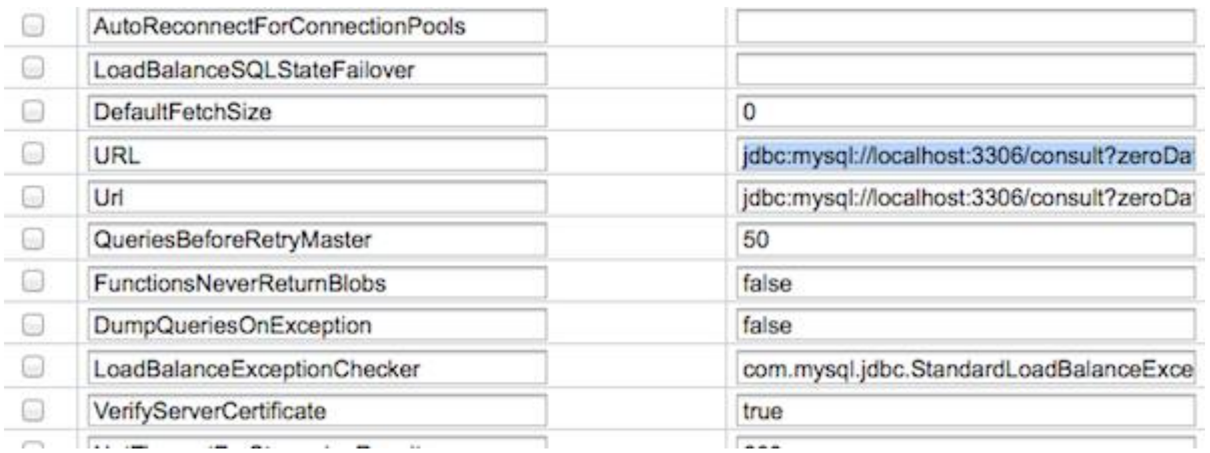
* Indicates required field

General Settings

Pool Name: *	consultPool
Resource Type:	javax.sql.ConnectionPoolDataSource Must be specified if the datasource class implements more than 1 of the interface.
Database Driver Vendor:	MySql Select or enter a database driver vendor
Introspect:	Enabled If enabled, data source or driver implementation class names will enable introspection.

Figure 27. New JDBC Connection Pool pane in the GlassFish Admin Console

1. Type **consultPool** as the Pool Name, select **javax.sql.ConnectionPoolDataSource** as the Resource Type and select **MySql** as the Database Driver Vendor. Click Next.
2. In Step 2, locate and specify the values for the **URL**, **username** and **password** properties. Click Finish.



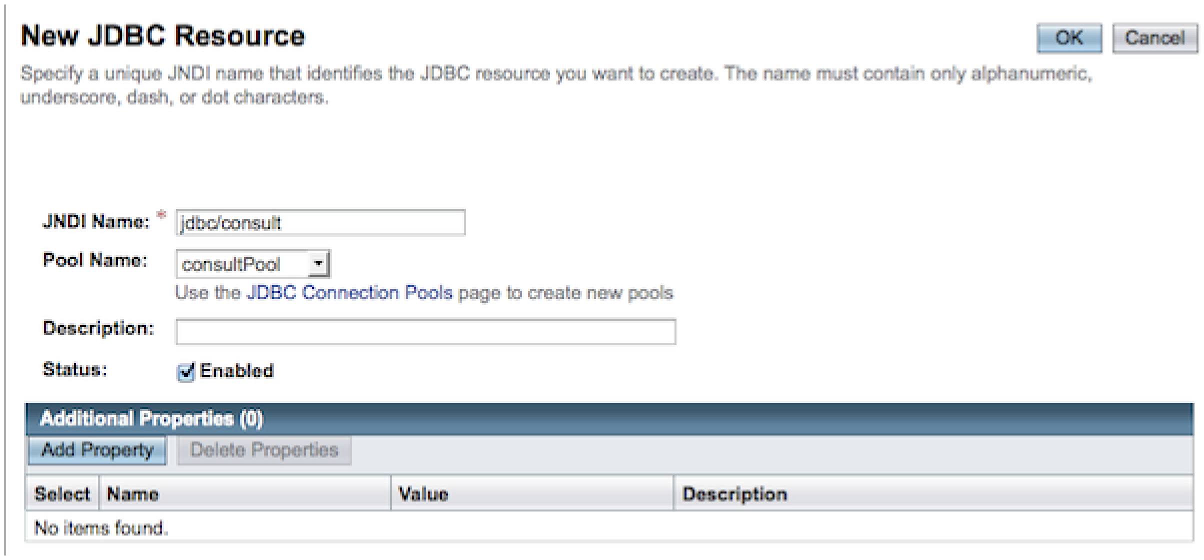
	AutoReconnectForConnectionPools	
	LoadBalanceSQLStateFailover	
	DefaultFetchSize	0
	URL	jdbc:mysql://localhost:3306/consult?zeroDa
	Url	jdbc:mysql://localhost:3306/consult?zeroDa
	QueriesBeforeRetryMaster	50
	FunctionsNeverReturnBlobs	false
	DumpQueriesOnException	false
	LoadBalanceExceptionChecker	com.mysql.jdbc.StandardLoadBalanceExce
	VerifyServerCertificate	true

Figure 28. New JDBC Connection Pool panel in the GlassFish Admin Console

You can find the values for the properties in `glassfish-resources.xml`.

The new connection pool is created on the server when you click Finish and a node for the connection pool is displayed under the JDBC Connection Pools node.

1. Click the **JDBC Resources** node in the Common Tasks navigation panel and click New.
2. Type **jdbc/consult** for the JNDI Name and select **consultPool** in the Pool Name drop-down list. Click OK.



New JDBC Resource

Specify a unique JNDI name that identifies the JDBC resource you want to create. The name must contain only alphanumeric, underscore, dash, or dot characters.

JNDI Name: *	jdbc/consult
Pool Name:	consultPool Use the JDBC Connection Pools page to create new pools
Description:	
Status:	✓ Enabled

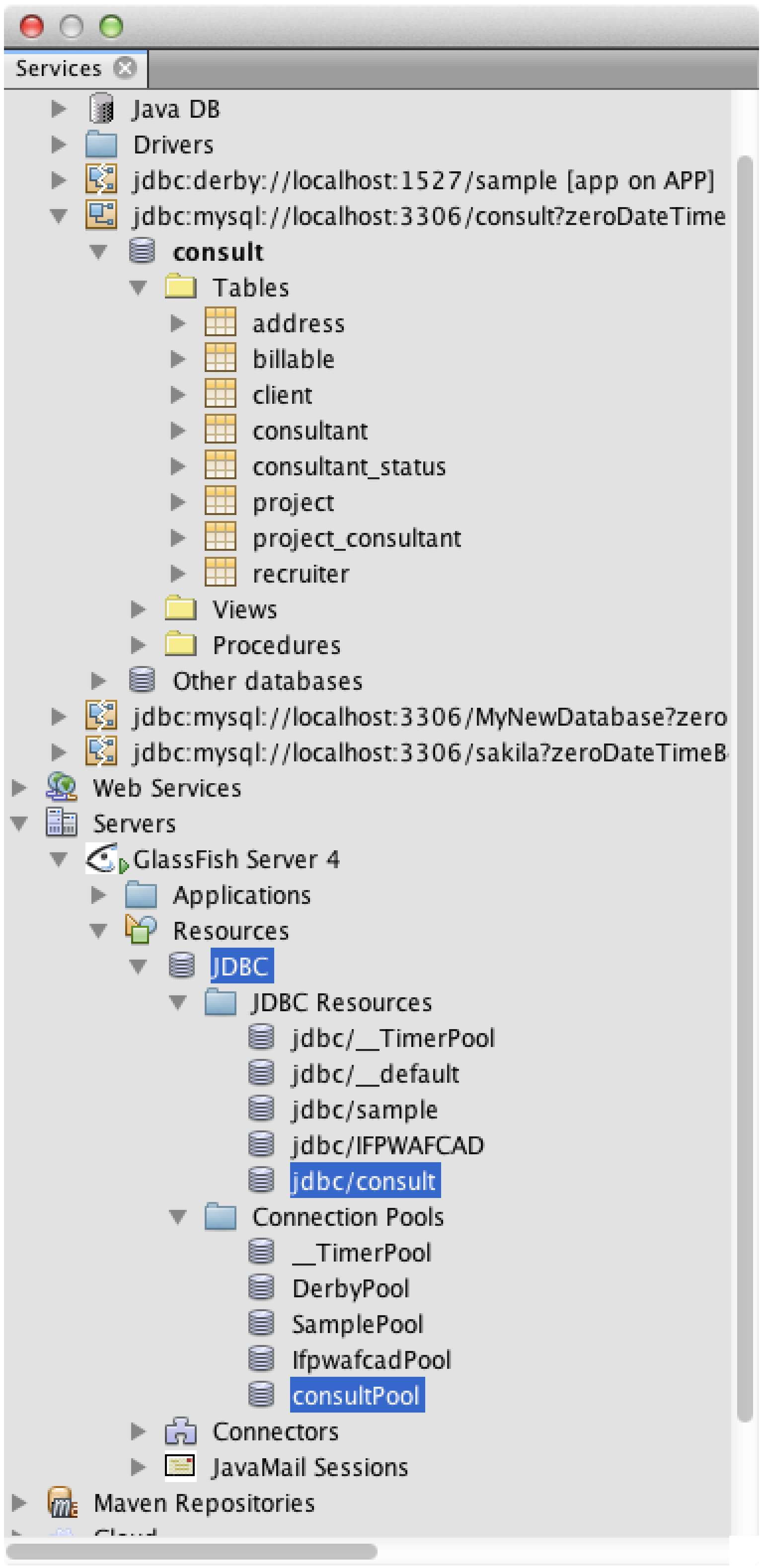
Additional Properties (0)

Select	Name	Value	Description
No items found.			

Figure 29. New JDBC Resource pane in the GlassFish Admin Console

The new JDBC resource is created on the server when you click OK and a node for the resource is displayed under the JDBC Resources node.

In the Service window of the IDE, you can expand the Resources node under the GlassFish Server and see that IDE added the new resources. You might need to refresh the view (right-click Resources and choose Refresh) to view the changes.



```
Services
▶ Java DB
▶ Drivers
▶ jdbc:derby://localhost:1527/sample [app on APP]
▼ jdbc:mysql://localhost:3306/consult?zeroDateTime
    ▼ consult
        ▼ Tables
            ▶ address
            ▶ billable
            ▶ client
            ▶ consultant
            ▶ consultant_status
            ▶ project
            ▶ project_consultant
            ▶ recruiter
        ▶ Views
        ▶ Procedures
    ▶ Other databases
▶ jdbc:mysql://localhost:3306/MyNewDatabase?zero
▶ jdbc:mysql://localhost:3306/sakila?zeroDateTimeB
▶ Web Services
▼ Servers
    ▼ GlassFish Server 4
        ▶ Applications
        ▼ Resources
            ▼ JDBC
                ▼ JDBC Resources
                      jdbc/__TimerPool
                      jdbc/__default
                      jdbc/sample
                      jdbc/IFPWAFCAD
                      jdbc/consult
                ▼ Connection Pools
                      __TimerPool
                      DerbyPool
                      SamplePool
                      IfpwafcadPool
                      consultPool
            ▶ Connectors
            ▶ JavaMail Sessions
▶ Maven Repositories
```

Figure 30. JDBC Resources displayed in the Services window of the IDE

For more tips on troubleshooting problems when using MySQL and the IDE, see the following documents:

- Connecting to a MySQL Database tutorial.
- The troubleshooting section of Creating a Simple Web Application Using a MySQL Database Send Feedback on This Tutorial

Courtesy: https://netbeans.apache.org/kb/docs/web/jsf20-crud.html

Modified: 2021.10.04.7.22.AM

Dököll Solutions, Inc.