Your First Cup: An Introduction to the Java EE Platform

Part No: 821–1770–10
June 2010

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related software documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

100616@24378

Contents

Preface

This is *Your First Cup: An Introduction to Java Platform, Enterprise Edition*, a short tutorial for beginning Java EE programmers. This tutorial is designed to give you a hands-on lesson on developing an enterprise application from initial coding to deployment.

Who Should Use This Book

This tutorial is for novice Java EE developers. You should be familiar with the Java programming language, particularly the features introduced in Java Platform, Standard Edition 6. While familiarity with enterprise development and Java EE technologies is helpful, this tutorial assumes you are new to developing Java EE applications.

Before You Read This Book

Before you start this tutorial, you should:

- Be familiar with the Java programming language
- Be able to install software on your work machine
- Have a modern web browser installed on your work machine

Related Books and Projects

The following books and projects may be helpful to you in understanding this tutorial:

- *The Java EE 6 Tutorial*
- The Oracle GlassFish Server documentation set
- The NetBeans IDE documentation set

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Oracle is not responsible for the availability of third-party web sites mentioned in this document. Oracle does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Oracle will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

◆ ◆ ◆ CHAPTER 1

Introduction

An introduction to this tutorial. This chapter outlines the goals and the prerequisites for completing this tutorial.

Goals of This Tutorial

At the completion of this tutorial, you will:

- Understand the basics of tiered applications
- Understand the basics of the Java EE platform
- Have created a multi-tiered Java EE application
- Have deployed and run your application on a Java EE server
- Know where to go next for more information on the Java EE platform

Requirements for This Tutorial

A Checklist

To complete this tutorial, you need to:

- Get the Java EE 6 Software Development Kit.
- Get NetBeans IDE and all necessary plugins.
- Configure your environment.
- Get the latest updates to the tutorial bundle.

For up-to-the-minute information on which versions of the required software are compatible with this tutorial see the First Cup 2.0 compatibility page (`http://wiki.glassfish.java.net/Wiki.jsp?page=FirstCup2Compatibility`).

Getting the Java EE 6 SDK

To get the Java EE 6 SDK, go to `http://java.sun.com/javaee/downloads/index.jsp`.

Getting NetBeans IDE

To get NetBeans IDE go to `http://www.netbeans.org/downloads/index.html`.

Configuring Your Environment

Once you have all the necessary downloads, you must configure the tutorial bundle to reflect your environment.

▼ Adding GlassFish Server as a Server in NetBeans IDE

To run the tutorial examples in NetBeans IDE, you must register your GlassFish Server installation as a NetBeans Server Instance. Follow these instructions to register the GlassFish Server in NetBeans IDE.

1. **Select Tools → Server Manager to open the Servers dialog.**
2. **Click Add Server.**
3. **Under Server, select GlassFish v3 and click Next.**
4. **Under Platform Location, browse to or enter the location of your GlassFish Server installation.**
5. **Click Next.**
6. **Under Domain, use the drop-down list to select an existing domain, type in the path to the domain directly in the field, or type the name of a new domain to create.**
7. **Click Finish.**

Getting the Latest Updates to the Tutorial

Check for any updates to the tutorial by using the Update Center included with the Java EE 6 SDK.

▼ Updating the Tutorial Through the Update Center

Open the Update Center and check for any updates to the tutorial.

1. **Open the Services tab in NetBeans IDE and expand Servers.**
2. **Right-click the GlassFish Server instance and select View Update Center to display the Update Tool.**

3. **Select Available Updates in the tree to display a list of updated packages.**
4. **Look for updates to the First Cup 2. 0 for Java EE 6 (`javaee-firstcup-tutorial`) package.**
5. **If there is an updated version of First Cup 2.0, select First Cup 2. 0 for Java EE 6 (`javaee-firstcup-tutorial`) and click Install.**

◆ ◆ ◆ CHAPTER 2

Understanding Java Platform, Enterprise Edition

This chapter outlines the features of Java Platform, Enterprise Edition (Java EE), how it differs from Java Platform, Standard Edition (Java SE), Java Platform, Micro Edition (Java ME), and Java FX, and the basic concepts behind enterprise application development.

Differences between Java EE and Java SE

Java technology is both a programming language and a platform. The Java programming language is a high-level object-oriented language that has a particular syntax and style. A Java platform is a particular environment in which Java programming language applications run.

There are several Java platforms. Many developers, even long-time Java programming language developers, do not understand how the different platforms relate to each other.

The Java Programming Language Platforms

There are four platforms of the Java programming language:

- Java Platform, Standard Edition (Java SE)
- Java Platform, Enterprise Edition (Java EE)
- Java Platform, Micro Edition (Java ME)
- Java FX

All Java platforms consist of a Java Virtual Machine (VM) and an application programming interface (API). The Java Virtual Machine is a program, for a particular hardware and software platform, that runs Java technology applications. An API is a collection of software components that you can use to create other software components or applications. Each Java platform provides a virtual machine and an API, and this allows applications written for that platform to run on any compatible system with all the advantages of the Java programming language: platform-independence, power, stability, ease-of-development, and security.

Java SE

When most people think of the Java programming language, they think of the Java SE API. Java SE's API provides the core functionality of the Java programming language. It defines everything from the basic types and objects of the Java programming language to high-level classes that are used for networking, security, database access, graphical user interface (GUI) development, and XML parsing.

In addition to the core API, the Java SE platform consists of a virtual machine, development tools, deployment technologies, and other class libraries and toolkits commonly used in Java technology applications.

Java EE

The Java EE platform is built on top of the Java SE platform. The Java EE platform provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications.

Java ME

The Java ME platform provides an API and a small-footprint virtual machine for running Java programming language applications on small devices, like mobile phones. The API is a subset of the Java SE API, along with special class libraries useful for small device application development. Java ME applications are often clients of Java EE platform services.

Java FX

Java FX technology is a platform for creating rich internet applications written in Java FX Script. Java FX Script is a statically-typed declarative language that is compiled to Java technology bytecode, which can then be run on a Java VM. Applications written for the Java FX platform can include and link to Java programming language classes, and may be clients of Java EE platform services.

Overview of Enterprise Applications

This section describes enterprise applications and how they are designed and developed.

As stated above, the Java EE platform is designed to help developers create large-scale, multi-tiered, scalable, reliable, and secure network applications. A shorthand name for such applications is "enterprise applications," so called because these applications are designed to solve the problems encountered by large enterprises. Enterprise applications are not only useful for large corporations, agencies, and governments, however. The benefits of an enterprise application are helpful, even essential, for individual developers and small organizations in an increasingly networked world.

The features that make enterprise applications powerful, like security and reliability, often make these applications complex. The Java EE platform is designed to reduce the complexity of enterprise application development by providing a development model, API, and runtime environment that allows developers to concentrate on functionality.

Tiered Applications

In a multi-tiered application, the functionality of the application is separated into isolated functional areas, called tiers. Typically, multi-tiered applications have a client tier, a middle tier, and a data tier (often called the enterprise information systems tier). The client tier consists of a client program that makes requests to the middle tier. The middle tier's business functions handle client requests and process application data, storing it in a permanent datastore in the data tier.

Java EE application development concentrates on the middle tier to make enterprise application management easier, more robust, and more secure.

The Client Tier

The client tier consists of application clients that access a Java EE server and that are usually located on a different machine from the server. The clients make requests to the server. The server processes the requests and returns a response back to the client. Many different types of applications can be Java EE clients, and they are not always, or even often Java applications. Clients can be a web browser, a standalone application, or other servers, and they run on a different machine from the Java EE server.

The Web Tier

The web tier consists of components that handle the interaction between clients and the business tier. Its primary tasks are the following:

- Dynamically generate content in various formats for the client.
- Collect input from users of the client interface and return appropriate results from the components in the business tier.
- Control the flow of screens or pages on the client.
- Maintain the state of data for a user's session.
- Perform some basic logic and hold some data temporarily in JavaBeans components.

Java EE Technologies Used in the Web Tier

The following Java EE technologies are used in the web tier in Java EE applications:

TABLE 2–1 Web-Tier Java EE Technologies

Technology	Purpose
Servlets	Java programming language classes that dynamically process requests and construct responses, usually for HTML pages
JavaServer Faces technology	A user-interface component framework for web applications that allows you to include UI components (such as fields and buttons) on a page, convert and validate UI component data, save UI component data to server-side data stores, and maintain component state.
JavaServer Faces Facelets technology	Facelets applications are a type of JavaServer Faces applications that use XHTML pages rather than JSP pages.
Expression Language	A set of standard tags used in JSP and Facelets pages to refer to Java EE components.
JavaServer Pages (JSP)	Text-based documents that are compiled into servlets and define how dynamic content can be added to static pages, such as HTML pages.
JavaServer Pages Standard Tag Library	A tag library that encapsulates core functionality common to JSP pages
JavaBeans Components	Objects that act as temporary data stores for the pages of an application

The Business Tier

The business tier consists of components that provide the business logic for an application. Business logic is code that provides functionality to a particular business domain, like the financial industry, or an e-commerce site. In a properly designed enterprise application, the core functionality exists in the business tier components.

Java EE Technologies Used in the Business Tier

The following Java EE technologies are used in the business tier in Java EE applications:

- Enterprise JavaBeans (enterprise bean) components
- JAX-RS RESTful web services
- JAX-WS web service endpoints
- Java Persistence API entities

The Enterprise Information Systems Tier

The enterprise information systems (EIS) tier consists of database servers, enterprise resource planning systems, and other legacy data sources, like mainframes. These resources typically are located on a separate machine than the Java EE server, and are accessed by components on the business tier.

Java EE Technologies Used in the EIS Tier

The following Java EE technologies are used to access the EIS tier in Java EE applications:

- The Java Database Connectivity API (JDBC)
- The Java Persistence API
- The Java EE Connector Architecture
- The Java Transaction API (JTA)

Java EE Servers

A Java EE server is a server application that the implements the Java EE platform APIs and provides the standard Java EE services. Java EE servers are sometimes called application servers, because they allow you to serve application data to clients, much like web servers serve web pages to web browsers.

Java EE servers host several application component types that correspond to the tiers in a multi-tiered application. The Java EE server provides services to these components in the form of a *container*.

Java EE Containers

Java EE containers are the interface between the component and the lower-level functionality provided by the platform to support that component. The functionality of the container is defined by the platform, and is different for each component type. Nonetheless, the server allows the different component types to work together to provide functionality in an enterprise application.

The Web Container

The web container is the interface between web components and the web server. A web component can be a servlet, a JavaServer Faces Facelets page, or a JSP page. The container manages the component's lifecycle, dispatches requests to application components, and provides interfaces to context data, such as information about the current request.

The Application Client Container

The application client container is the interface between Java EE application clients, which are special Java SE applications that use Java EE server components, and the Java EE server. The application client container runs on the client machine, and is the gateway between the client application and the Java EE server components that the client uses.

The EJB Container

The EJB container is the interface between enterprise beans, which provide the business logic in a Java EE application, and the Java EE server. The EJB container runs on the Java EE server and manages the execution of an application's enterprise beans.

◆ ◆ ◆ CHAPTER 3

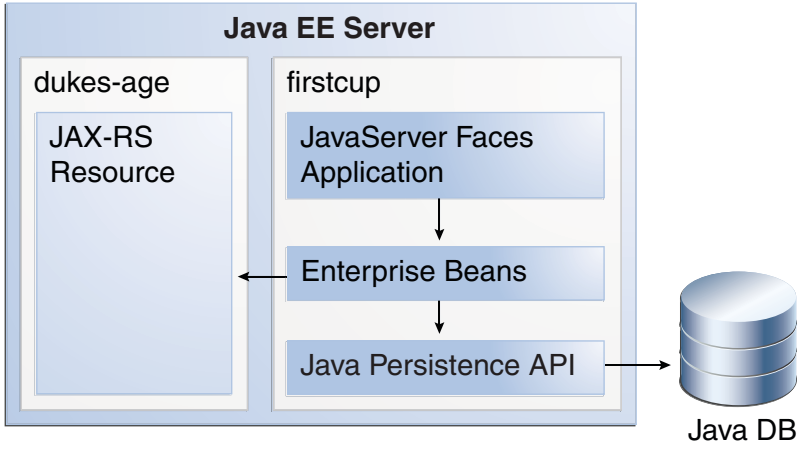
Creating Your First Java EE Application

This chapter gives an overview of the example application and step-by-step instructions on coding the example application.

Architecture of the Example Application

The example application consists of four main components: `DukesAgeResource`, a JAX-RS RESTful web service; `DukesBirthdayBean`, an enterprise bean; `FirstcupUser`, a Java Persistence API entity; and `firstcup`, a web application created with JavaServer Faces Facelets technology.

FIGURE 3–1 Architecture of the First Cup Example Application



`DukesAgeResource` is a JAX-RS resource that calculates the age of Duke, the Java mascot. Duke was born May 23, 1995, when the first demo of Java technology was publicly released.

`DukesBirthdayBean` is a local, no-interface view stateless session bean that calculates the difference between the user's age and Duke's age, and stores the user-submitted data in a Java Persistence API entity.

`FirstcupUser` is a Java Persistence API entity that represents a particular user's birthday. It is stored in a Java DB database table, and managed by `DukesBirthdayBean`'s business methods.

The `firstcup` web application is a JavaServer Faces Facelets application that accesses `DukesAgeResource` to display Duke's age, reads in a date provided by the user, accesses `DukesBirthdayBean` to calculate who is older, and then displays the difference in years between the user and Duke and the average age difference of all users.

The `firstcup` web application consists of the following:

- `greeting.xhtml`: An XHTML page that uses the JavaServer Faces Facelets tag libraries. Users can enter their birth date in a field and submit it for comparison against Duke's birth date.
- `response.xhtml`: A Facelets-enabled XHTML page that tells the user whether he or she is older or younger than Duke, based on the date the user entered in the `greeting.xhtml` page, and displays the average age difference of all users.
- `DukesBDay.java`: A JavaServer Faces managed bean that defines properties to hold the user's birth date, get Duke's current age from the `DukesAgeResource` web service, and get the age difference between the user and Duke from the enterprise bean.
- `web.xml`: The web application's deployment descriptor, which is used to configure certain aspects of a web application when it is installed. In this case, it is used to provide a mapping to the application's `FacesServlet` instance, which accepts incoming requests, passes them to the life cycle for processing, and initializes resources.
- `WebMessages.properties` and `WebMessages_es.properties`: Java programming language properties files that contain the localized strings used in `greeting.xhtml` and `response.xhtml`. By default, the English language strings in `WebMessages.properties` are used, but Spanish language strings are also provided in `WebMessages_es.properties`.
- `DukesBirthdayBean.java`: as described above, the enterprise bean packaged within the `firstcup` application. `DukesBirthdayBean` that calculates the difference between the user's birthday and Duke's birthday.

Tiers in the Example Application

The example application has one web tier component (the `firstcup` web client), three business tier components (the `DukesAgeResource` web service, the `FirstcupUser` entity, and the `DukesBirthdayBean` enterprise bean), and one enterprise information system (EIS) tier (the

data in the Java DB database table). The user's web browser is the client tier component, as it accesses the rest of the application through the web tier.

Java EE Technologies Used in the Example Application

The `DukesAgeResource` web service is a JAX-RS resource. The `DukesBirthdayBean` enterprise bean is a stateless session bean. The `FirstcupUser` entity is a Java Persistence API entity. The `firstcup` web client is a JavaServer Faces application that runs in the web container of the Java EE server.

Coding the Example Application

This section describes how to code the example application.

Getting Started

Before you start coding the example, you need to perform some configuration tasks:

1. Register the server with your NetBeans IDE as described in "Configuring Your Environment" on page 10.
2. Create a directory for the example you will build.

▼ Create a Directory for the Example

- **Create another directory at the same level as the *tut-install*`/example` directory, where *tut-install* is the location of the `firstcup` tutorial installation, and call it `myexample`. You'll create the applications described in this tutorial in this directory.**

Creating the Web Service

The `DukesAgeResource` endpoint is a simple RESTful web service. REST stands for *representational state transfer*, and software architectures that conform to the principles of REST are referred to as *RESTful*. RESTful web services are web-based applications that use the HTTP protocol to access, modify, or delete information contained within a *resource*. A RESTful web service resource is a source of specific information identifiable by a uniform resource identifier (URI), for example `http://example.com/someResource`, and may be manipulated by calling the HTTP protocol's methods, for example `GET` or `POST`.

RESTful web services are often contrasted to SOAP web services (for example web services created with the JAX-WS API that is part of Java EE 6). Compared to SOAP web services, RESTful web services are simpler, as they use HTTP directly rather than as a transport mechanism for an underlying XML document format, and typically offer better performance.

Web services are designed to be independent of their clients. Typically RESTful web services are publicly available to a wide variety of clients, and the clients are located throughout the internet. This is called "loose coupling," as the clients and servers are connected only by the standard HTTP-based requests and responses, and do not need to know each other's implementation details. For this reason, `DukesAge` will be developed in its own application module, and deployed separately from the `DukesBirthdayBean` enterprise bean and `firstcup` web client. `DukesAge` could be deployed on a completely different machine without affecting the functionality of the `firstcup` web client.

JAX-RS Resources

`DukesAgeResource` is a JAX-RS resource class that responds to HTTP `GET` requests and returns a `String` representing the age of Duke at the time of the request.

To create `DukesAgeResource`, use the wizard provided by the JAX-RS plug-in for NetBeans IDE to generate the resource class. This class is annotated with the `javax.ws.rs.Path` annotation, which specifies the URL suffix to which the resource will respond. `DukesAgeResource` has a single method, `getText`, annotated with the `javax.ws.rs.GET` and `javax.ws.rs.Produces` annotations. `@GET` marks the method as a responder to HTTP `GET` requests, and `@Produces` specifies the MIME-type of the response sent back from `getText` to clients. In this case, the MIME-type is `text/plain`.

Creating the Endpoint

In NetBeans IDE, create a web project with a source file called `DukesAgeResource.java` in the `firstcup.webservice` package using the RESTful Web Service wizard.

▼ Create the Project in NetBeans

1. **Select File → New Project.**
2. **Select Java Web in the Categories pane.**
3. **Select Web Application in the Projects pane.**
4. **Click Next.**
5. **Set Project Name to `dukes-age`.**
6. **Set the Project Location to *tut-install*`/myexample`.**
7. **Click Next.**
8. **Select your GlassFish Server instance from the Server menu.**

9. **Select Java EE 6 Web from the Java EE Version menu.**
10. **Set the Context Path to** `/DukesAgeService`
11. **Click Finish.**
    You should now see the module you created in the Projects pane.
12. **From the Projects pane, right-click on the** `index.jsp` **file and select Delete. Click Yes in the dialog.**

▼ Create the `DukesAgeResource` Class

1. **Make sure** `dukes-age` **is selected in the Project menu.**
2. **Select File → New File.**
3. **Select Web Services in the Categories pane.**
4. **Select RESTful Web Services From Patterns in the File Types pane.**
5. **Click Next.**
6. **Under Select Pattern select Singleton and click Next.**
7. **Set Resource Package to** `firstcup.webservice`**.**
8. **Under Path enter** `dukesAge`**.**
9. **Under Class Name enter** `DukesAgeResource`**.**
10. **Under MIME Type select** `text/plain`**.**
11. **Click Finish.**
    You should now see the `DukesAgeResource.java` file inside the `firstcup.webservice` package in the Projects pane. The `DukesAgeResource.java` file should also be open in the editor pane.

▼ Configure the `dukes-age` Web Application

By default, NetBeans IDE bundles the JAX-RS 1.0 JARs with web applications that use JAX-RS 1.0. GlassFish Server already has the JAX-RS 1.0 JARs in the server classpath, so there is no need to separately include the JARs.

The default URL that is brought up in a web browser when you run `dukes-age` can also be configured in NetBeans IDE.

1. **Right-click on `dukes-age` in the Projects tab and select Properties.**
2. **Click Libraries.**
3. **Uncheck the boxes under Compile-time Libraries for restlib-gfv3ee6.**
4. **Click Run.**
5. **Set Relative URL to `/resources/dukesAge`.**
6. **Click OK.**

Remove the `putText` Method

The `DukesAgeResource` JAX-RS resource doesn't respond to HTTP `PUT` requests. Delete the generated `putText` method in `DukesAgeResource`.

- **Highlight the following generated Javadoc and method definition and delete it.**

```
/**
 * PUT method for updating or creating an instance of DukesAgeResource
 * @param content representation for the resource
 * @return an HTTP response with content of the updated or created resource.
 */
@PUT
@Consumes("text/plain")
public void putText(String content) {
}
```

Implement the `getText` Method

Add code to `DukesAgeResource.getText` that calculates Duke's age at the time of the request. To do this, use the `java.util.Calendar` and `java.util.GregorianCalendar` classes to create an object representing the date May 23, 1995, Duke's birthday. Then create another `Calendar` object representing today's date, and subtract today's year from Duke's birth year. If today's date falls before May 23, subtract a year from this result. Then return the result as a `String` representation.

1. **Highlight the current code in `getText` and replace it with the following code:**

```
// Create a new Calendar for Duke’s birthday
Calendar dukesBirthday = new GregorianCalendar(1995, Calendar.MAY, 23);
// Create a new Calendar for today
Calendar now = GregorialCalendar.getInstance();

// Subtract today’s year from Duke’s birth year, 1995
int dukesAge = now.get(Calendar.YEAR) - dukesBirthday.get(Calendar.YEAR);
dukesBirthday.add(Calendar.YEAR, dukesAge);
```

```
// If today's date is before May 23, subtract a year from Duke's age
if (now.before(dukesBirthday)) {
    dukesAge--;
}
// Return a String representation of Duke's age
return new String("" + dukesAge);
```

2. **Right-click in the editor window and select Format.**
3. **Right-click in the Editor and select Fix Imports.**
4. **Select File → Save from the menu to save the file.**

Building and Deploying the Web Service

Build the JAX-RS web application and deploy it to your GlassFish Server instance.

▼ Building and Deploying the Web Service Endpoint

Compile, package, and deploy `dukes-age.war` to GlassFish Server. This task gives instructions on deploying `dukes-age.war` in NetBeans IDE.

1. **Select** `dukes-age` **in the Projects tab.**
2. **Right-click** `dukes-age` **and select Run.**

   After `dukes-age.war` deploys successfully to GlassFish Server a web browser will load the URL of the `DukesAgeResource` path, and you'll see the returned `String` representing Duke's age.

Example 3–1 Output of `DukesAgeResource`

Here's an example of the output of `DukesAgeResource` displayed in a web browser.

```
14
```

Creating the `firstcup` Project

The `firstcup` web application project consists of the Java Persistence API entity, the enterprise bean, and the JavaServer Faces web front-end.

▼ Create the Web Application Project

Follow these steps to create a new web application project in NetBeans IDE.

1. **Select File → New Project.**
2. **Under Categories select Java Web, then under Projects select Web Application, and click Next.**
3. **Set the Project Name to `firstcup`.**
4. **Set the Project Location to *tut-install*`/myexample`, where *tut-install* is the location of the `firstcup` tutorial installation.**
5. **Click Next.**
6. **Select your GlassFish Server instance from the Server menu.**
7. **Select Java EE 6 from the Java EE Version menu.**
8. **Set Context Path to `/firstcup` and click Next.**
9. **Under Frameworks select JavaServer Faces.**
10. **Click the Configuration Tab, then under Servlet URL Pattern enter `/firstcupWeb/*` and click Finish.**

Creating the Java Persistence API Entity

The Java Persistence API allows you to create and use Java programming language classes that represent data in a database table. A Java Persistence API *entity* is a lightweight, persistent Java programming language object that represents data in a data store. Entities can be created, modified, and removed from the data store by calling the operations of the Java Persistence API *entity manager*. Entities, or the data encapsulated by the persistent fields or properties of a entity, can be queried using the Java Persistence Query Language (JPQL), a language similar to SQL that operates on entities.

In `firstcup`, there is a single entity that defines one query.

▼ Creating the `FirstcupUser` Entity Class

The `FirstcupUser` Java Persistence API entity represents a particular `firstcup` user, and stores the user's birthday and the difference in age between the user and Duke. `FirstcupUser` also defines a Java Persistence API query used to calculate the average age difference of all users.

1. **With the `firstcup` project selected in the Projects pane, select File → New File.**

2. **Under Categories select Persistence, then under File Types select Entity Class, and click Next.**
3. **Enter `FirstcupUser` under Class Name and `firstcup.entity` under Package.**
4. **Click Create Persistence Unit, select `jdbc/__default` under Data Source, and click Create.**
5. **Click Finish.**

Add Properties to the `FirstcupUser` Entity

Create the `FirstcupUser` entity's two properties: `birthday`, of type `java.util.Calendar`; and `ageDifference`, of type `int`.

The `birthday` property must be annotated with the `javax.persistence.Temporal` annotation to mark the property as date field in the underlying database table. All persistent fields or properties of type `java.util.Calendar` or `java.util.Date` must be annotated with `@Temporal`.

1. **Right-click the editor window, select Insert Code, then Add Property.**
2. **In the Add Property dialog, enter `birthday` under Name, `java.util.Calendar` under Type, and click OK.**
3. **Click the error glyph next to the new `birthday` field and select `Add @Temporal Annotation`.**
4. **Right-click the editor window, select Insert Code, then Add Property.**
5. **In the Add Property dialog, enter `ageDifference` under Name, `int` under Type, and click OK.**

Add a Named Query to the `FirstcupUser` Entity

Add a JPQL named query to the `FirstcupUser` entity that returns the mean average age difference of all `firstcup` users.

This query uses the `AVG` aggregate function to return the mean average of all the values of the `ageDifference` property of the `FirstcupUser` entities.

- **Directly before the class definition, paste in the following code:**

```
@NamedQuery(name="findAverageAgeDifferenceOfAllFirstcupUsers",
query="SELECT AVG(u.ageDifference) FROM FirstcupUser u)
```

The `@NamedQuery` annotation appears just before the class definition of the entity, and has two required attributes: `name`, with the unique name for this query; and `query`, the JPQL query definition.

▼ Add a Business Method to `DukesBirthdayBean` that Gets the Average Age Difference of `firstcup` Users

Once the `FirstcupUser` entity is complete, add a business method to the `DukesBirthdayBean` session bean to call the named query that returns the average age difference of all users.

1. **Double-click `DukesBirthdayBean` in the Projects Pane under Enterprise Beans.**

2. **Below the class definition, add a `@PersistenceContext` annotation and field of type `EntityManager`:**

```
@PersistenceContext
private EntityManager em;
```

3. **Right-click in the Editor window, select Insert Code, then Add Business Method.**

4. **In the Add Business Method dialog, enter `getAverageAgeDifference` under Name, set the Return Type to `Double`, and click OK.**

5. **Replace the body of the newly created `getAverageAgeDifference` method with the following code:**

```
public Double getAverageAgeDifference() {
    Double avgAgeDiff = (Double)
        em.createNamedQuery("findAverageAgeDifferenceOfAllFirstcupUsers")
        .getSingleResult();
    logger.info("Average age difference is: " + avgAgeDiff);
    return avgAgeDiff;
}
```

   The named query in `FirstcupUser` is called by using the `EntityManager`'s `createNamedQuery` method. Because this query returns a single number, the `getSingleResult` method is called on the returned `Query` object. The query returns a `Double`.

6. **Select File → Save.**

Creating the Enterprise Bean

`DukesBirthdayBean` is a *stateless session bean*. Stateless session beans are enterprise beans that do not maintain a conversational state with a client. With stateless session beans the client makes isolated requests that do not depend on any previous state or requests. If an application requires conversational state, use *stateful session beans*.

To create `DukesBirthdayBean` create one Java class: `DukesBirthdayBean`, the enterprise bean class. `DukesBirthdayBean` is a *local enterprise bean* that uses a *no-interface* view, meaning two things. First, a local enterprise bean is only visible within the application in which it is deployed. Second, enterprise beans with a no-interface view do not need a separate business interface that

the enterprise bean class implements. The enterprise bean class is the only coding artifact needed to create a local, no-interface enterprise bean.

`DukesBirthdayBean` will be packaged within the same WAR file as the Facelets web front-end.

Creating `DukesBirthdayBean` in NetBeans IDE

This section has instructions for creating the web application project and the `DukesBirthdayBean` enterprise bean.

▼ Creating the `DukesBirthdayBean` Enterprise Bean Class

Follow these steps to create the enterprise bean class in NetBeans IDE.

1. **Select `firstcup` project in the Projects tab.**
2. **Select File → New File.**
3. **Select Java EE in the Categories pane.**
4. **Select Session Bean in the File Types pane and click Next.**
5. **Set EJB Name to `DukesBirthdayBean`.**
6. **Set the Package name to `firstcup.ejb`.**
7. **Select Stateless under Session Type.**
8. **Click Finish.**

▼ Modify `DukesBirthdayBean.java`

Add a `java.util.Logger` instance and a business method that calculates the difference in age in years between Duke and the user, and creates a new `FirstcupUser` entity.

1. **Directly after the class declaration, paste in the following code:**

```
private static Logger logger =
        Logger.getLogger("firstcup.ejb.DukesBirthdayBean");
```

This code creates a logger for the session bean.

2. **Right-click in the editor window and select Insert Code, then Add Business Method.**
3. **Set the name to `getAgeDifference` and the return type to `int`.**

4 **Under Parameters click Add, set the Name to `date` and the Type to `java.util.Date`, then click OK.**

5 **Replace the body of the `getAgeDifference` method with the following code:**

```
int ageDifference;

Calendar theirBirthday = new GregorianCalendar();
Calendar dukesBirthday = new GregorianCalendar(1995, Calendar.MAY, 23);

// Set the Calendar object to the passed in Date
theirBirthday.setTime(date);

// Subtract the user's age from Duke's age
ageDifference = dukesBirthday.get(Calendar.YEAR) -
theirBirthday.get(Calendar.YEAR);
logger.info("Raw ageDifference is: " + ageDifference);
// Check to see if Duke's birthday occurs before the user's. If so,
// subtract one from the age difference
if (dukesBirthday.before(theirBirthday) && (ageDifference > 0)) {
    ageDifference--;
}

// create and store the user's birthday in the database
FirstcupUser user = new FirstcupUser(date, ageDifference);
em.persist(user);

logger.info("Final ageDifference is: " + ageDifference);

return ageDifference;
```

This method creates the `Calendar` objects used to calculate the difference in age between the user and Duke and performs the actual calculation of the difference in age.

Similar to the `DukesAgeResource.getText` code, `getAgeDifference` subtracts Duke's birthday year from the user's birthday year to get a raw age difference. If Duke's birthday falls before the users, and the raw difference is more than 0, subtract one year from the age difference.

A new `FirstcupUser` entity is created with the user's birthday and age difference, then stored in the JavaDB database by calling the `EntityManager`'s `persist` method.

The final age difference is returned as an `int`.

6 **Right-click in the editor window and select Format.**

7 **Right-click in the editor window and select Fix Imports.**

8 **Choose the `java.util.logging.Logger` fully-qualified name for the `Logger` class.**

9 **Click OK.**

10 **Select File → Save.**

Creating the Web Client

To create the web client, you need to perform the following tasks:

- Create a resource bundle to hold localized messages used by the Facelets pages.
- Configure the resource bundle in the configuration file.
- Create the `DukesBDay` managed bean class.
- Create the Facelets pages.

Creating a Resource Bundle

In this section, we'll create the resource bundle that contains the static text and error messages used by the Facelets pages. The `firstcup` client supports both English and Spanish locales. Therefore we need to create two properties files, each of which will contain the messages for one of the locales.

▼ Creating a Resource Bundle

1 **Right-click `firstcup` in the Projects pane.**

2 **Select New → Other from the popup menu.**

3 **Select the Other category, then Properties File from the New File dialog and click Next.**

4 **In the New Properties File dialog, enter `WebMessages` in the File Name field.**

5 **In the Folder field, enter `src/java/firstcup/web` as the location of the file.**

6 **Click Finish.**

7 **After NetBeans IDE creates the properties file, enter the following messages or copy them from here to the file:**

```
Welcome=Hi. My name is Duke. Let us find out who is older -- You or me.
DukeIs=Duke is
YearsOldToday=years old today.
Instructions=Enter your birthday and click submit.
YourBD=Your birthday
Pattern=MM/dd/yyyy
DateError=Please enter the date in the form MM/dd/yyyy.
YouAre=You are
```

```
Year=year
Years=years
Older=older than Duke!
Younger=younger than Duke!
SameAge= the same age as Duke!
Submit=Submit
Back=Back
AverageAge=The average age difference of all First Cup users is
```

These messages will be referenced from the XHTML pages.

8 **Save the file by selecting File → Save from the menu.**

9 **To add the Spanish translations of the messages, copy the properties file** `WebMessages_es.properties` **from** *tut-install*`/firstcup/example/firstcup/src/java/com/sun/firstcup/web` **to** *tut-install*`/firstcup/myexample/firstcup/src/java/firstcup/web`**.**

You can create multiple properties files, each with a set of messages for a different locale. By storing localized static text and messages in resource bundles, you don't need to create a separate set of XHTML pages for each locale.

Configuring the Resource Bundle in the Configuration File

To make the resource bundle available to the application, you need to configure it in the configuration file, by performing the following task.

▼ Creating a Configuration File

The `faces-config.xml` deployment descriptor contains configuration settings for the JavaServer Faces application. JSF applications don't require a deployment descriptor unless they use features that can only be configured in `faces-config.xml`. In `firstcup`, the deployment descriptor has settings defining the resource bundle that provides localized strings in English and Spanish.

1 **Select** `firstcup` **in the Projects Pane and select File → New File.**

2 **In the New File dialog, select JavaServer Faces under Categories, new JavaServer Face Configuration under File Types, click Next, then Finish.**

▼ Configuring the Resource Bundle

The `firstcup` application is localized for English and Spanish languages. JavaServer Faces applications can automatically select the proper language based on the locale of the user's web browser. Specify the default and supported locales in the `faces-config.xml` file.

1 **With the newly created** `faces-config.xml` **file open, click XML.**

2 **Add the following `<application>` tag to configure the resource bundle:**

```
<application>
    <resource-bundle>
        <base-name>firstcup.web.WebMessages</base-name>
        <var>bundle</var>
    </resource-bundle>
    <locale-config>
        <default-locale>en</default-locale>
        <supported-locale>es</supported-locale>
    </locale-config>
</application>
```

The `base-name` element of the `resource-bundle` element identifies the fully-qualified class name of the resource bundle. The `var` element identifies the name by which the XHTML pages will reference the resource bundle. The `locale-config` element identifies the locales supported by the resource bundle.

3 **Right-click in the editor window and select Format.**

4 **Select File → Save.**

Creating the `DukesBDay` Managed Bean Class

The `DukesBDay` JavaBeans component is a backing bean. A backing bean is a JavaServer Faces managed bean that acts as a temporary data storage for the values of the components included on a particular JavaServer Faces page. A managed bean is a JavaBeans component that a JavaServer Faces application instantiates and stores in scope. The section following this one describes more about managed beans and how to configure them.

This section describes how to create the `DukesBDay` class. To create the class you need to do the following:

- Create the managed bean class.
- Add a property that stores Duke's current age from the JAX-RS web service.
- Add a property that stores the user's current birth date.
- Add a property that stores the age difference from the `DukesBirthdayBean` enterprise bean.
- Add a property that stores the absolute value of the age difference.
- Add a property that stores the average age difference of all users.
- Add a method that calls `DukesBirthdaybean.getAgeDifference`, sets the absolute age difference, and forwards the user to the display page.

Creating the Managed Bean Class.

Create a JavaServer Faces managed bean class that will subsequently be modified.

1. **Right-click the `firstcup.web` package in the Projects pane.**
2. **Select New → Other.**
3. **Select JavaServer Faces under Categories, then JSF Managed Bean under File Types.**
4. **Enter `DukesBDay` in the Class Name field.**
5. **Select `firstcup.web` in the Package field.**
6. **Select session under Scope.**
7. **Click Finish.**

Adding an Enterprise Bean Reference

Add a `javax.ejb.EJB` annotation to inject a reference to the `DukesBirthdayBean` enterprise bean. This session bean will be called from the methods in `DukesBDay`.

1. **Right-click in the editor window and select Call Enterprise Bean.**
2. **In the Call Enterprise Bean dialog expand the `firstcup` application, select `DukesBirthdayBean`, and click OK. The following field will be added:**

```
@EJB
private DukesBirthdayBean dukesBirthdayBean;
```

Adding Properties to the Bean

During this task, you will add the following properties to the `DukesBDay` bean:

- `age` for getting Duke's age from the web service.
- `yourBD` to hold the user's birth date.
- `ageDiff` to get the age difference from the enterprise bean.
- `absAgeDiff` to hold the absolute value of the age difference.

1. **Right-click in the Editor window, select Insert Code, then Add Property.**
2. **Enter `age` under Name, and set the Type to `int`, click OK.**
3. **Repeat the above steps to create the following properties: `yourBD` of type `java.util.Date`, `ageDiff` of type `int`, `absAgeDiff` of type `int`, and `averageAgeDifference` of type `Double`.**

4 **After the newly created property fields, add the following `Logger` instance:**

```
private static Logger logger = Logger.getLogger("firstcup.web.DukesBDay");
```

5 **Initialize the variables in the default constructor:**

```
public DukesBDay() {
    age = -1;
    yourBD = null;
    ageDiff = -1;
    absAgeDiff = -1;
}
```

6 **Right-click in the editor and select Format.**

7 **Right-click in the editor and select Fix Imports.**

8 **Select `java.util.logging.Logger` for the `Logger` class, and click OK.**

▼ Getting Duke's Current Age

While performing this task, you will add some code to the `getAge` method to access Duke's current age from the JAX-RS web service.

Use the `java.net` and `java.io` classes to create an HTTP connection to the Duke's Age web service and read in the result.

1 **Add the following code to the `getAge` method.**

```
public int getAge() {
    // Use the java.net.* APIs to access the Duke's Age RESTful web service
    HttpURLConnection connection = null;
    BufferedReader rd = null;
    StringBuilder sb = null;
    String line = null;
    URL serverAddress = null;

    try {
        serverAddress = new URL(
                "http://localhost:8080/DukesAgeService/resources/dukesAge");
        connection = (HttpURLConnection) serverAddress.openConnection();
        connection.setRequestMethod("GET");
        connection.setDoOutput(true);
        connection.setReadTimeout(10000);

        // Make the connection to Duke's Age
        connection.connect();

        // Read in the response
```

```
            rd = new BufferedReader(
                    new InputStreamReader(connection.getInputStream()));
            sb = new StringBuilder();
            while ((line = rd.readLine()) != null) {
                sb.append(line);
            }

            // Convert the response to an int
            age = Integer.parseInt(sb.toString());
        } catch (MalformedURLException e) {
            logger.warning("A MalformedURLException occurred.");
            e.printStackTrace();
        } catch (ProtocolException e) {
            logger.warning("A ProtocolException occurred.");
            e.printStackTrace();
        } catch (IOException e) {
            logger.warning("An IOException occurred");
            e.printStackTrace();
        }

        return age;
    }
```

2 **Right-click in the editor window and select Format.**

3 **Right-click in the editor window and select Fix Imports.**

4 **Click OK.**

▼ Getting the Age Difference From the `DukesBirthdayBean` Enterprise Bean

During this task, you will create a `processBirthday` method to get the difference in age between the user's age and Duke's age from the EJB, set the `absAgeDiff` variable to the absolute value of the age difference, and set a result string that is will forward the user to the display page..

1 **Add a `processBirthday` method with the following code:**

```
public String processBirthday() {
    this.setAgeDiff(dukesBirthday.getAgeDifference(yourBD));
    logger.info("age diff from dukesbday " + ageDiff);
    this.setAbsAgeDiff(Math.abs(this.getAgeDiff()));
    logger.info("absAgeDiff " + absAgeDiff);
    return new String("success");
}
```

This method calls the `getAgeDifference` method of `DukesBirthdayBean` to get the age difference and store it in the `ageDiff` property, sets the absolute age difference stored in the

`absAgeDiff` property, and returns a status code of `success`. This status code will be used by the JavaServer Faces runtime to forward the user to the appropriate page.

2. **Right-click in the editor window and select Format.**

3. **Select File → Save.**

Creating the Facelets Client

The Facelets client consists of a *resource library*, a *composite component*, and two XHTML files.

Resource Libraries in `firstcup`

A JavaServer Faces resource library is a collection of user-created components collected in a standard location in a web application. Resource libraries are identified according to a *resource identifier*, a string that represents a particular resource within a web application. Resources can be packaged either at the root of the web application or on the web application's classpath.

A resource packaged in the web application root must be in a subdirectory of a `resources` directory at the web application root.

`resources/`*resource identifier*

A resource packaged in the web application classpath must be in a subdirectory of the `META-INF/resources` directory within a web application.

`META-INF/resources/`*resource identifier*

Resource identifiers are unique strings that conform to the following format:

`[`*localePrefix*`/][`*libraryName*`/][`*libraryVersion*`/]`*resource name*`[/`*resourceVersion*`]`

Elements of the resource identifier in brackets (`[]`) are optional. A resource name, identifying a particular resource (a file or a graphic, for example), is required. In `firstcup`, a resource library with the name `components` is packaged in the web application root, and this library contains one resource, a file called `inputDate.xhtml`. The resource identifier for this resource is therefore `components/inputDate.xhtml`, and it is located in the web application root at `resources/components/inputDate.xhtml`.

The `inputDate` Composite Component

A composite component is a set of user-defined JavaServerFaces and Facelets components located in a resource. In `firstcup`, the `inputDate.xhtml` resource, located in the `components` resource library is a composite component that contains tags for reading in a date the user enters in a form. Composite components consist of an *interface* definition and an *implementation*.

The interface definition is specified with the `<composite:interface>` tag to define which attributes are exposed to pages that use the composite component. Attributes are identified with the `<composite:attribute>` tag.

EXAMPLE 3–2 `inputDate` Composite Component Interface Definition

The `inputDate.xhtml` interface definition is as follows. It defines a single attribute, `date`, that must be specified in pages that use the `inputDate` composite component.

```
<composite:interface>
    <composite:attribute name="date" required="true" />
</composite:interface>
```

The implementation of the composite component is specified with the `<composite:implementation>` tag. The tags within the `<composite:implementation>` are the actual component tags that will be added to pages that use the composite component. They can be any HTML Render Kit, JavaServer Faces, or Facelets tags. The `#{cc.attrs.`*attribute name*`}` expression is used to get the value of the specified attribute from the page or component that is using the composite component.

EXAMPLE 3–3 The `inputDate` Composite Component Implementation

The implementation of the `inputDate` composite component is as follows. An HTML input text component will store the entered text into the `date` attribute, accessed by the `#{cc.attrs.date}` expression. A JavaServer Faces `convertDateTime` component will convert the entered text to a date with the form of `MM/dd/yyyy` (04/13/2009, for example).

```
<composite:implementation>
    <h:inputText value="#{cc.attrs.date}">
        <f:convertDateTime pattern="MM/dd/yyyy" />
    </h:inputText>
</composite:implementation>
```

▼ Creating the `inputDate` Composite Component

Create the `inputDate` composite component as a resource in the `components` resource library.

1. **In the `firstcup` project, select File → New File.**
2. **Select JavaServer Faces under Categories, JSF Composite Component under File Types, and click Next.**
3. **In the New JSF Composite Component dialog, enter `inputDate` under File Name, `components` under Folder, and click Finish.**

4. **Add the composite component interface definition between the** `<body>` **and** `</body>` **tags in** `inputDate.xhtml`**:**

```
<composite:interface>
    <composite:attribute name="date" required="true" />
</composite:interface>
```

5. **Add the composite component implementation below the interface definition:**

```
<composite:implementation>
    <h:inputText value="#{cc.attrs.date}">
        <f:convertDateTime pattern="MM/dd/yyyy" />
    </h:inputText>
</composite:implementation>
```

6. **Right-click in the editor window and select Format.**

7. **Select File → Save.**

The Facelets Web Interface

The `firstcup` web application interface has two XHTML files. The `greeting.xhtml` file displays Duke's current age and the form to the user for her to enter her birthday. The `response.xhtml` file displays the age difference between the user and Duke.

The `greeting.xhtml` contains several pieces of the `firstcup` application detailed previously. It uses the localized strings contained in `WebMessages.properties` and `WebMessages_es.properties`. It uses the `DukesBDay` managed bean to call both the `DukesAgeResource` JAX-RS web service and the `DukesBirthdayBean` enterprise bean. It uses the `inputDate` composite component to create the input for the user to enter her birthday.

EXAMPLE 3–4 The `greeting.xhtml` File

Here's the content of the `greeting.xhtml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:fc="http://java.sun.com/jsf/composite/components">
<head>
    <title>Firstcup Greeting Page</title>
</head>
<body>
<h:form>
```

EXAMPLE 3–4 The `greeting.xhtml` File *(Continued)*

```
    <h2>
        <h:outputText value="#{bundle.Welcome}"/>
    </h2>
    <h:outputText value="#{bundle.DukeIs} "/>
    <h:outputText value="#{DukesBDay.age} #{bundle.YearsOldToday}"/>
    <p/>
    <h:outputText value="#{bundle.Instructions}"/>
    <p/>
    <h:outputText value="#{bundle.YourBD} "/>
    <fc:inputDate id="userBirthday" date="#{DukesBDay.yourBD}" />
    <h:outputText value=" #{bundle.Pattern}"/>
    <p/>
    <h:commandButton value="#{bundle.Submit}" action="response"/>
    <p/>
    <h:message for="userBirthday" style="color:red"/>
</h:form>
</body>
</html>
```

The `greeting.xhtml` file uses the JSF Core, HTML Render Kit, and the `components` resource library tag libraries. The `components` tag library has a prefix of `fc`, and is used to specify the `inputDate` composite component in the form below. The `<fc:inputDate id="userBirthday" date="#{DukesBDay.yourBD}" />` tag has the required `date` attribute, and stores the value in the `yourBD` property in the `DukesBDay` managed bean by using the EL expression `#{DukesBDay.yourBD}`.

The localized strings are referred to by using the EL expressions `#{bundle.`*property name*`}`. For example, the `<h:outputText value="#{bundle.Welcome}"/>` tag will display the following string in English locales:

```
Hi. I’m Duke. Let’s find out who’s older -- You or I.
```

The `<h:commandButton value="#{bundle.Submit}" action="response"/>` tag creates a submit button and specifies that a successful submission should render the `response.xhtml` file by setting the `action` attribute to `response`. The `action` attribute is used to define navigation rules for forms in Facelets pages.

If the form submission is unsuccessful, a warning message is displayed. This is done with the `<h:message for="userBirthday" style="color:red"/>` tag, which is connected to the `inputDate` composite component with the id `userBirthday`. That is, if there's an error with the input of the `inputDate` component, a warning message is displayed.

The `response.xhtml` displays the age difference between the user and Duke. Different strings are displayed based on whether the user is the same age, younger, or older than duke. The text can be displayed or not based on the conditions specified by the `rendered` attribute of the

<h:outputText> tag. The conditions used in the rendered attribute are EL language alternatives to the Java programming language conditional operators to allow XML parsing of the XHTML file.

TABLE 3–1 Conditional Operator EL Language Alternatives

Logical Condition	Java Programming Language Conditional Operator	EL Language Alternative
AND	&&	&&
EQUALS	==	==
LESS THAN	<	lt
GREATER THAN	>	gt

EXAMPLE 3–5 The response.xhtml File

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
<head>
    <title>Response Page</title>
</head>
<body>
<h:form>
    <h:outputText value="#{bundle.YouAre} "/>
    <h:outputText value="#{bundle.SameAge}"
                  rendered="#{DukesBDay.ageDiff == 0}"/>
    <h:outputText value="#{DukesBDay.absAgeDiff}"
                  rendered="#{DukesBDay.ageDiff lt 0}"/>
    <h:outputText value=" #{bundle.Year} "
                  rendered="#{DukesBDay.ageDiff == -1}"/>
    <h:outputText value=" #{bundle.Years} "
                  rendered="#{DukesBDay.ageDiff lt -1}"/>
    <h:outputText value="#{bundle.Younger}"
                  rendered="#{DukesBDay.ageDiff lt 0}"/>
    <h:outputText value="#{DukesBDay.absAgeDiff}"
                  rendered="#{DukesBDay.ageDiff gt 0}"/>
    <h:outputText value=" #{bundle.Year} "
                  rendered="#{DukesBDay.ageDiff == 1}"/>
    <h:outputText value=" #{bundle.Years} "
                  rendered="#{DukesBDay.ageDiff gt 1}"/>
    <h:outputText value="#{bundle.Older}" rendered="#{DukesBDay.ageDiff gt 0}"/>    <p/>
    <h:commandButton id="back" value="#{bundle.Back}" action="greeting"/>
```

EXAMPLE 3–5 The `response.xhtml` File *(Continued)*

```
</h:form>
</body>
</html>
```

For example, the `#{bundle.SameAge}` string is displayed if the user and Duke have the same birthday as specified by the condition `#{DukesBDay.ageDiff == 0}` in the `rendered` attribute. That is, display the following string if the `ageDiff` property of `DukesBDay` equals `0`:

```
You are the same age as Duke!
```

The form also contains a `<h:commandButton>` tag that creates a back button that will direct the user back to the `greeting.xhtml` page, as specified in the `action` attribute.

▼ Creating the XHTML Files

Create the Facelets XHTML files in NetBeans IDE.

1. **In the `firstcup` project select Web Pages in the left pane in NetBeans IDE.**
2. **Select File → New File.**
3. **Select Java Server Faces under Categories, New JSF Page under File Types, and click Next.**
4. **Enter `greeting` under File Name and click Finish.**
5. **Repeat the previous steps to create a new Facelets Simple File named `response`.**

▼ Set the Welcome File in the `web.xml` Deployment Descriptor

Configure the application to use `greeting.xhtml` as the welcome file by modifying `web.xml`

1. **In the `firstcup` project under Configuration Files double-click `web.xml`.**
2. **Click Pages.**
3. **Click Browse under Welcome Files, expand Web Pages, select `greeting.xhtml`, and click Select File.**
4. **Select File → Save.**

▼ Adding Tag Libraries to the XHTML Files

Modify `greeting.xhtml` to include the `components` tag library.

1. **In the `firstcup` project open `greeting.xhtml` by double-clicking the file name under Web Pages in the left pane.**

2. **Add the `components` tag library to the `<html>` tag.**

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:fc="http://java.sun.com/jsf/composite/components">
```

   The `components` resource library is referred to by the `fc` prefix. The JSF Core and HTML Render Kit tag libraries are also used in `greeting.xhtml`.

3. **Add a title directly after the `<html>` tag.**

```
<head>
    <title>Firstcup Greeting Page</title>
</head>
```

4. **Select File → Save.**

5. **Open `response.xhtml` and add the following title:**

```
<head>
    <title>Response Page</title>
</head>
```

6. **Select File → Save.**

▼ Adding the Form to `greeting.xhtml`

Add the form that provides the user interface for displaying Duke's age and entering the user's birthday.

1. **In the `firstcup` project in `greeting.xhtml` add the following tags between the `<body>` and `</body>` tags.**

```
<h:form>
    <h2>
        <h:outputText value="#{bundle.Welcome}"/>
    </h2>
    <h:outputText value="#{bundle.DukeIs} "/>
    <h:outputText value="#{DukesBDay.age} #{bundle.YearsOldToday}"/>
    <p/>
    <h:outputText value="#{bundle.Instructions}"/>
    <p/>
    <h:outputText value="#{bundle.YourBD} "/>
    <fc:inputDate id="userBirthday" date="#{DukesBDay.yourBD}" />
    <h:outputText value=" #{bundle.Pattern}"/>
    <p/>
```

```
    <h:commandButton value="#{bundle.Submit}" action="#{DukesBDay.processBirthday}"/>
    <p/>
    <h:message for="userBirthday" style="color:red"/>
</h:form>
```

2 **Right-click in the editor window and select Format.**

3 **Select File → Save.**

▼ Adding the Form to `response.html`

Add a form that displays the age difference between Duke and the user, displays the average age difference of all users, and allows the user to navigate back to `greeting.xhtml`.

1 **In the `firstcup` project in `response.xhtml` add the following tags between the `<body>` and `</body>` tags.**

```
<h:form>
    <h:outputText value="#{bundle.YouAre} "/>
    <h:outputText value="#{bundle.SameAge}"
                  rendered="#{DukesBDay.ageDiff == 0}"/>
    <h:outputText value="#{DukesBDay.absAgeDiff}"
                  rendered="#{DukesBDay.ageDiff lt 0}"/>
    <h:outputText value=" #{bundle.Year} "
                  rendered="#{DukesBDay.ageDiff == -1}"/>
    <h:outputText value=" #{bundle.Years} "
                  rendered="#{DukesBDay.ageDiff lt -1}"/>
    <h:outputText value="#{bundle.Younger}"
                  rendered="#{DukesBDay.ageDiff lt 0}"/>
    <h:outputText value="#{DukesBDay.absAgeDiff}"
                  rendered="#{DukesBDay.ageDiff gt 0}"/>
    <h:outputText value=" #{bundle.Year} "
                  rendered="#{DukesBDay.ageDiff == 1}"/>
    <h:outputText value=" #{bundle.Years} "
                  rendered="#{DukesBDay.ageDiff gt 1}"/>
    <h:outputText value="#{bundle.Older}" rendered="#{DukesBDay.ageDiff gt 0}"/>
    <p/>
    <h:outputText value="#{bundle.AverageAge}": #{DukesBday.averageAgeDifference}" />
    <p/>
    <h:commandButton id="back" value="#{bundle.Back}" action="greeting"/>
</h:form>
```

2 **Right-click in the editor window and select Format.**

3 **Select File → Save.**

Setting the Navigation for `firstcup`

Add a navigation rule to `faces-config.xml` that forwards the user to `response.xhtml` when the returned status of the `DukeBDay.processBirthday` method is `success`.

1. **With the `firstcup` project selected in the Projects pane, expand Configuration File and double-click `faces-config.xml`.**
2. **Click PageFlow in the top left corner of the editor pane to display the visual navigation editor.**
3. **Select `greeting.xhtml` click and hold the navigation arrow box on the far right, and drag a navigation arrow to the `response.xhtml` file.**
4. **Double-click the default `case1` outcome associated with the navigation arrow and change it to `success`.**
5. **Select File → Save.**

Building, Packaging, Deploying, and Running the `firstcup` Web Application

In this section, you will build the `firstcup` web application, deploy it to the server, and run the application.

Building and Packaging the `firstcup` Web Application

While performing this task, you'll build and package the `DukesBirthdayBean` enterprise bean and the `firstcup` web client into an WAR file, `firstcup.war`, in the `dist` directory.

1. **Select `firstcup` in the Projects tab.**
2. **Right-click `firstcup` and select Run.**

Running the `firstcup` Application

This section describes how to run the `firstcup` application.

1. **Launch an internet browser.**
2. **Enter the following URL in the address field of the browser:**

   ```
   http://localhost:8080/firstcup
   ```

3. **Enter your birth date in the Your birthday text field. Make sure you use the date pattern specified on the page: MM/dd/yyyy.**

4. **Click Submit.**
5. **After the** `response.xhtml` **page is displayed, click Back to return to the** `greeting.xhtml` **page.**

Example 3–6 A Successful Response Page for `firstcup`

```
You are 20 years older than Duke!
The average age difference of all First Cup users is: 20
```

◆ ◆ ◆ CHAPTER 4

Next Steps

This chapter points the user at additional resources for learning more about enterprise application architecture, the Java EE platform, and Oracle GlassFish Server.

The Java EE Tutorial

The Java EE Tutorial (`http://download.oracle.com/docs/cd/E17410_01/javaee/6/tutorial/doc/`) documents the technologies that make up the Java EE platform. The Java EE Tutorial describes each piece of the platform in detail, and includes code examples that demonstrate how to use each piece of the platform.

More Information on the Java EE Platform

For more information on the Java EE platform, see these resources:

- The Java EE Platform site (`http://java.sun.com/javaee`)
- GlassFish (`https://glassfish.dev.java.net`)
- The Aquarium (`http://blogs.sun.com/theaquarium`), a blog about GlassFish and open-source Java EE projects.

Java EE Servers

Java EE servers are application servers that implement the Java EE platform technologies.

Oracle GlassFish Server

GlassFish Server is the reference implementation of the Java EE platform APIs.

GlassFish

The GlassFish (`https://glassfish.dev.java.net`) project is the open-source basis for GlassFish Server.

Other Java EE Servers

There are several other popular Java EE servers used by developers and enterprises.

- Oracle's WebLogic Suite (`http://www.oracle.com/appserver/weblogic/weblogic-suite.html`).
- Red Hat's JBoss (`http://www.jboss.org`).
- Apache Geronimo (`http://geronimo.apache.org/`).
- IBM's WebSphere (`http://www.ibm.com/software/websphere`).