

# Java EE Intro

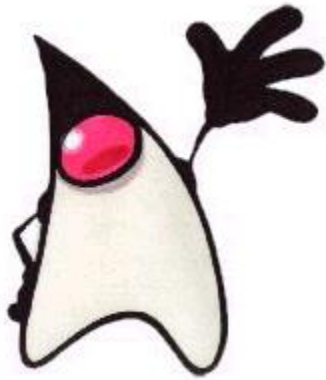
CSE 305 – Principles of Database Systems

Paul Fodor

Stony Brook University

<http://www.cs.stonybrook.edu/~cse305>

# JavaEE Setup is much of the battle



(c) <APACHE ANT>

ny Brook)



# Lots and lots of stuff

- Tools
  - i.e. NetBeans, Glassfish, etc.
- Core Libraries
  - JSF, JPA, Ajax, etc.
- Related APIs
  - PrimeFaces, Struts, Spring, etc.

# Common Java Frameworks

1. Spring MVC (23%)
2. Struts 1.x (15%)
3. Apache Axis (15%)
4. Apache Xerces (14%)
5. Hibernate (12%)
6. JDOM (12%)
7. Java Applet (8.1%)
8. Apache Velocity (7.9%)
9. Apache ORO (7.0%)
10. JAX-WS (6.5%)

Source: VeraCode Blog

# A good place to start

- Core Libraries
  - Java Server Faces
  - JSTL & EL
  - Java Persistence API
  - Enterprise Java Beans
  - JavaScript & Ajax

# A good place to start

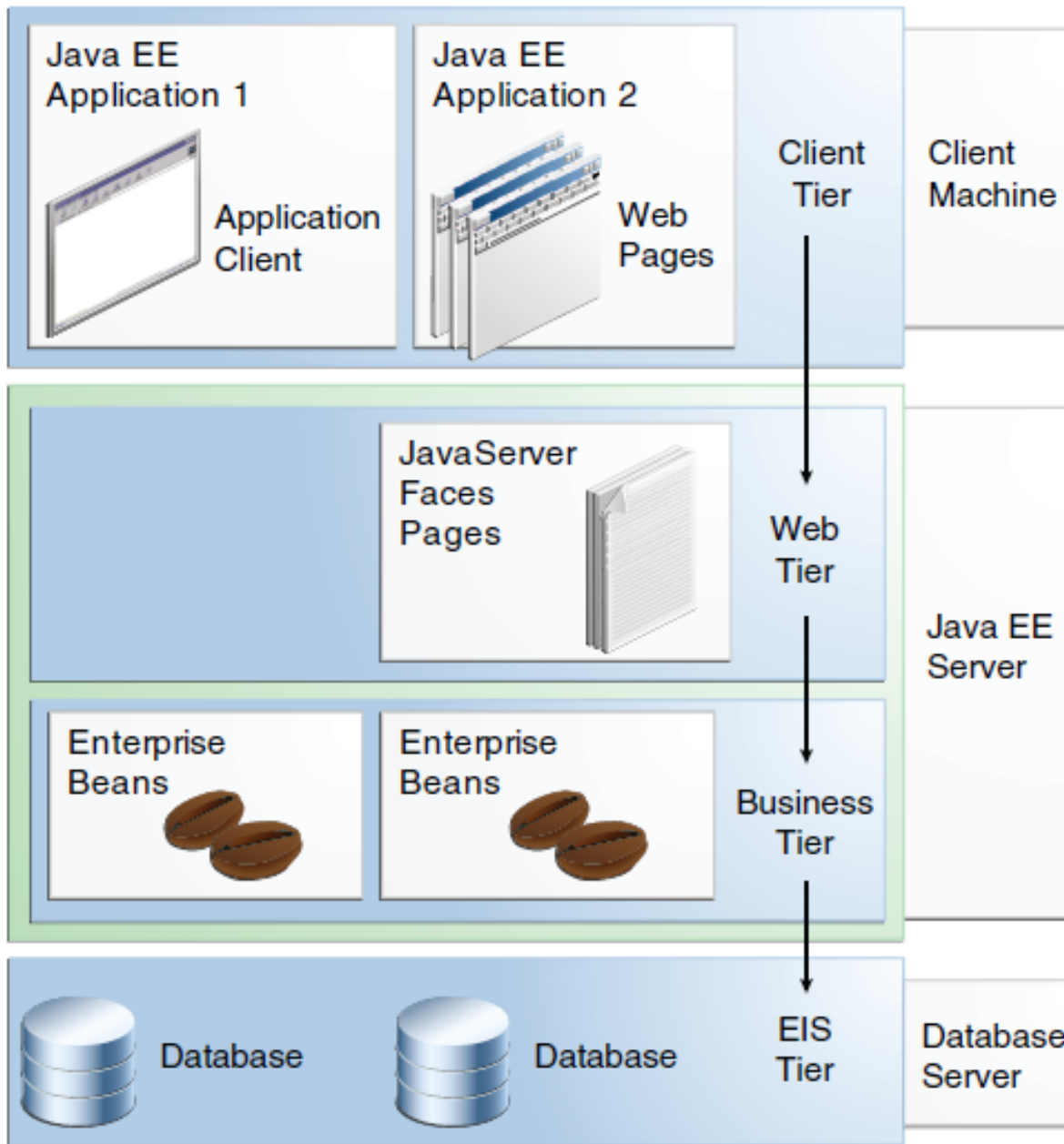
- Related AP
  - RichFaces
  - PrimeFaces
  - Spring
  - Struts
  - JQuery

# Servers, Servers, and More Servers

- What are:
  - Web Servers?
  - Application Servers?
  - Enterprise Servers?
  - Database Servers?
  - Java EE Servers?
  
- Abstraction, abstraction, abstraction



# Multi-Tier JavaEE Applications





# First: Annotations

- Provide data about a program to:

- compiler

```
@Override
```

- Tools

```
void mySuperMethod() { }
```

- JVM

```
@SuppressWarnings("unchecked")
```

```
void myMethod() { }
```

- Can be applied to:

- classes

- fields

- Methods

- Can contain elements with values

# Annotations Look-Up

- Scattered in the Java EE 6 API. Ex:
  - <http://docs.oracle.com/javaee/6/api/javax/annotation/package-summary.html>
  - <http://docs.oracle.com/javaee/6/api/javax/faces/bean/package-summary.html>
- Via cheat sheet:
  - <http://www.physics.usyd.edu.au/~rennie/javaEE6ReferenceSheet.pdf>

# Java EE 6 Annotations

## Alternatives for management

CDI	JSF	EJB
-----	-----	-----

### CDI: javax.inject

CMF @Inject  
 @Named(value="")  
 @Singleton

### CDI: javax.enterprise.context

TMF @ApplicationScoped  
 TMF @SessionScoped  
 TMF @ConversationScoped  
 TMF @RequestScoped  
 TMF @Dependent

### CDI: javax.enterprise.inject

TMF @New(value=Class.class)  
 TMF @Alternative  
 TMF @Any  
 MF @Produces, @Disposes

### JSF management: javax.faces.bean

T @ManagedBean(name="", eager=false)  
 T @CustomScoped(value="")  
 T @NoneScoped  
 T @ApplicationScoped  
 T @SessionScoped  
 T @ViewScoped  
 T @RequestScoped  
 F @ManagedProperty(name="", value="")  
 T @ReferencedBean(name="")

### EJB injection: javax.ejb

TMF @EJB(name="", beanInterface=Interface.class, mappedName="", lookup="""| beanName="", description="")  
 T @EJBs(@EJB[])

### Resource injection: javax.annotation

TMF @Resource(name="", type=Class.class, authenticationType=[AuthenticationType.CONTAINER, APPLICATION], shareable=true, lookup="", mappedName="")  
 T @Resources(@Resource[])

## EJB Types: javax.ejb

### Session beans

T @Stateless(name="ClassName")  
 T @Stateful(name="ClassName")  
 T @Singleton(name="ClassName")  
 T @Local(Class.class[])  
 T @Remote(Class.class[])  
 T @LocalBean  
 TM @Asynchronous  
 TM @Lock({LockType.WRITE, READ})  
 T @ConcurrencyManagement({CONTAINER, BEAN})  
 T @DependsOn(String[])  
 T @Startup

### Non-session beans

T @MessageDriven(name="ClassName", activationConfig=@ActivationConfigProperty())  
 @ActivationConfigProperty(propertyName="", propertyValue="")  
 T @ManagedBean(value="") [in javax.annotation.\*]

## Timeouts: javax.ejb

TM @AccessTimeout(value="0", unit=MILLISECONDS)  
 T @StatefulTimeout(value="0", unit=MILLISECONDS)  
 M @Timeout  
 M @Schedule(year="\*", month="\*", bimonthly="\*", dayOfWeek="\*", hour="0", minute="0", info="", persistent=true, timezone="")  
 M @Schedules(@Schedule[])

## Transaction: javax.ejb

T @TransactionManagement({CONTAINER, BEAN})  
 TM @TransactionAttribute({TransactionAttributeType.MANDATORY, REQUIRED, REQUIRES\_NEW, SUPPORTS, NOT\_SUPPORTED, NEVER})  
 M @AfterBegin  
 M @BeforeCompletion  
 M @AfterCompletion

## Lifecycle: javax.ejb

M @PostConstruct [in javax.annotation.\*]  
 M @PreDestroy [in javax.annotation.\*]  
 M @PostActivate  
 M @PrePassivate  
 M @Remove(retainIfException=false)

## Interceptors: javax.interceptor

TM @Interceptors(Class.class[])  
 TM @ExcludeDefaultInterceptors  
 M @ExcludeClassInterceptors  
 M @AroundInvoke  
 M @AroundTimeout  
 T @Interceptor [only required with @InterceptorBinding]

## Security: javax.annotation.security

T @RunAs(String rolename)  
 T @DeclareRoles(String[])  
 TM @RolesAllowed(String[])  
 TM @PermitAll  
 TM @DenyAll

```
-- Possible source file layout for web app -
1 lib/ [potentially copied to /lib/ inside an EAR]
  -- extra.jar [jar shared between all modules]
2 src/java/ [potentially packaged as EJB-JAR inside EAR -
  -- ValidationMessages.properties or under
  -- JSFStrings.properties WEB-INF/classes/
  -- META-INF/ inside WAR]
  | -- persistence.xml [for JPA config]
  | -- ejb-jar.xml [for deployment descriptors]
  -- com/
  | -- myBusiness/
  | | -- entities/
  | | | -- Entities.java
  | | -- EJBs.java
  web/ [potentially packaged as a WAR inside an EAR]
  | -- WEB-INF/
  | | -- beans.xml [for CDI config]
  | | -- faces-config.xml [for JSF config]
  | | -- web.xml [for Servlet 2.5 config]
  -- resources/
  | -- css/
  | | -- standard.css
  | -- javascript/
  | | -- standard.js
  EAR class loader levels
  -- jsfpages.xhtml
```

### Legend

TCMF Annotation for Type, Constructor, Method, Field  
 VALUE Default Value

## Java EE 6 Annotations Cheat Sheet

Version 1.2 ©2005,2011 Philipp Meier

Version 1.4 (2012-02-27) by Chris Rennie, based on

Java EE 6 API Doc: EJB 3.1, JSF 2.0, JPA 2.0  
[www.physics.usyd.edu.au/~rennie/](http://www.physics.usyd.edu.au/~rennie/)

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 2.0 Germany License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/2.0/de/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

# CDI?

- Context & Dependency Injection
  - f.k.a Web Beans
- Contexts?
  - lets you use JavaBeans, EJBs, etc. in other contexts
- Dependency Injection?
  - context polymorphism
- CDI will do much of the work behind our annotations

# CDI At Work

- **@Named**

- makes a bean accessible via a Facelet page. Ex:

```
@Named("cart")
@SessionScoped
public class ShoppingCart
```

- **bookcatalog.xhtml:**

```
<h:commandLink id="check" action="bookshowcart"
  immediate="true"
  rendered="#{cart.numberOfItems > 0}">
  <h:outputText value="#{bundle.CartCheck}"/>
</h:commandLink>
```

# So what are facelets?

- A page declaration language
  - used to build JSF views
- And tag libraries:
  - <ui : for templating
  - <h : for HTML components
  - <f : for custom actions
  - <c : for JSTL (Java language features)
  - <fn : more JSTL (Java API features)
- **JSTL: JavaServerPages Standard Tag Library**  
(c) Pearson Education Inc. and Paul Fodor (CS Stony Brook)

# Facelets use EL

- Expression Language
- For what?
  - evaluate expressions
  - get/set data
  - invoke methods
- EL defines:
  - how to write expressions: `${customer.age + 20}`
  - how to reference get/set: `#{customer.name}`
  - how to invoke methods: `val="#{customer.validateName}"`

# What's a managed bean?

- In JSF apps, typically each page connects to one. Why?
  - defines methods/properties associated with the page's components
  - Why?



# We've Seen Front-End JavaEE

- JavaServerFaces



- XHTML



- CSS

style.css

# Now for the Back-End

- What's the Java Persistence API (JPA)?
- What's an Enterprise Java Bean (EJB)?
  - server-side component
  - encapsulates business logic
  - Ex:
    - check inventory
    - order products

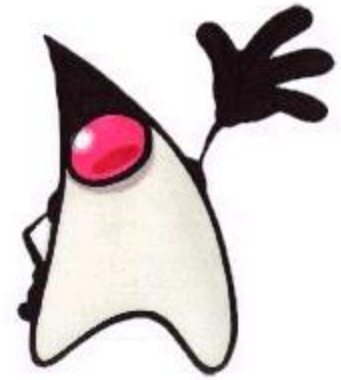
# Java Persistence API (JPA)

- Provides object/relational mapping to relational DB
- What does that mean?
  - makes it easy to talk to Dbs
- Why?
  - separate roles and employ abstraction



# The JPA Entity

- A JSP Domain
- It represents a DB Table
  - an Entity Instance would be a row
- Mapping done via annotations



```
@Entity  
public class Book  
{
```

# Why use EJBs?

- Simplify development of large, distributed apps
- Scalability
- EJB Containers provide system-level services to EJBs:
  - transaction management
  - concurrency
  - security features
- Again, separation of roles
  - thinner clients

# Two types of EJBs

- Session EJBs
  - performs a task for a client
- Message-Driven EJBs
  - acts as message listener (like for JMS)

# Session Beans

- Clients (i.e. facelets) invoke session bean methods
- Does its work on the server
- Not persistent
  - its data not saved to database
- Three Types:
  - stateful, stateless, & singleton

# Stateful Session Beans

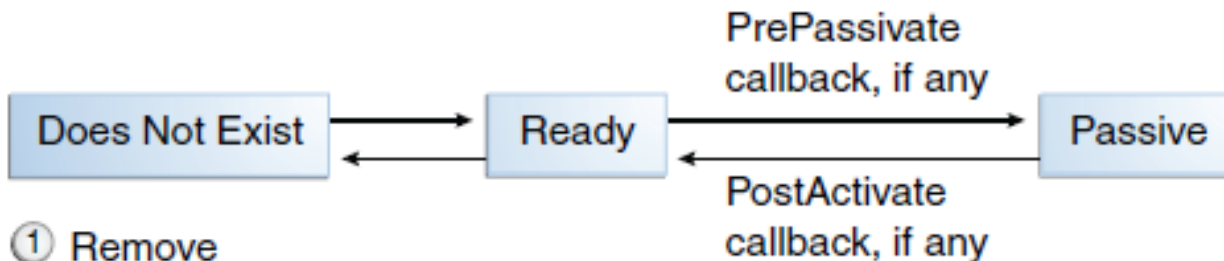
- Not shared
  - belongs to a single client
  - can store info about clients
- Lasts for duration of client/server session
  - when client terminates, bean terminates

① Create

② Dependency injection, if any

③ PostConstruct callback, if any

④ Init method, or ejbCreate<METHOD>, if any



① Remove

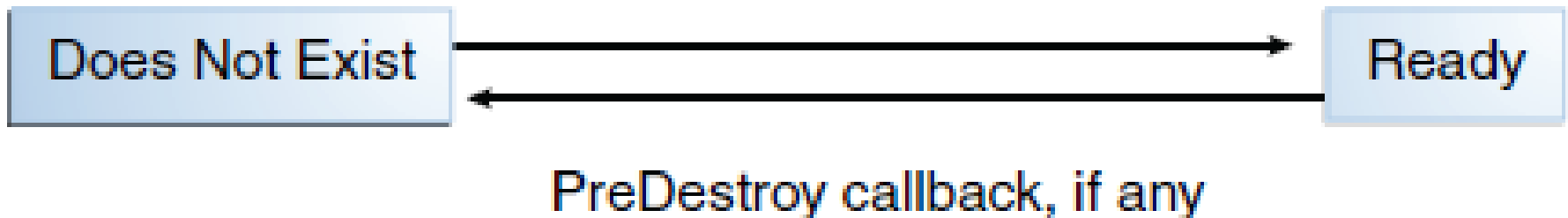
② PreDestroy callback, if any



# Stateless Session Beans

- Support multiple clients
- Methods do not “remember” clients
- Scalability advantages over stateful beans

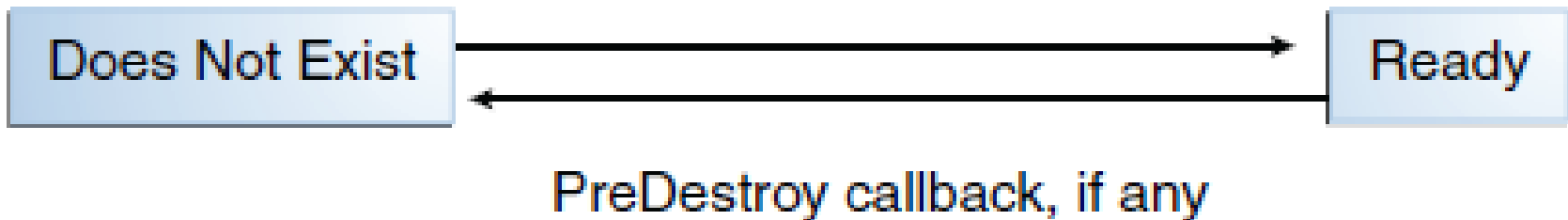
- ① Dependency injection, if any
- ② PostConstruct callback, if any



# Singleton Session Beans

- Lives for duration of application
- Only one of them
- Shared among clients

- ① Dependency injection, if any
- ② PostConstruct callback, if any



# Why use stateful beans?

- The bean's state represents the interaction between the bean and a specific client
- The bean needs to hold info about the client across method invocations
- The bean mediates between the client and the other components of the application, presenting a simplified view to the client
- Behind the scenes the bean manages the workflow of several EJBs

# Why use stateless beans?

- The bean has no data for a specific client
- In a single method invocation, the bean performs a generic task for all clients
- The bean implements a web service

# Why use singleton beans?

- State needs to be shared across the application
- A single enterprise bean needs to be accessed by multiple threads concurrently
- The application needs an enterprise bean to perform tasks upon application startup and shutdown
- The bean implements a web service

# How does a client use EJBs?

- Dependency injection
  - i.e. @EJB

OR

- JNDI Lookup
  - for non Java EE apps

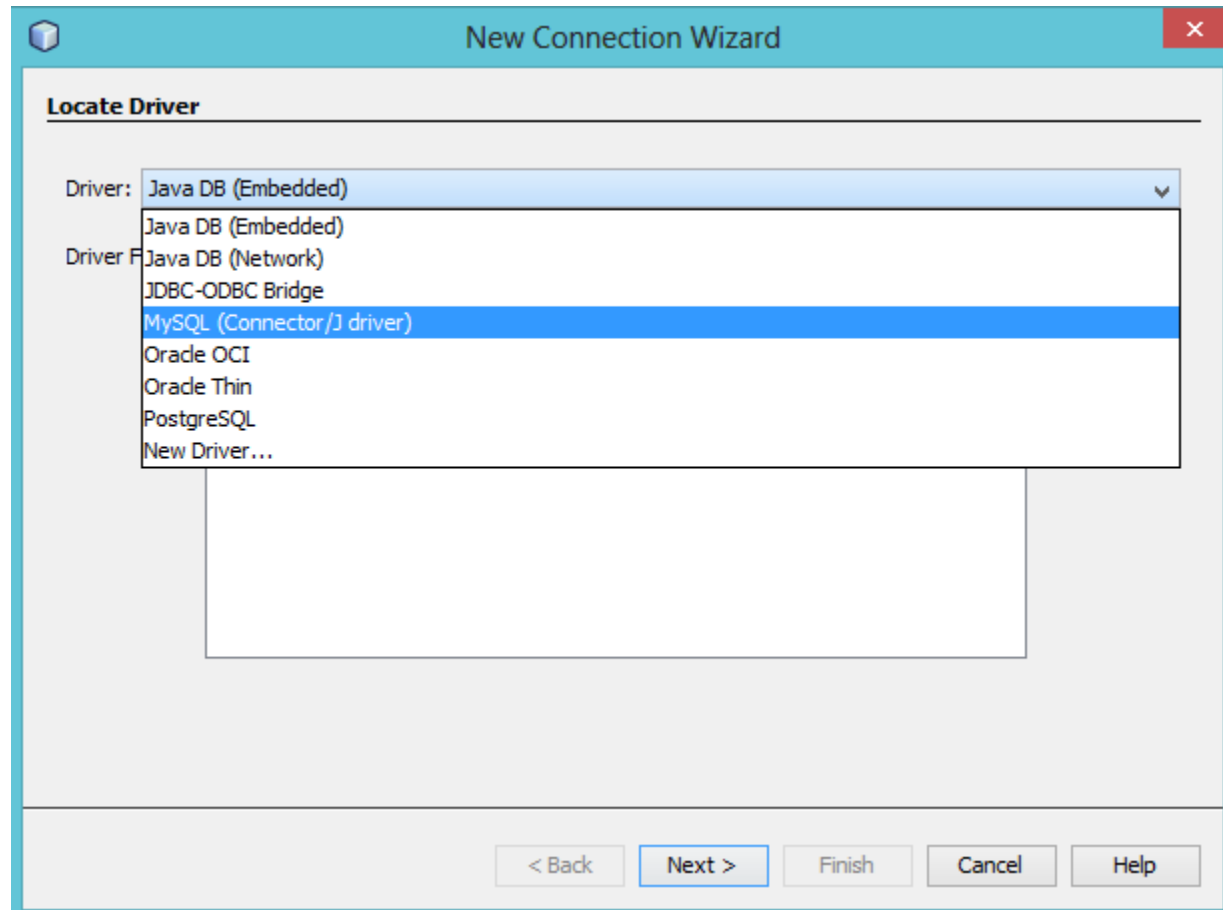
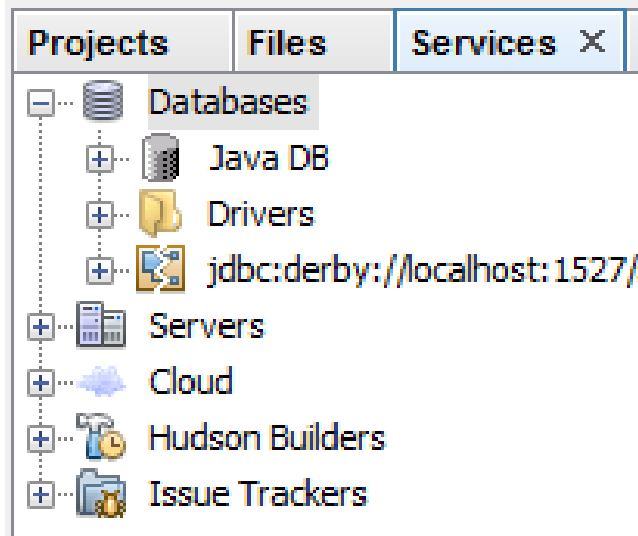
# The IDE

## NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	C/C++	PHP	All
NetBeans Platform SDK	•	•			•
Java SE	•	•			•
Java FX	•	•			•
Java EE		•			•
Java ME					•
HTML5		•		•	•
Java Card™ 3 Connected					•
C/C++			•		•
Groovy					•
PHP				•	•
Bundled servers					
GlassFish Server Open Source Edition 4.0		•			•
Apache Tomcat 7.0.34		•			•
	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>
	Free, 81 MB	Free, 204 MB	Free, 52 MB	Free, 52 MB	Free, 224 MB

- We want **EVERYTHING!**

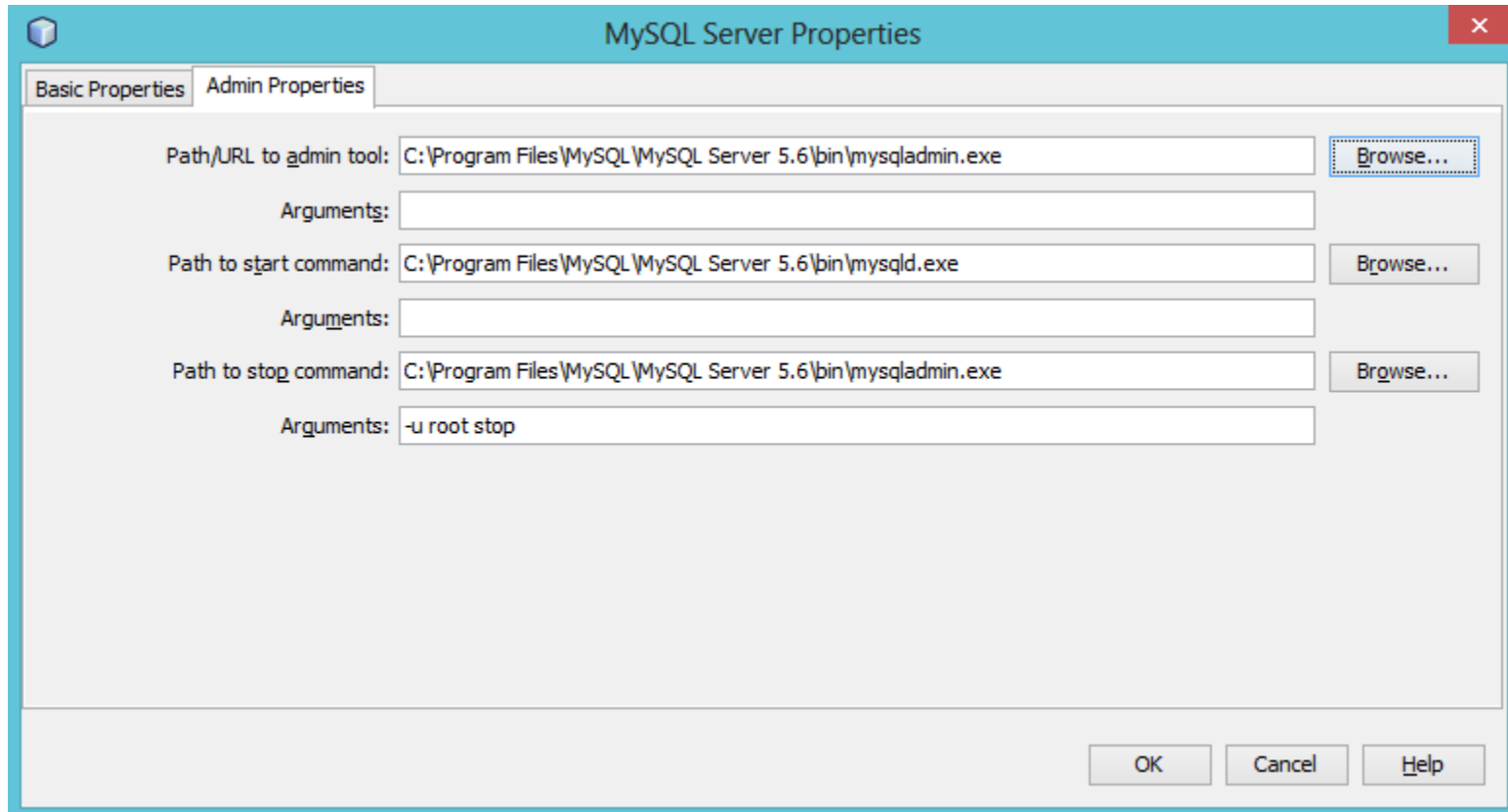
# Hook-Up NetBeans to MySQL



- Services Tab
- New Connection



# Hook-Up NetBeans to MySQL



- Now we can start building databases
- Connect to it

# CRUD?

- Create Read Update Delete
- Basic functions of persistent storage
- *Create* or add new entries
- *Read*, retrieve, search, or view existing entries
- *Update* or edit existing entries
- *Delete*/deactivate existing entries
- If you have these, you can make any complex Web app

# Generating a JavaServer Faces 2.x CRUD Application from a Database

- <https://netbeans.org/kb/docs/web/jsf20-crud.html>

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.

# First create the database

- Download mysql-consult.zip from <https://netbeans.org/projects/samples/downloads/directory/Samples/Java%20Web/ConsultingAgencySolution>
  - mysql\_create\_consult.sql
  - mysql\_insert\_data\_consult.sql
- Start the database server
- Create Database (called consult)
- Connect to database
- File → Open File → mysql\_create\_consult.sql

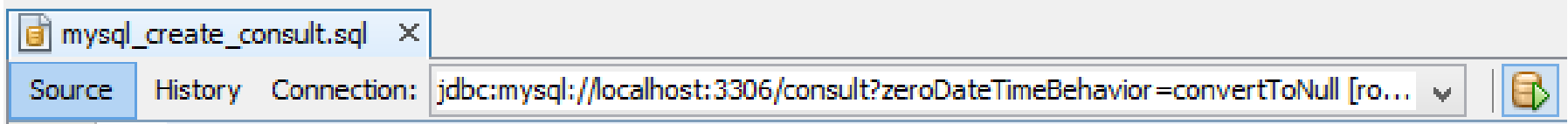
mysql\_create\_consult.sql

Source

History Connection:

```
1 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
2 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
3 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';
4
5 DROP SCHEMA IF EXISTS consult;
6 CREATE SCHEMA consult;
7 USE consult;
8
9 CREATE TABLE address (
10     address_id INTEGER NOT NULL AUTO_INCREMENT,
11     line1 VARCHAR(50) NOT NULL,
12     line2 VARCHAR(50) NULL,
13     city VARCHAR(50) NOT NULL,
14     region VARCHAR(50) NOT NULL,
15     country VARCHAR(50) NOT NULL,
16     postal_code VARCHAR(50) NOT NULL,
17     CONSTRAINT address_pk PRIMARY KEY ( address_id )
18 )ENGINE=InnoDB DEFAULT CHARSET=utf8;
19
20 CREATE TABLE consultant_status (
21     status_id CHAR NOT NULL,
22     description VARCHAR(50) NOT NULL,
23     CONSTRAINT consultant_status_pk PRIMARY KEY ( status_id )
24 )ENGINE=InnoDB DEFAULT CHARSET=utf8;
25
26 CREATE TABLE consultant (
27     consultant_id INTEGER NOT NULL AUTO_INCREMENT,
28     status_id CHAR NOT NULL,
29     email VARCHAR(50) NOT NULL,
30     password VARCHAR(50) NOT NULL,
31     hourly_rate DECIMAL(6,2) NOT NULL,
32     billable hourly_rate DECIMAL(6,2) NOT NULL,
```

# Run the SQL script



- This will build the tables

...

Executed successfully in 0.316 s, 0 rows affected.

Line 100, column 1

...

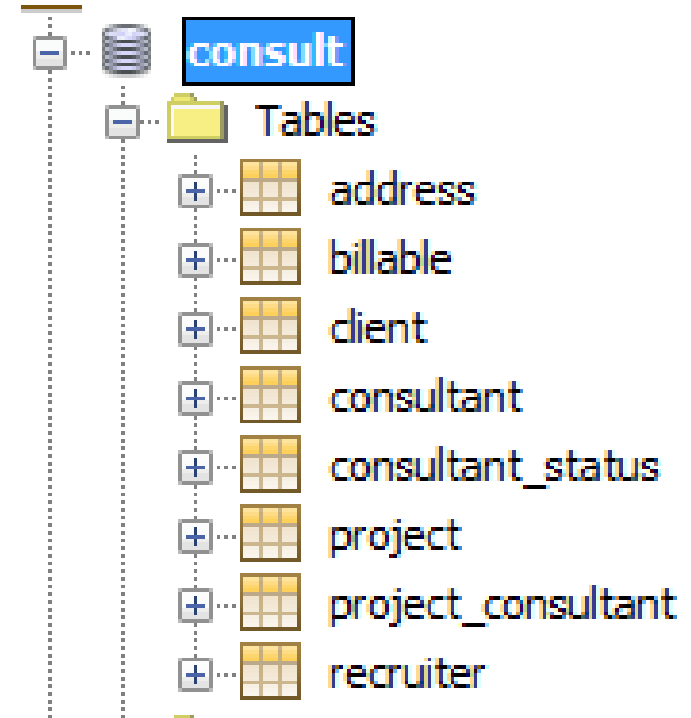
Executed successfully in 1.193 s, 0 rows affected.

Line 101, column 1

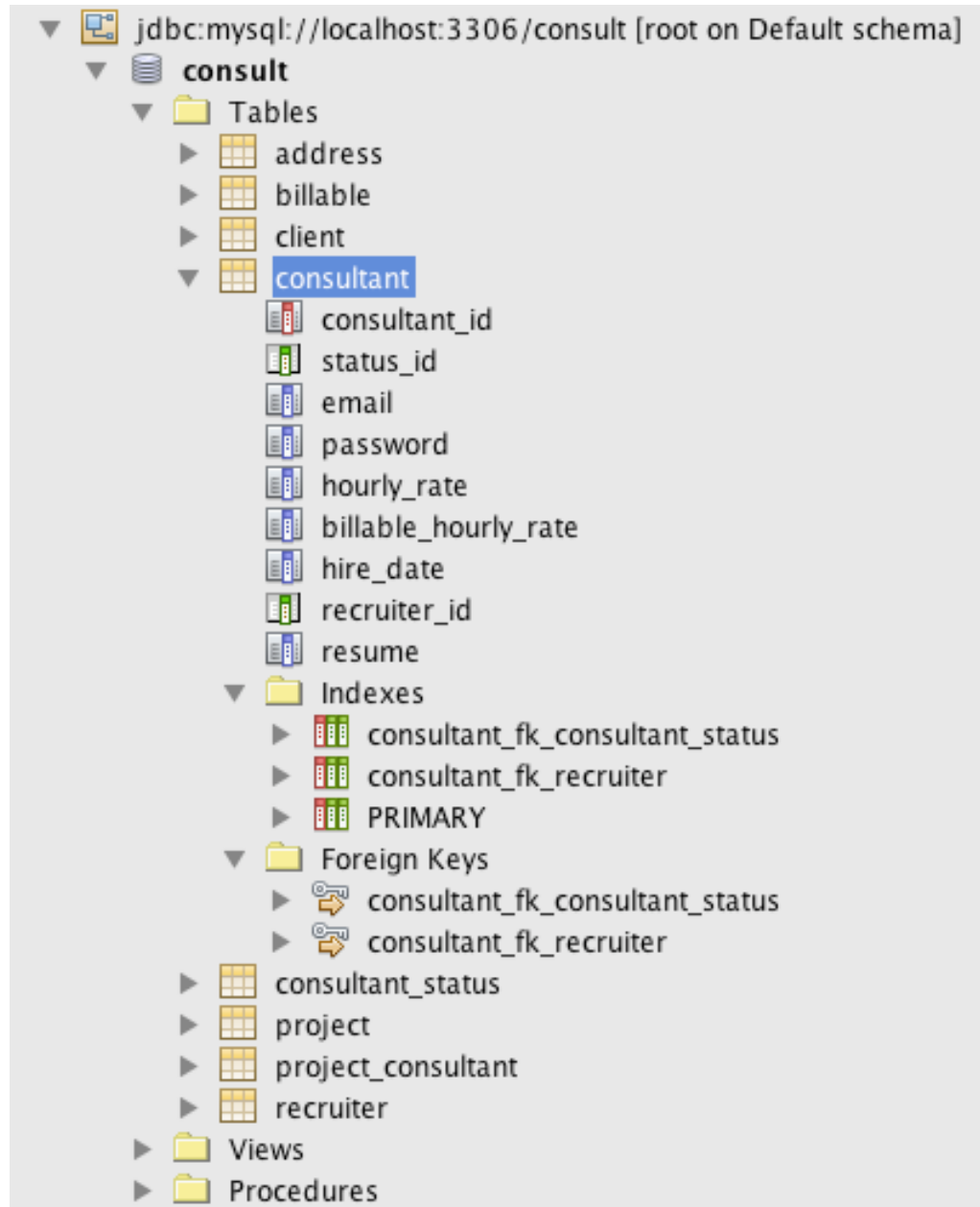
Executed successfully in 0 s, 0 rows affected.

Line 108, column 1

Execution finished after 17.665 s, 0 error(s) occurred.

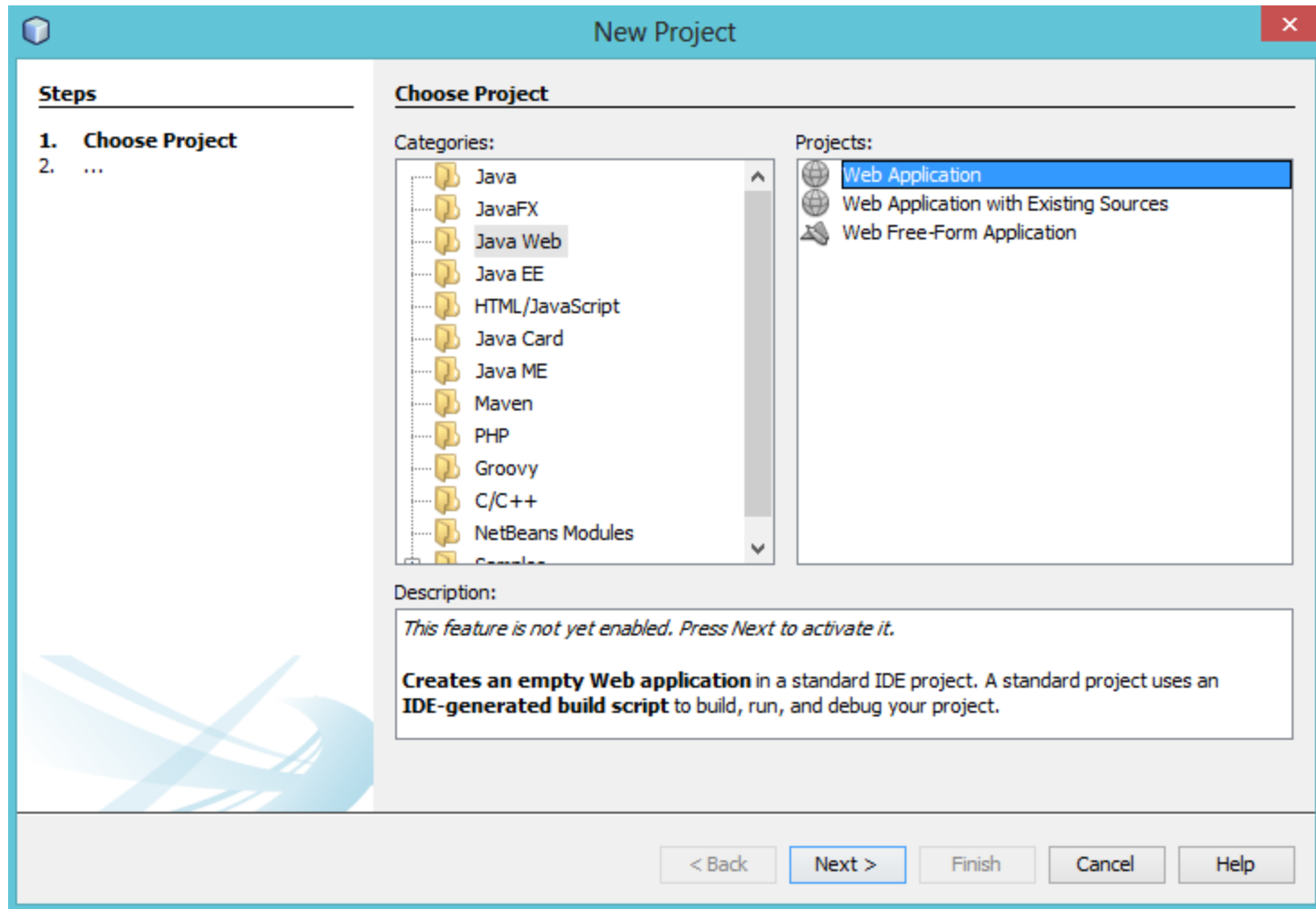


# Examine the Database



# Create the Web App

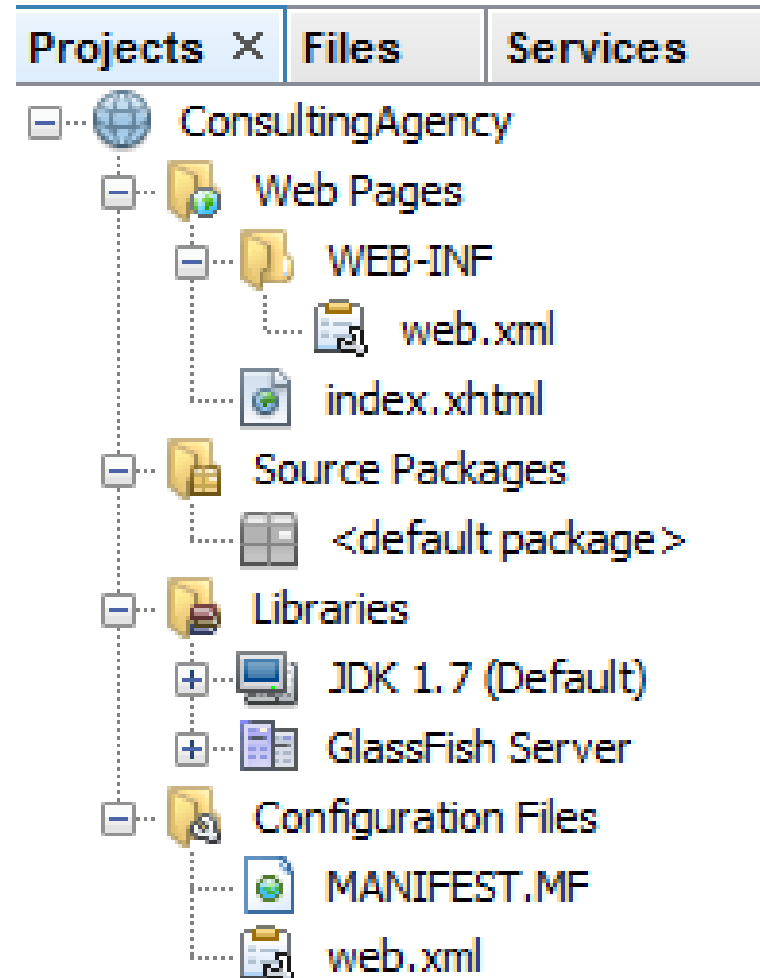
- File → New Project → Consulting Agency



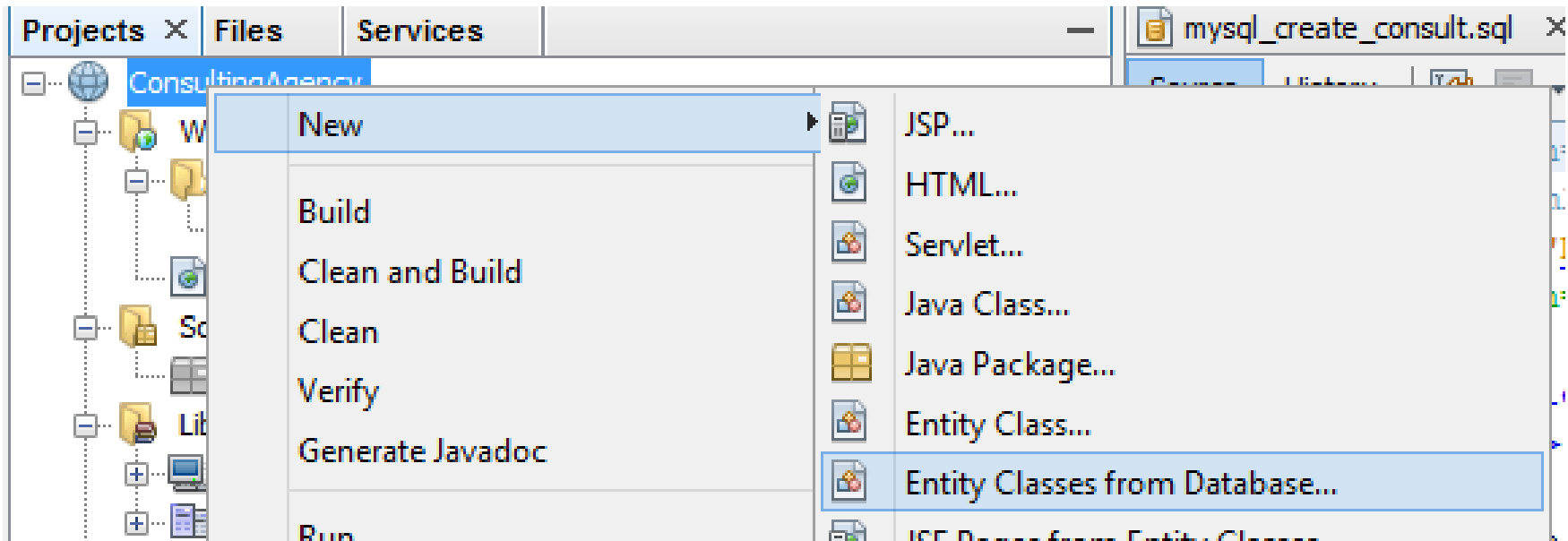


# Project Setup

- Specify your server
  - GlassFish 4.0
- Select your Framework
  - we'll use JavaServerFaces
  - note the other choices
- Now what?
  - We want our Web app to be able to talk to the DB



# Generate Entity Classes from the Database



- We'll need to create a New Data Source

# New Data Source: jdbc/consult

The screenshot shows the 'New Entity Classes from Database' wizard. The 'Steps' pane on the left lists: 1. Choose File Type, 2. Database Tables (highlighted), 3. Entity Classes, and 4. Mapping Options. The 'Database Tables' pane shows a 'Data Source' dropdown set to 'New Data Source...', and empty 'Available Tables' and 'Selected Tables' lists. A 'Create Data Source' dialog box is overlaid, containing: 'JNDI Name: jdbc/consult', 'Database Connection: jdbc:mysql://localhost:3306/consult?zeroDateTimeBehavior...', 'OK', 'Cancel', and 'Help' buttons. Below the dialog, there is a dropdown set to 'Any' and a checked checkbox for 'Include Related Tables'. A red warning icon and the text 'Select the table source.' are visible at the bottom of the wizard. At the very bottom, navigation buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help' are present.

**Steps**

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

**Database Tables**

Data Source: New Data Source...

Available Tables:

Selected Tables:

**Create Data Source**

JNDI Name: jdbc/consult

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

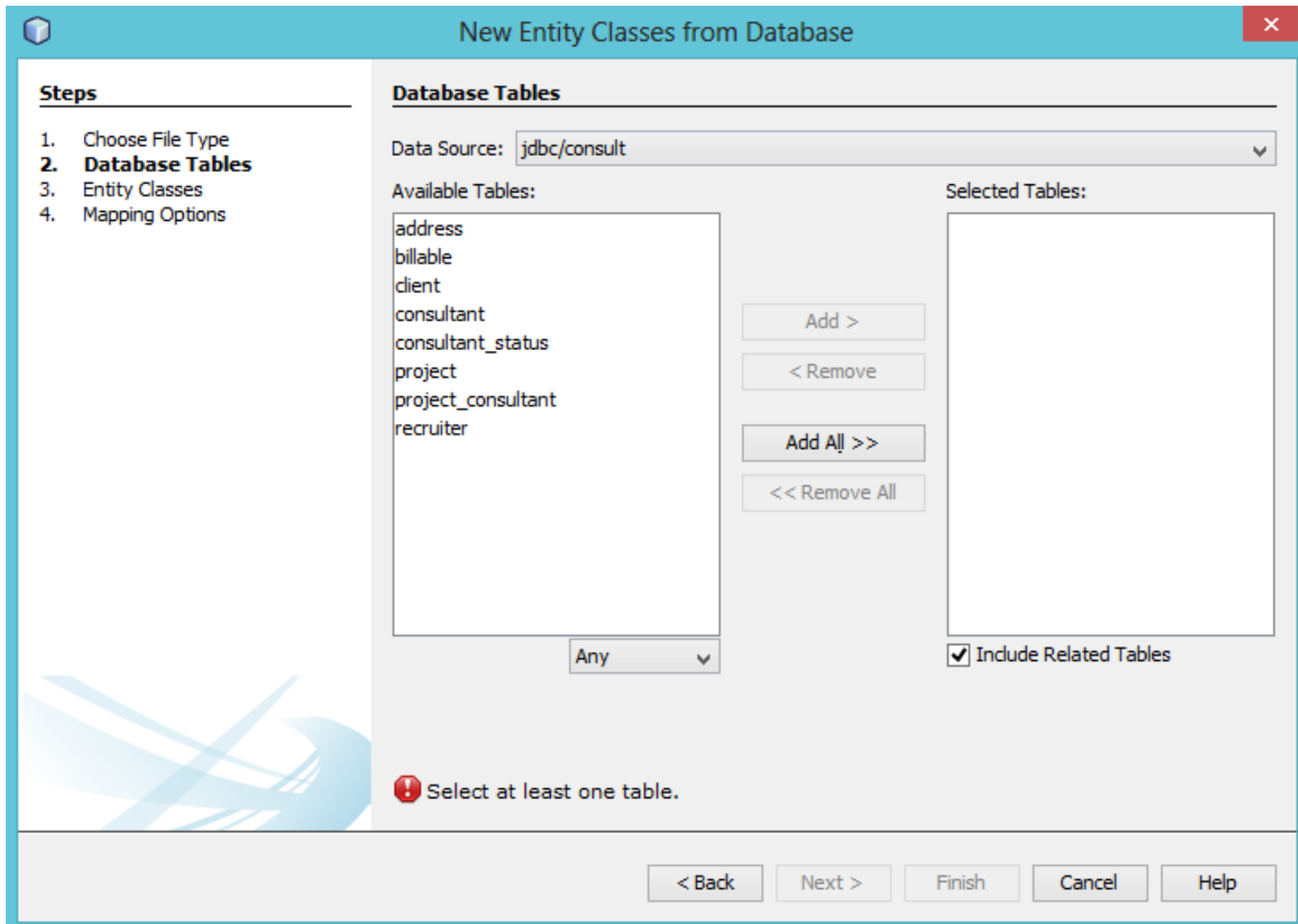
OK Cancel Help

Any  Include Related Tables

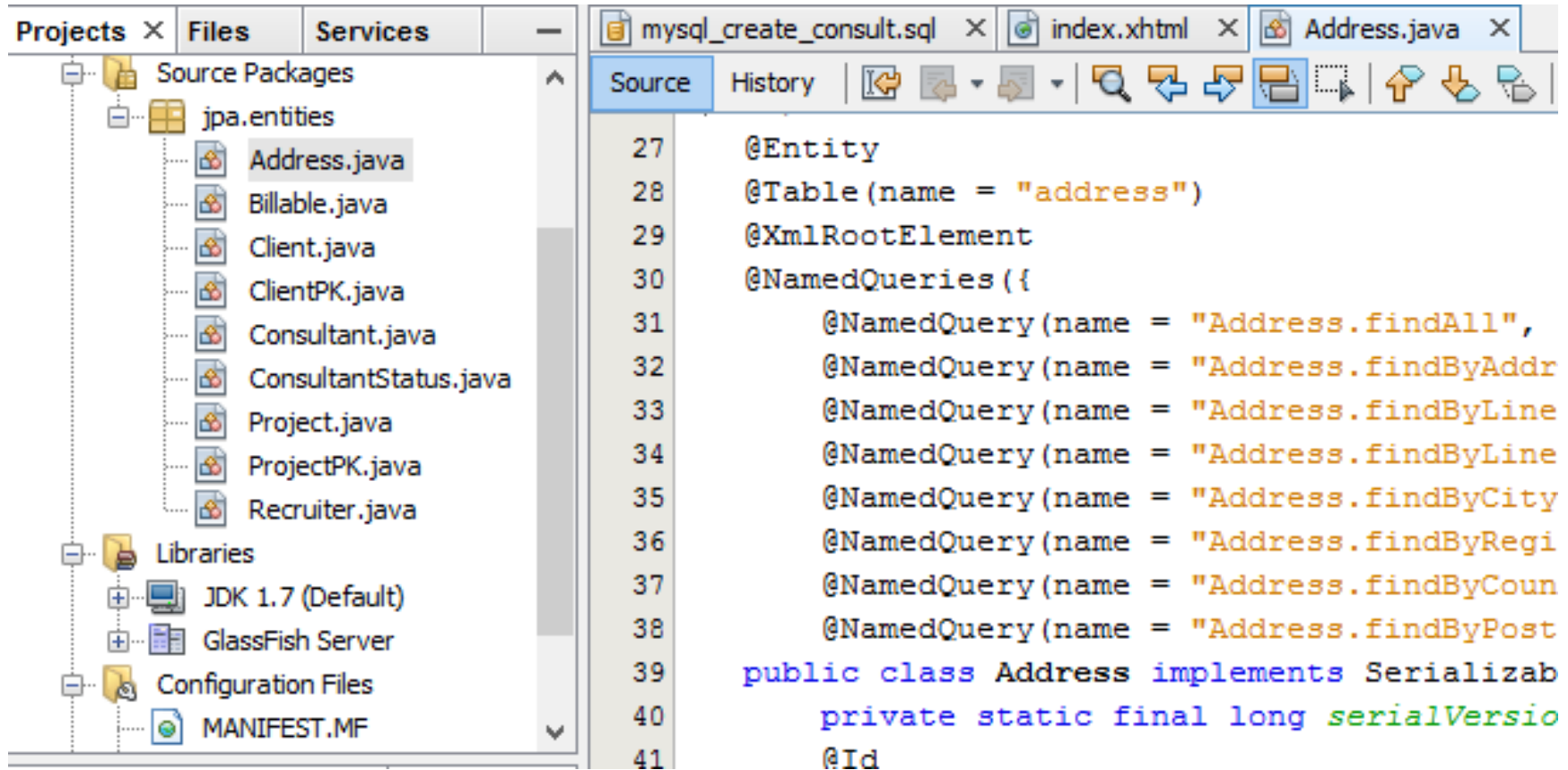
Select the table source.

< Back Next > Finish Cancel Help

# Add All >> to jpa.entities package



# Examine the generated Entity Classes



The screenshot shows an IDE interface with a project structure on the left and the source code of the Address.java entity class on the right. The project structure includes Source Packages (jpa.entities) and Libraries (JDK 1.7 (Default), GlassFish Server, Configuration Files, MANIFEST.MF). The source code of Address.java is as follows:

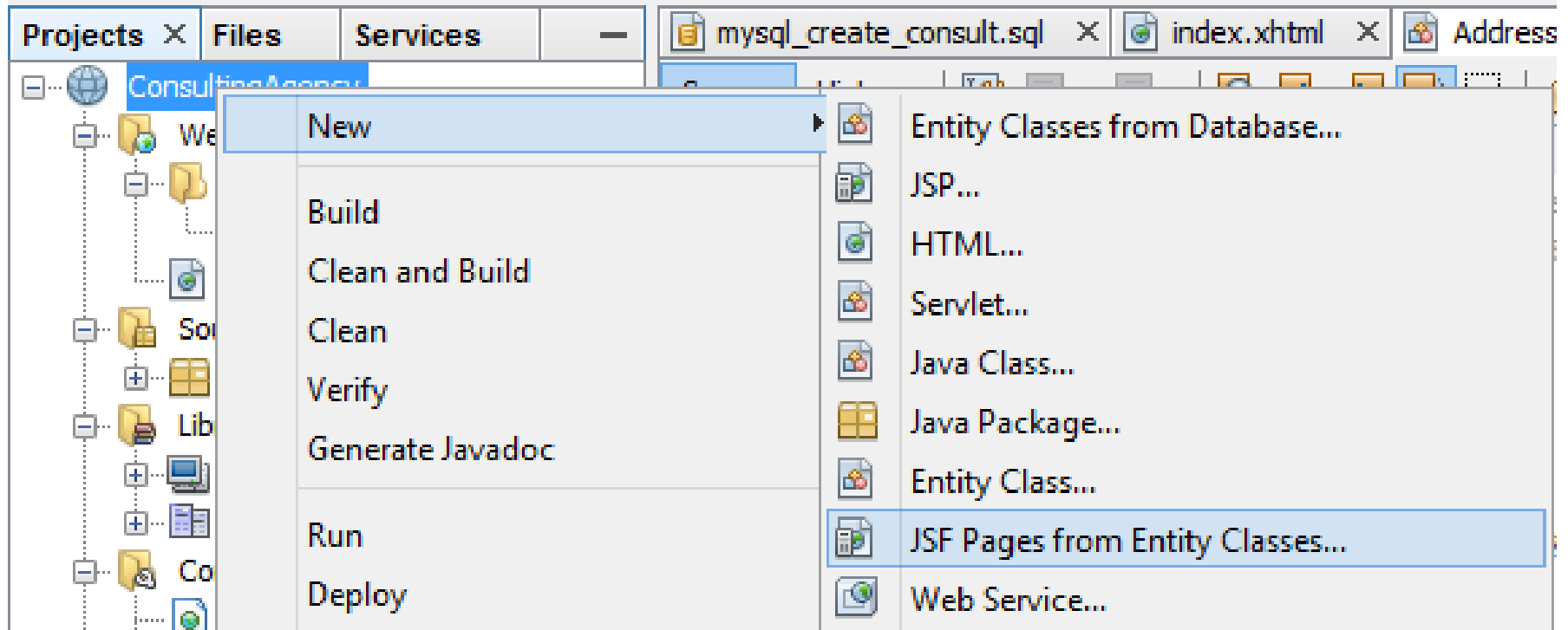
```
27 @Entity
28 @Table(name = "address")
29 @XmlRootElement
30 @NamedQueries({
31     @NamedQuery(name = "Address.findAll",
32     @NamedQuery(name = "Address.findByAddr
33     @NamedQuery(name = "Address.findByLine
34     @NamedQuery(name = "Address.findByLine
35     @NamedQuery(name = "Address.findByCity
36     @NamedQuery(name = "Address.findByRegi
37     @NamedQuery(name = "Address.findByCoun
38     @NamedQuery(name = "Address.findByPost
39 public class Address implements Serializab
40 private static final long serialVersio
41 @Id
```

- Note the generated SQL & Annotations
- DB Note:
  - no entity class generated for join tables
  - additional classes generated for tables with composite primary keys

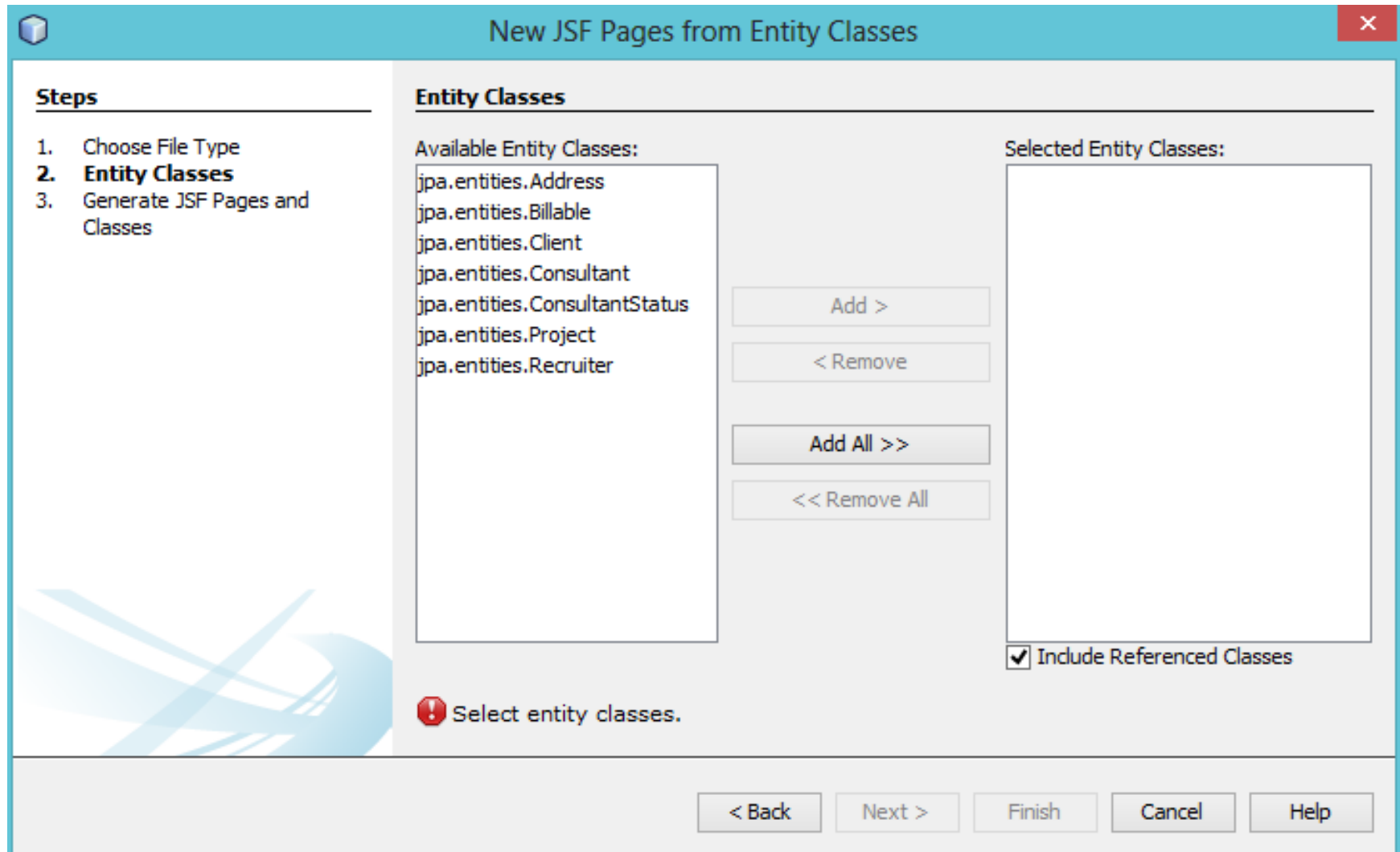
# Generate JSF Pages from Entity Classes

- What are the JSF Pages for?
  - viewing & modifying the data
  - that's what web apps do
- For each entity class:
  - a stateless session bean that extends `AbstractFacade.java`
  - a JSF session-scoped, managed bean
  - a directory containing four Facelets files for CRUD capabilities
- Also:
  - `AbstractFacade.java` class that contains the business logic for creation, retrieval, modification and removal of entity instances
  - utility classes, default stylesheet, localized message properties, etc.

# Generate JSF Pages from Entity Classes



# Add All >> to jpa.session package





# Add All >> to jpa.session package

**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 EJBs and the package of JSF classes.

Project: ConsultingAgency

Location: Source Packages

Session Bean Package: jpa.session

JSF Classes Package: jsf

Specify the location of new JSF pages.

JSF Pages Folder:  Browse...

Localization Bundle Name: /resources/Bundle

Override existing files

**Customize Templates**

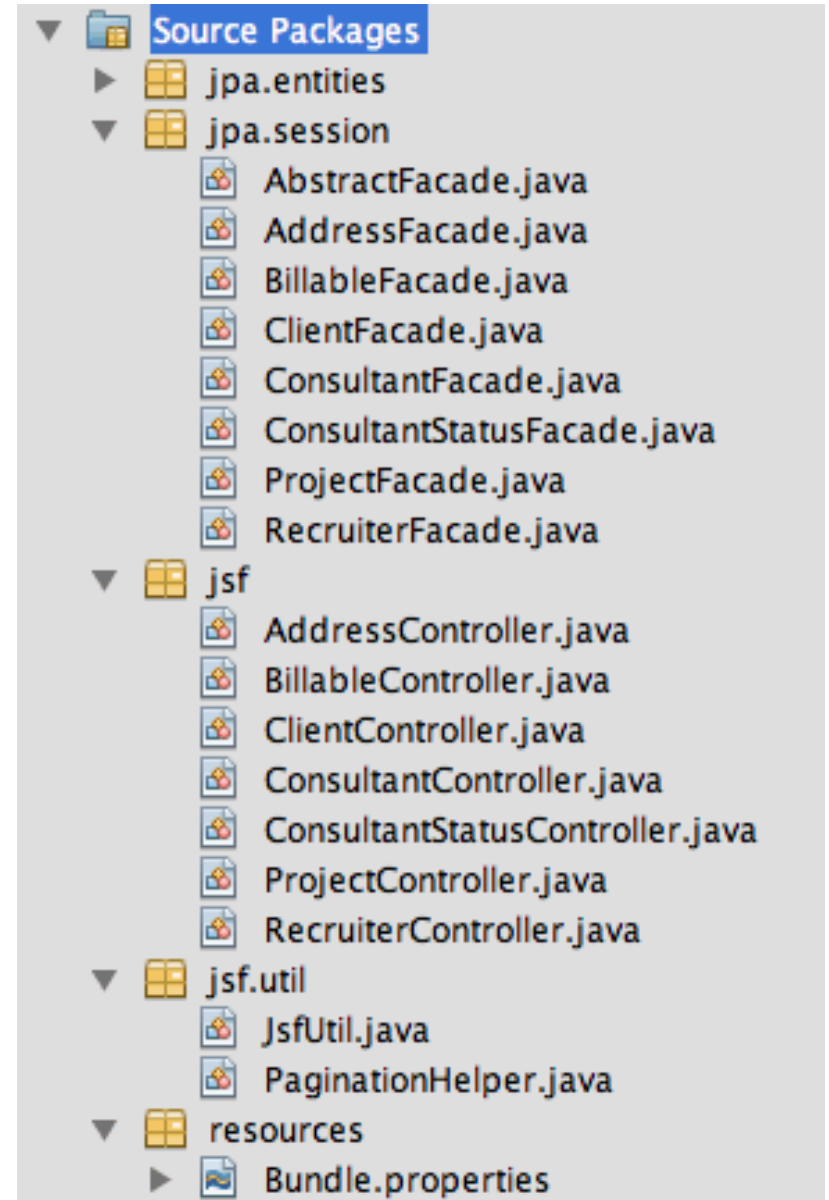
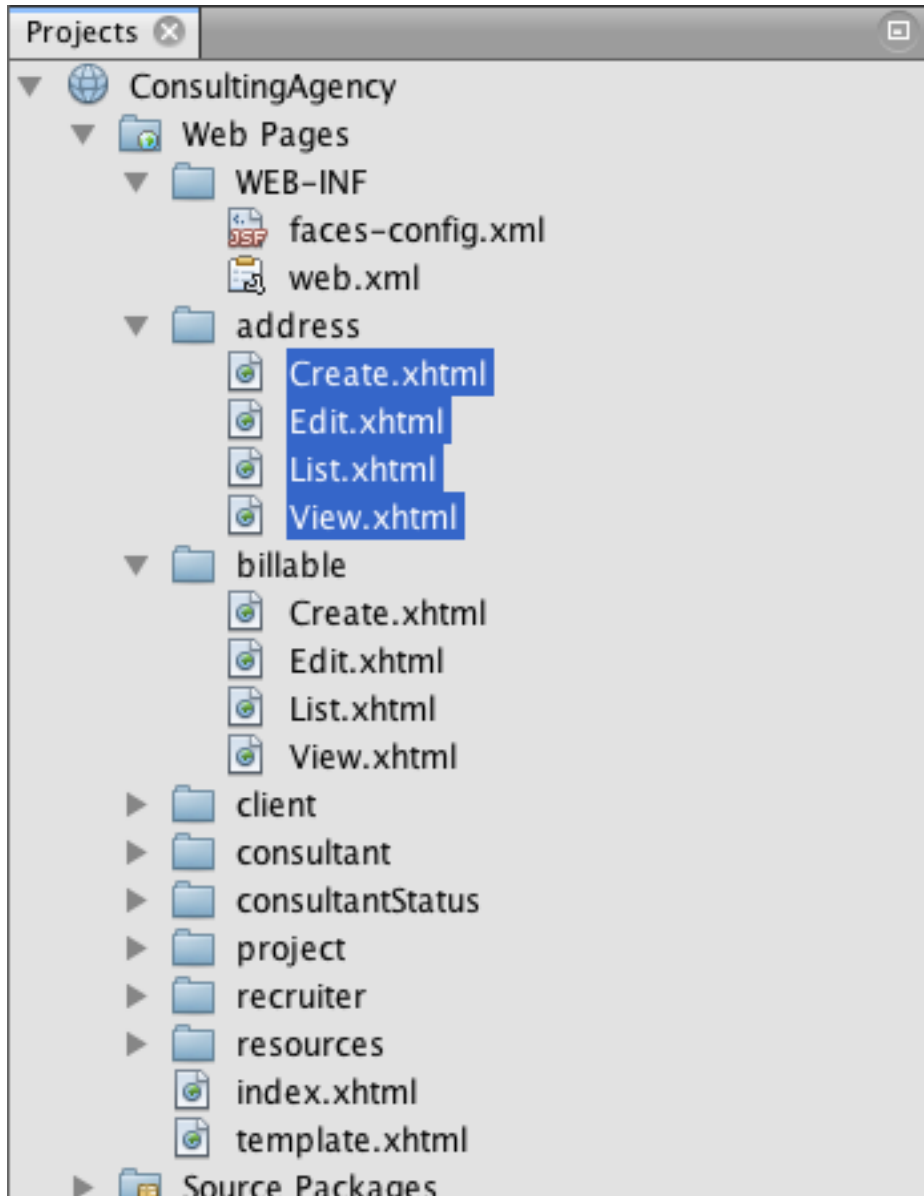
- All Templates
- View.xhtml Template
- Edit.xhtml Template
- Create.xhtml Template**
- List.xhtml Template
- EntityController.java Template
- PaginationHelper.java Template
- JsfUtil.java Template
- Bundle.properties Template

Next > Finish Cancel Help

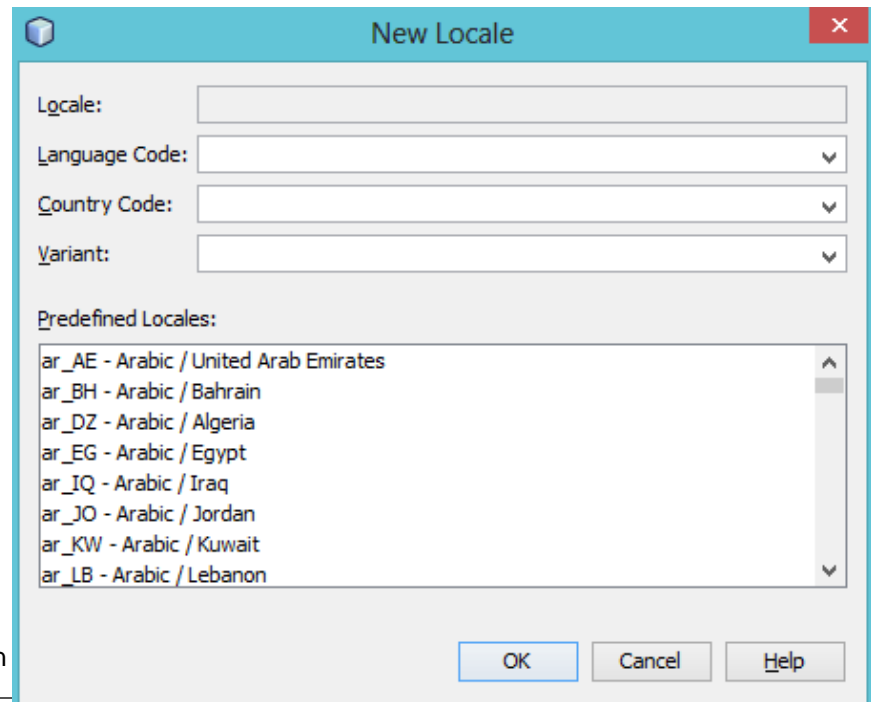
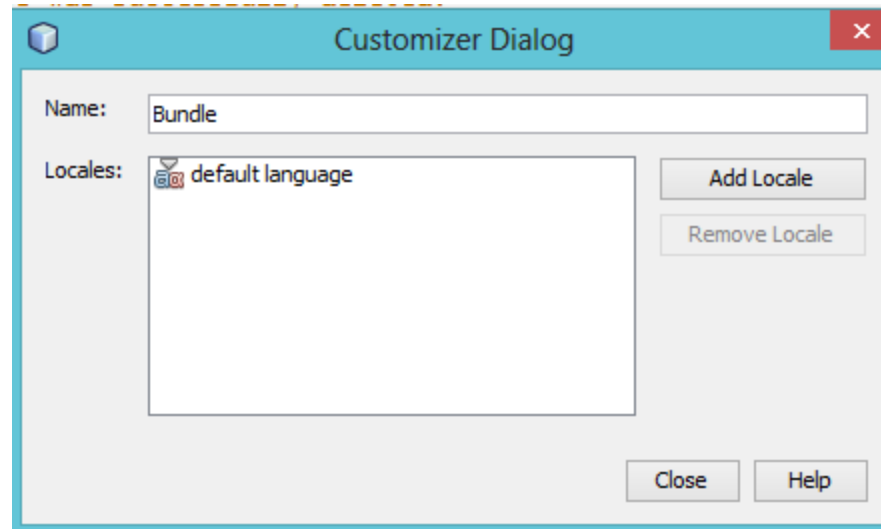
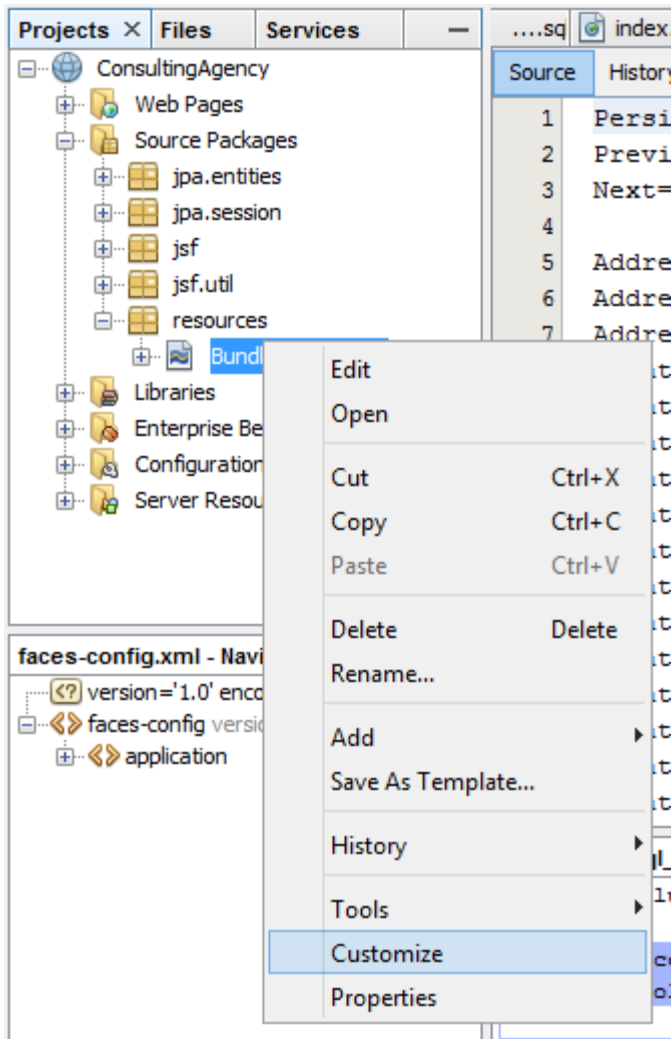
Successfully in 0.698 s, 0 rows affected.  
column 1

- Note the JSF classes package and /resources/Bundle

# Facelets & JSF Managed Beans



# You can change the language



# Run the Project

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](#)

## List

(No Address Items Found)

[Create New Address](#)

[Index](#)

## Create New Address

AddressId:

Line1:

Line2:

City:

Region:

Country:

PostalCode:

Client:

[Save](#)

[Show All Address Items](#)

[Index](#)

AddressId	Line1	Line2	City	Region	Country	PostalCode	Client	
1	321 Main Street		Cleveland	OH	USA	21111		<a href="#">View</a> <a href="#">Edit</a> <a href="#">Destroy</a>

# So what are the pieces & what are they doing?

- Entity classes
  - map Java data to database
  - provides queries for getting querying table
  
- Facade Bean classes
  - provides means for Creating, Retrieving, Updating, Deleting elements
  - Done through JPA's EntityManager
  
- Session Scoped managed bean controller classes:
  - AbstractFacade.java class that contains the business logic for creation, retrieval, modification and removal of entity instances
  - utility classes, default stylesheet, localized message properties, etc.