**Section 30.1 Introduction**

• Web-based apps create content for web browser clients. This content includes eXtensible HyperText Markup Language (XHTML), JavaScript client-side scripting, Cascading Style Sheets (CSS), images and binary data.

• XHTML is an XML (eXtensible Markup Language) vocabulary that is based on HTML (HyperText Markup Language).

• Java multitier applications are typically implemented using the features of Java Enterprise Edition (Java EE).

• The JavaServer Faces subset of Java EE is a web-application framework (p. 2) for building multitier web apps by extending the framework with your application-specific capabilities. The framework handles the details of receiving client requests and returning responses for you.

#### Section 30.2 HyperText Transfer Protocol (HTTP) Transactions

• In its simplest form, a web page is nothing more than an XHTML document that describes to a web browser how to display and format the document’s information.

• XHTML documents normally contain hyperlinks that link to different pages or to other parts of the same page. When the user clicks a hyperlink, the requested web page loads into the browser.

• Computers that run web-server software (p. 3) make resources available, such as web pages, images, PDF documents and even objects that perform complex tasks.

• The HTTP protocol allows clients and servers to interact and exchange information.

• HTTP uses URLs (Uniform Resource Locators) to locate resources on the Internet.

• A URL contains information that directs a browser to the resource that the user wishes to access.

• The computer that houses and maintains resources is usually referred to as the host (p. 3).

• Host names are translated into IP addresses by domain-name system (DNS) servers (p. 3).

• The path in a URL typically specifies a virtual directory on the server. The server translates the this into a real location, thus hiding a resource’s true location.

• When given a URL, a web browser performs an HTTP transaction to retrieve and display the web page at that address.

• HTTP headers (p. 5) provide additional information about the data that will be sent.

• Multipurpose Internet Mail Extensions (MIME; p. 5) is an Internet standard that specifies data formats so that programs can interpret data correctly.

• The two most common HTTP request types are GET and POST (p. 5). A GET request typically asks for a specific resource on a server. A POST request typically posts (or sends) data to a server.

• GET requests and POST requests can both be used to send form data to a web server, yet each request type sends the information differently. A GET request sends information to the server in the URL’s query string (p. 6). A POST request sends form data as part of the HTTP message.

• Browsers often cache (p. 6) web pages for quick reloading. If there are no changes between the cached version and the current version on the web, this speeds up your browsing experience.

• An HTTP response can indicate the length of time for which the content remains “fresh.” If this amount of time has not been reached, the browser can avoid another request to the server.

• Browsers typically do not cache the server’s response to a POSTrequest, because the next POST might not return the same result.

#### Section 30.3 Multitier Application Architecture

• Web-based applications are multitier (n-tier) applications (p. 6) that divide functionality into separate tiers (i.e., logical groupings of functionality). Although tiers can be located on the same computer, the tiers of web-based applications often reside on separate computers.

• The information tier (p. 7) maintains data pertaining to the application.

• The middle tier (p. 7) implements business logic, controller logic and presentation logic to control interactions between the application’s clients and the application’s data. Business logic in the middle tier enforces business rules and ensures that data is reliable before the server application updates the database or presents the data to users. Business rules dictate how clients can and cannot access application data, and how applications process data.

• The client tier (p. 7) is the application’s user interface, which gathers input and displays output. Users interact directly with the application through the user interface. In response to user actions (e.g., clicking a hyperlink), the client tier interacts with the middle tier to make requests and to retrieve data from the information tier.

#### Section 30.4 Your First JSF Web App

• The JSF web-application framework’s Faces servlet (p. 8) processes each requested JSF page so that the server can eventually return a response to the client.

#### Section 30.4.1 The Default *index.xhtml* Document: Introducing Facelets

• You present your web app’s content in JSF using Facelets (p. 9)—a combination of XHTML markup and JSF markup.

• XHTML (p. 9) specifies the content of a web page that is displayed in a web browser. XHTML separates the presentation of a document from the structure of the document’s data.

• Presentation is specified with Cascading Style Sheets (CSS).

• JSF uses the XHTML 1.0 Transitional Recommendation by default. Transitional markup may include some non-CSS formatting, but this is not recommended.

• The starting <html> tag may contain one or more xmlnsattributes (p. 9). Each has a name and a value separated by an equal sign (=), and specifies an XML namespace of elements that are used in the document.

• The attribute xmlns:h="http://java.sun.com/jsf/html"specifies a prefix and a URL for JSF’s HTML Tag Library (p. 10), allowing the document to use JSF’s elements from that library.

• A tag library defines a set of elements that can be inserted into the XHTML markup.

• The elements in the HTML Tag Library generate XHTML elements.

• The h:head element (p. 10) defines the XHTML page’s headelement.

• The document’s title typically appears in the browser window’s title bar, or a browser tab if you have multiple web pages open in the browser at once.

• The h:body (p. 10) element represents the page’s content.

• A JSF Expression Language (EL; p. 10) expression can interact with a JavaBean to obtain data.

#### Section 30.4.2 Examining the *WebTimeBean* Class

• JSF documents typically interact with one or more Java objects to perform the app’s tasks.

• JavaBeans objects (p. 10) are instances of classes that follow certain conventions for class design. A JavaBean exposes its data as properties (p. 10). Properties can be read/write, read-only or write-only. To define a read/write property, a JavaBean class provides set and get methods for that property. A read-only property would have only a get method and a write-only property only a set method.

• The JavaBeans used in JSF are also POJOs (plain old Java objects; p. 11)

• The JSF framework creates and manages objects of your JavaBean classes for you.

• The @ManagedBean annotation (from the packagejavax.faces.bean; p. 11) indicates that the JSF framework should create and manage instances of the class. The parentheses following the annotation contain the optional nameattribute. If you specify the annotation without the parentheses and the name attribute, the JSF framework will use the class name with a lowercase first letter as the default bean name.

• When the Faces servlet encounters an EL expression that accesses a bean property, it automatically invokes the property’sset or get method based on the context in which the property is used.

#### Section 30.5 Model-View-Controller Architecture of JSF Apps

• JSF applications adhere to the Model-View-Controller (MVC; p. 16) architecture, which separates an application’s data (contained in the model) from the graphical presentation (the view) and the processing logic (the controller).

• In JSF, the controller is the JSF framework and is responsible for coordinating interactions between the view and the model. The model contains the application’s data (typically in a database), and the view presents the data stored in the model (typically as web pages).

#### Section 30.6 Common JSF Components

• Elements from the JSF HTML Tag Library are mapped by the JSF framework to a combination of XHTML elements and JavaScript code that enables the browser to render the page.

• The h:form element (p. 17) contains the components with which a user interacts to provide data, such as registration or login information, to a JSF app.

• An h:panelGrid element (p. 19) organizes elements in an XHTML table. The columns attribute specifies the number of columns in the table. The style attribute specifies the CSS formatting for the table.

• A h:graphicImage (p. 19) displays an image (specified by thename attribute) in the page.

• As of JSF 2.0, you add resources (p. 19) that are used throughout your app—such as images, CSS files, JavaScript files—to your web apps by placing them in the app’s resources folder within your project’s **Web Pages** node. Each subfolder of resources represents a resource library (p. 19).

• An empty element (p. 20) does not have content between its start and end tags. In such an element, data can be specified as attributes in the start tag. You can close an empty element either by placing a slash immediately preceding the start tag’s right angle bracket or by explicitly writing an end tag.

• An h:selectOneMenu element (p. 20) is typically rendered in a web page as a drop-down list. Each item to display appears between the start and end tags of this element as anf:selectItem element (from the JSF Core Tag Library; p. 20). Anf:selectItem’s itemLabel is the string that the user will see in the browser, and its itemValue is the value that’s returned when you programmatically retrieve the user’s selection from the drop-down list.

• An h:outputLink element (p. 20) inserts a hyperlink in a web page. Its value attribute specifies the resource that’s requested when a user clicks the hyperlink.

• An h:selectOneRadio element (p. 20) provides a series of radio buttons from which the user can select only one.

• An h:commandButton element (p. 21) triggers an action when clicked. An h:commandButton typically maps to an XHTML inputelement with its type attribute set to "submit". Such elements are often used to submit a form’s user input values to the server for processing.

#### Section 30.7 Validation Using JSF Standard Validators

• Form validation (p. 21) helps prevent processing errors due to incomplete or improperly formatted user input.

• An f:validateLength validator (p. 21) determines whether a field contains an acceptable number of characters.

• f:validateDoubleRange and f:validateLongRange validators (p. 21) determine whether numeric input falls within acceptable ranges.

• An f:validateRequired validator (p. 21) determines whether a field contains a value.

• An f:validateRegex validator (p. 21) determines whether a field contains a string that matches a specified regular expression pattern.

• An f:validateBean validator (p. 21) invokes a bean method that performs custom validation.

• An h:outputText element (p. 26) inserts text in a page.

• An input element’s required attribute (when set to "true"; p. 26) ensures that the user has made a selection or entered some text in a required input element is a basic type of validation.

• An input element’s requiredMessage attribute (p. 26) specifies the message that should be displayed if the user submits the form without first providing a value for the required element.

• An input element’s validatorMessage attribute (p. 26) specifies the message to display if a validator is associated with the element and the data the user enters is invalid.

• The messages specified by the requiredMessage andvalidatorMessage attributes are displayed in an associatedh:message element (p. 26) when validation fails.

• To limit the length of user input in an h:inputText, set itsmaxlength attribute (p. 27)—the element’s cursor will not advance beyond the maximum allowable number of characters.

• In a postback (p. 27), the browser re-requests the page and sends the values of the form’s fields to the server for processing.

#### Section 30.8 Session Tracking

• Personalization (p. 28) makes it possible for e-businesses to communicate effectively with their customers and also improves the user’s ability to locate desired products and services.

• A trade-off exists between personalized e-business service and protection of privacy. Some consumers embrace the idea of tailored content, but others fear the possible adverse consequences if the information they provide to e-businesses is released or collected by tracking technologies.

• HTTP is a stateless protocol—it does not provide information that would enable web servers to maintain state information regarding particular clients.

• To help the server distinguish among clients, each client must identify itself to the server. Tracking individual clients, known as session tracking, can be achieved in a number of ways. One popular technique uses cookies; another uses the @SessionScopedannotation.

#### Section 30.8.1 Cookies

• A cookie (p. 29) is a piece of data stored in a small text file on the user’s computer. A cookie maintains information about the client during and between browser sessions.

• The expiration date (p. 29) of a cookie determines how long the cookie remains on the client’s computer. If you do not set an expiration date for a cookie, the web browser maintains the cookie for the duration of the browsing session.

#### Section 30.8.2 Session Tracking with *@SessionScoped* Beans

• A @SessionScoped bean (p. 30) can maintain a user’s selections throughout the user’s session. Such a bean is created when a session begins and exists throughout the entire session.

• A @SessionScoped bean can be accessed by all of the app’s pages, and the app server maintains a separate @SessionScoped bean for each user.

• By default a session expires after 30 minutes of inactivity or when the user closes the browser that was used to begin the session. When the session expires, the server discards the bean that was associated with that session.

• The ui:repeat element (from the JSF Facelets Tag Library; p. 35) inserts its nested element(s) once for each element in a collection. The collection is specified by the value attribute’s EL expression, which must return an array, a List, a java.sql.ResultSet or anObject.

• The ui:repeat element’s var attribute creates a variable named book to which each item in the collection is assigned in sequence.

### Self-Review Exercises

[**30.1**](http://proquest.safaribooksonline.com/9780133813036/ch30lev1sec12_html#ch30ans01)State whether each of the following is true or false. If false, explain why.

a) A URL contains information that directs a browser to the resource that the user wishes to access.

b) Host names are translated into IP addresses by web servers.

c) The path in a URL typically specifies a resource’s exact location on the server.

d) GET requests and POST requests can both be used to send form data to a web server.

e) Browsers typically cache the server’s response to a POST request.

f) A tag library defines a set of elements that can be inserted into the XHTML markup.

g) You must create and manage the JavaBean objects that are used in your JSF web applications.

h) When the Faces servlet encounters an EL expression that accesses a bean property, it automatically invokes the property’sset or get method based on the context in which the property is used.

i) An h:panelGrid element organizes elements in an XHTML table.

j) An h:selectOneMenu element is typically rendered in a web page as a set of radio buttons.

k) The messages specified by an element’s requiredMessage andvalidatorMessage attributes are displayed in an associatedh:message element when validation fails.

l) The HTTP protocol provides information that enables web servers to maintain state information regarding particular clients.

m) The ui:repeat element inserts its nested element(s) once for each element in a collection. The collection can be anyIEnumerable type.

[**30.2**](http://proquest.safaribooksonline.com/9780133813036/ch30lev1sec12_html#ch30ans02)Fill in the blanks in each of the following statements:

a) Java multitier applications are typically implemented using the features of \_\_\_\_\_\_\_\_\_.

b) Computers that run \_\_\_\_\_\_\_\_\_ software make resources available, such as web pages, images, PDF documents and even objects that perform complex tasks.

c) The JSF web-application framework’s \_\_\_\_\_\_\_\_\_ processes each requested JSF page.

d) A(n) \_\_\_\_\_\_\_\_\_ exposes its data as read/write, read-only or write-only properties.

e) The \_\_\_\_\_\_\_\_\_ annotation indicates that the JSF framework should create and manage instances of the class.

f) A(n) \_\_\_\_\_\_\_\_\_ element contains the components with which a user interacts to provide data, such as registration or login information, to a JSF app.

g) A(n) \_\_\_\_\_\_\_\_\_ element triggers an action when clicked.

h) A(n) \_\_\_\_\_\_\_\_\_ validator determines whether a field contains an acceptable number of characters.

i) A(n) \_\_\_\_\_\_\_\_\_ validator determines whether a field contains a string that matches a specified regular expression pattern.

j) In a(n) \_\_\_\_\_\_\_\_\_, the browser re-requests the page and sends the values of the form’s fields to the server for processing.

k) A(n) \_\_\_\_\_\_\_\_\_ bean is created when a session begins and exists throughout the entire session.

### Answers to Self-Review Exercises

[**30.1**](http://proquest.safaribooksonline.com/9780133813036/ch30lev1sec11_html#ch30que01) a) True. b) False. Host names are translated into IP addresses by DNS servers. c) False. The server translates a virtual directory into a real location, thus hiding a resource’s true location. d) True. e) False. Browsers typically do not cache the server’s response to a POST request, because the next POST might not return the same result. f) True. g) False. The JSF framework creates and manages objects of your JavaBean classes for you. h) True. i) True. j) False. An h:selectOneRadio element is rendered as a set of radio buttons. An h:selectOneMenu is rendered as a drop-down list. k) True. l) False. HTTP is a stateless protocol that does not provide information that enables web servers to maintain state information regarding particular clients—a separate tracking technology must be used. m) False. A ui:repeat element can iterate over only arrays, Lists andResultSets. For any other object, the elements in a ui:repeat element will be inserted once.

[**30.2**](http://proquest.safaribooksonline.com/9780133813036/ch30lev1sec11_html#ch30que02) a) Java Enterprise Edition (Java EE). b) web-server. c) Faces servlet. d) JavaBean. e) @ManagedBean. f) h:form. g) h:commandButton. h)f:validateLength. i) f:validateRegex. j) postback. k) @SessionScoped.

### Exercises

**30.3** ***(Registration Form Modification)*** Modify the WebComponentsapplication to add functionality to the **Register** button. When the user clicks **Register**, validate all input fields to make sure the user has filled out the form completely and entered a valid email address and phone number. Then, display a message indicating successful registration and show the user’s registration information at the bottom of the page. (This is similar to the example in Section 30.7.) You’ll need to create an appropriate bean class to store the user’s registration information.

**30.4** ***(Shopping Cart Application)*** Using the techniques you learned in Section 30.8.2, create a simple shopping cart application. Display a list of books as an h:selectOneRadio element. When the user submits the form, store the user’s selection in a @SessionScoped managed bean. Allow the user to return to the list of books and make additional selections. Provide a link to view the shopping cart. On the shopping cart page, display the list of selections the user made, the price of each book and the total of all books in the cart.

**30.5** **(Guestbook** **Application)** In Section 30.8.2, you used an@SessionScoped managed bean to maintain an individual user’s selections. JSF also provides the **@ApplicationScoped** **annotation** for managed beans that should be shared among all users of a JSF app. For this exercise, create an @ApplicationScoped GuestbookBean that maintains a List of GuestbookEntry objects. The application should provide a form, similar to the one in [Fig. 30.15](http://proquest.safaribooksonline.com/9780133813036/ch30lev2sec19_html#ch30fig15) that enables a user to enter a name, an email address and a message. When the user submits the form, the GuestbookBean should create a GuestbookEntryobject containing the submitted values and insert it at the beginning of the List of GuestbookEntry objects—this places the most recent entry first. The GuestbookBean should also provide a read-only property that returns the List of GuestbookEntry objects. The page should use aui:repeat element to display all of the items in the List so the user can see all the guestbook entries so far. [Note: In the next chapter, you’ll implement a similar exercise that stores this information in a database.]