Containers, Filters and Sessions

Content

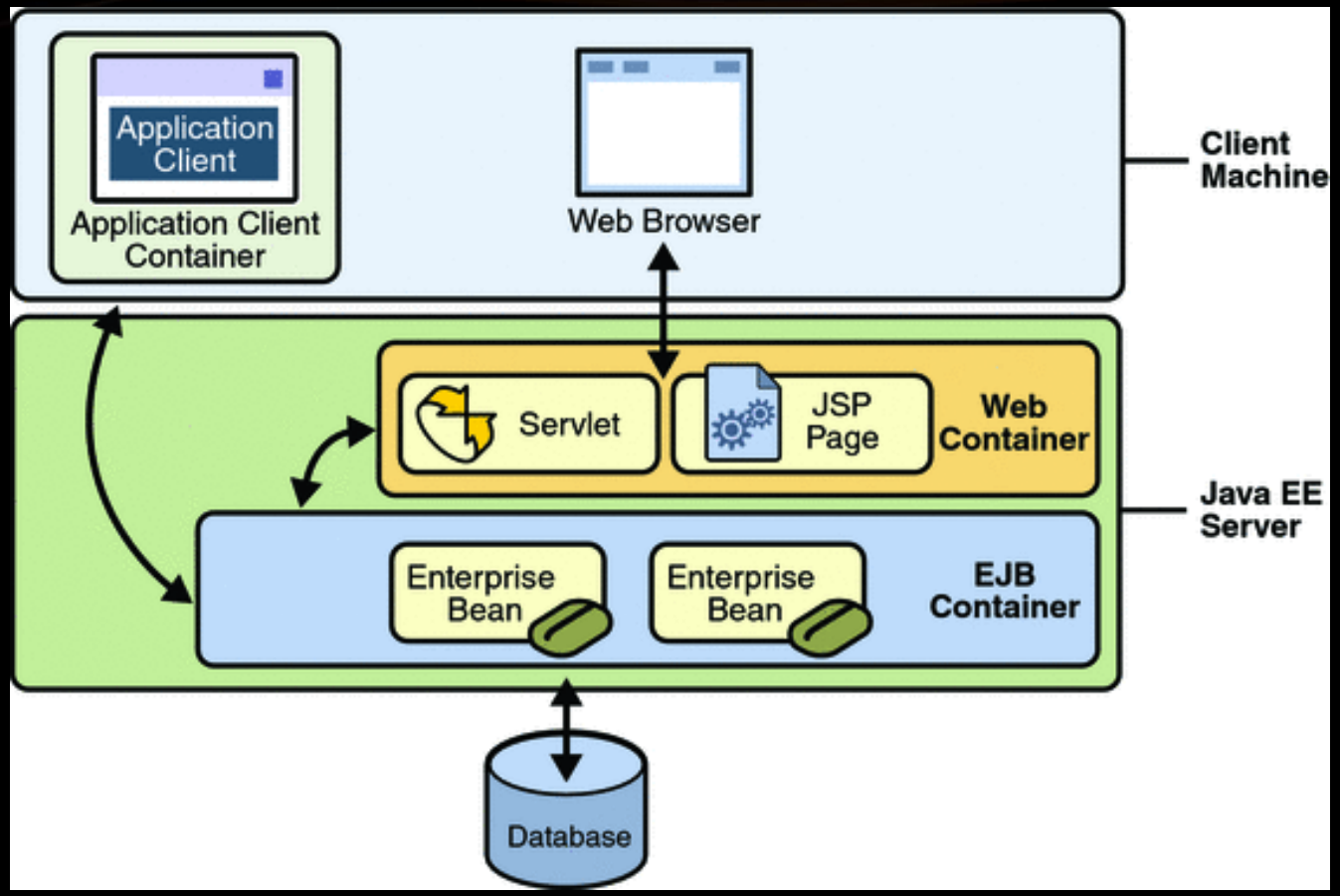
- Containers
- Sessions
- Listeners
- Filters

Containers

Java EE Containers

- Containers are the interface between a component and the low-level platform-specific functionality that supports the component.
- Before a web, enterprise bean, or application client component can be executed, it must be assembled into a Java EE module and deployed into its container.
- The container wraps each new component and presents numbers of services:
  - Life cycle control
  - Security
  - Transactions
  - Inversion of control (IoC)
  - Etc.

Container Types



Container Types(2)

- Java EE server: The runtime portion of a Java EE product. A Java EE server provides EJB and web containers.
- Enterprise JavaBeans (EJB) container: Manages the execution of enterprise beans for Java EE applications. Enterprise beans and their container run on the Java EE server.
- Web container: Manages the execution of JSP page and servlet components for Java EE applications. Web components and their container run on the Java EE server.
- Application client container: Manages the execution of application client components. Application clients and their container run on the client.
- Applet container: Manages the execution of applets. Consists of a web browser and Java Plug-in running on the client together.

Sessions

Maintaining Client State

- Many applications require that a series of requests from a client be associated with one another.
  - For example, a web application can save the state of a user's shopping cart across requests.
- Web-based applications are responsible for maintaining such state, called a session, because HTTP is stateless.
- To support applications that need to maintain state, Java Servlet technology provides an API for managing sessions and allows several mechanisms for implementing sessions.

HTTP Session

- Sessions are represented by an HttpSession object.
- You access a session by calling the getSession method of a request object. This method returns the current session associated with this request or, if the request does not have a session, this method creates one.
- You can associate object-valued attributes with a session by name. Such attributes are accessible by any web component that belongs to the same web context and is handling a request that is part of the same session.

Session Management

- Because an HTTP client has no way to signal that it no longer needs a session, each session has an associated timeout so that its resources can be reclaimed.
- The timeout period can be accessed by using a session's getMaxInactiveInterval and setMaxInactiveInterval methods.
- To ensure that an active session is not timed out, you should periodically access the session by using service methods because this resets the session's time-to-live counter.
- When a particular client interaction is finished, you use the session's invalidate method to invalidate a session on the server side and remove any session data.

Session Tracking

- To associate a session with a user, a web container can use several methods, all of which involve passing an identifier between the client and the server.
- The identifier can be maintained on the client as a cookie, or the web component can include the identifier in every URL that is returned to the client.
- If your application uses session objects, you must ensure that session tracking is enabled by having the application rewrite URLs whenever the client turns off cookies.
  - You do this by calling the response's encodeURL (URL) method on all URLs returned by a servlet.
  - This method includes the session ID in the URL only if cookies are disabled – otherwise, the method returns the URL unchanged.

Cookies

- Cookies are data, stored in small text files, on your computer.
- When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.
- Cookies were invented to solve the problem "how to remember information about the user":
  - When a user visits a web page, his name can be stored in a cookie.
  - Next time the user visits the page, the cookie "remembers" his name.
- Cookies are saved in name-value pairs like:
  - username=John Doe
- When a browser request a web page from a server, cookies belonging to the page is added to the request. This way the server gets the necessary data to "remember" information about users.

Sessions

Demo

MusalaSoft

SOFTWARE UNIVERSITY FOUNDATION

Listeners

Servlet Lifecycle

- The lifecycle of a servlet is controlled by the container in which the servlet has been deployed. When a request is mapped to a servlet, the container performs the following steps.
  - If an instance of the servlet does not exist, the web container
    - Loads the servlet class.
    - Creates an instance of the servlet class.
    - Initializes the servlet instance by calling the init method.
  - Invokes the service method, passing request and response objects.
- If it needs to remove the servlet, the container finalizes the servlet by calling the servlet's destroy method.

Handling Servlet Lifecycle Events

- You can monitor and react to events in a servlet's lifecycle by defining listener objects whose methods get invoked when lifecycle events occur.
- To use these listener objects, you must define and specify the listener class.
  - Listener class is defined as an implementation of a listener interface.
  - When a listener method is invoked, it is passed an event that contains information appropriate to the event.
  - Example – the methods in the HttpSessionListener interface are passed an HttpSessionEvent, which contains an HttpSession.

Handling Servlet Lifecycle Events

Object	Event	Listener Interface and Event Class
Web context	Initialization and destruction	ServletContextListener and ServletContextEvent
Web context	Attribute added, removed, or replaced	ServletContextAttributeListener and ServletContextAttributeEvent
Session	Creation, invalidation, activation, passivation, and timeout	HttpSessionListener, HttpSessionActivationListener, and HttpSessionEvent
Session	Attribute added, removed, or replaced	HttpSessionAttributeListener and HttpSessionBindingEvent
Request	A servlet request has started being processed by web components	ServletRequestListener and ServletRequestEvent
Request	Attribute added, removed, or replaced	ServletRequestAttributeListener and ServletRequestAttributeEvent

Using Listeners

- Listener class is defined as an implementation of a listener interface.
- The @WebListener annotation is used to declare a class as Listener, however the class should implement one or more of the Listener interfaces.
- Listener could be defined in web.xml as well

- Here's a typical XML fragment from web.xml:

```
<listener>
   <listener-class>bg.jwd.listeners.SessionListenerExample</listener-class>
</listener>
```

Listeners

Demo

MusalaSoft

SOFTWARE UNIVERSITY FOUNDATION

Filters

Filtering Requests and Responses

- A filter is an object that can transform the header and content (or both) of a request or response.
- Filters differ from web components in that filters usually do not themselves create a response.
- A filter provides functionality that can be "attached" to any kind of web resource.
- Consequently, a filter should not have any dependencies on a web resource for which it is acting as a filter – this way, it can be composed with more than one type of web resource.

Filter applications

- The main tasks that a filter can perform are as follows:
  - Query the request and act accordingly.
  - Block the request-and-response pair from passing any further.
  - Modify the request headers and data. You do this by providing a customized version of the request.
  - Modify the response headers and data. You do this by providing a customized version of the response.
  - Interact with external resources.
- Applications of filters include authentication, logging, image conversion, data compression, encryption, tokenizing streams, XML transformations, and so on.
- You can configure a web resource to be filtered by a chain of zero, one, or more filters in a specific order.

Programming filters

- The filtering API is defined by the Filter, FilterChain, and FilterConfig interfaces in the javax.servlet package. You define a filter by implementing the Filter interface.
- Use the @WebFilter annotation to define a filter in a web application. This annotation is specified on a class and contains metadata about the filter being declared.
- The annotated filter must specify at least one URL pattern. This is done by using the urlPatterns or value attribute on the annotation. All other attributes are optional, with default settings.
- Classes annotated with the @WebFilter annotation must implement the javax.servlet.Filter interface.

Programming filters(2)

- The most important method in the Filter interface is doFilter, which is passed request, response, and filter chain objects.
- This method can perform the following actions:
  - Examine the request headers.
  - Customize the request object if the filter wishes to modify request headers or data.
  - Customize the response object if the filter wishes to modify response headers or data.
  - Invoke the next entity in the filter chain.
  - Examine response headers after invoking the next filter in the chain.
  - Throw an exception to indicate an error in processing.

Programming Customized Requests and Responses

- There are many ways for a filter to modify a request or a response. For example, a filter can add an attribute to the request or can insert data in the response.
- A filter that modifies a response must usually capture the response before it is returned to the client.
- To override request methods, you wrap the request in an object that extends either ServletRequestWrapper or HttpServletRequestWrapper.
- To override response methods, you wrap the response in an object that extends either ServletResponseWrapper or HttpServletResponseWrapper.

Using filters

- To use the filter, you have to deploy it and configure your deployment descriptor, the web.xml file (alternative to the annotation).
- The required steps are:
  - Declare the filter.
  - Map the filter.
- The filter declaration is a typical declaration:
  - You define a name for your filter class.
  - The filter mapping defines when the filter execution is triggered.
  - You can specify an URL pattern and an optional dispatcher type.

Using filters(2)

- Here's a typical XML fragment from web.xml:

```
<filter>
   <filter-name>LogFilter</filter-name>
   <filter-class>bg.jdw.filters.LogFilter</filter-class>
   <init-param>
      <param-name>test-param</param-name>
      <param-value>This parameter is for testing.</param-value>
   </init-param>
</filter>

<filter-mapping>
   <filter-name>LogFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

Filters

Demo