FPT UNIVERSITY
DREAM OF INNOVATION

Servlet Session Management

FPT UNIVERSITY
DREAM OF INNOVATION

Objectives

- Maintaining state using session

+ Understand Web container model and Session

+ Store and Retrieve data to session

Web Container Model

- The servlet container is a compiled, executable program. The container is the intermediary between the Web server and the servlets in the container.
- The container loads, initializes, and executes the servlets. When a request arrives, the container maps the request to a servlet, translates the request, and then passes the request to the servlet. The servlet processes the request and produces a response. The container translates the response into the network format, then sends the response back to the Web server.

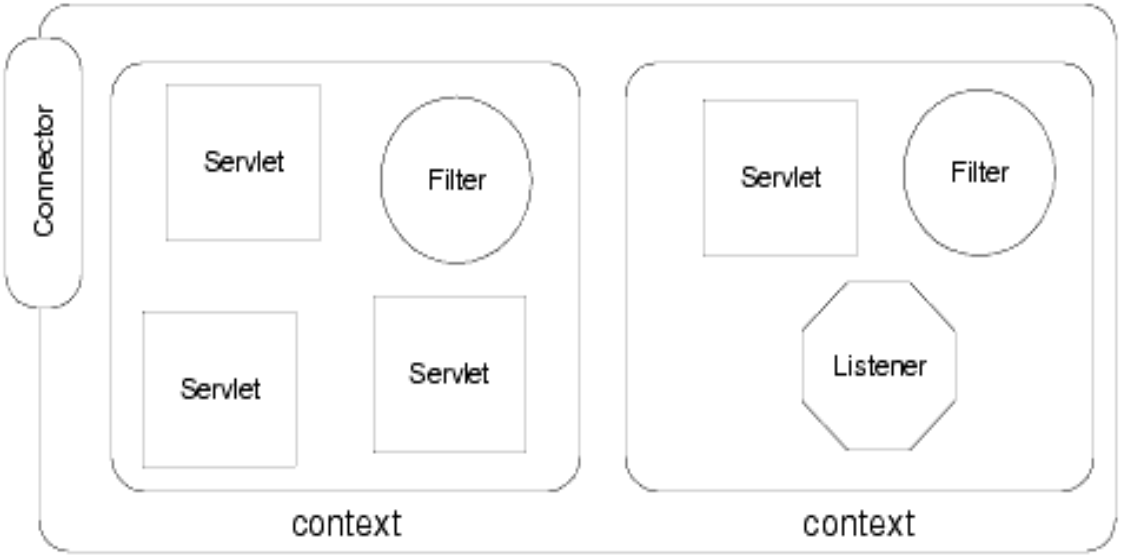
Web Container Model…

- The container is designed to perform well while serving large numbers of requests.
- A container can hold any number of active servlets, filters, and listeners.
- Both the container and the objects in the container are multithreaded. The container creates and manages threads as necessary to handle incoming requests. The container handles multiple requests concurrently, and more than one thread may enter an object at a time. Therefore, each object within a container must be threadsafe.

Web Container Model…

- Fortunately, you are a web *component* developer, not a web *container* developer. So you can take for granted much of what is built into the web container. You are a consumer of what the web container provides—and have to understand the infrastructure only insofar as it affects your own business applications.

ServletContext



ServletContext…

- A servlet container can manage any number of distinct applications. An application consists of any number of servlets, filters, listeners, and static Web pages. A set of components working together is a Web application.
- The container uses a *context* to group related components. The container loads the objects within a context as a group, and objects within the same context can easily share data.
- Each context usually corresponds to a distinct Web application.

ServletContext…

- For example, the directory structure below describes two contexts, one named orders and one named catalog. The catalog context contains a static HTML page, intro.html.

webapps

\orders

\WEB-INF

web.xml

\catalog

intro.html

\WEB-INF

web.xml

ServletContext Initialization Parameters

- What if you want some fundamental information available to all the dynamic resources (servlets, JSPs) within your web application?

ServletContext Initialization Parameters…

- **Setting Up the Deployment Descriptor (web.xml)**

```
<web-app>
    <context-param>
            <param-name>machineName</param-name>
            <param-value>GERALDINE</param-value
    </context-param>
    <context-param>
            <param-name>secretParameterFile</param-name>
            <param-value>/WEB-INF/xml/secretParms.xml</param-value>
    </context-param>
</web-app>
```

ServletContext.getInitParameter()

ServletContext Initialization Parameters…

- **Writing Code to Retrieve ServletContext Initialization Parameters**

```
ServletContext sc = getServletContext();
String database = sc.getInitParameter("machineName");
String secret = sc.getInitParameter("secretParameterFile");
```

ServletConfig Initialization Parameters

- **Setting Up the Deployment Descriptor (web.xml)**

```
<servlet>
    <servlet-name>InitParamaters</servlet-name>
    <servlet-class>InitParamaters</servlet-class>
      <init-param>
        <param-name>name</param-name>
        <param-value>FPT University</param-value>
      </init-param>
</servlet>
```

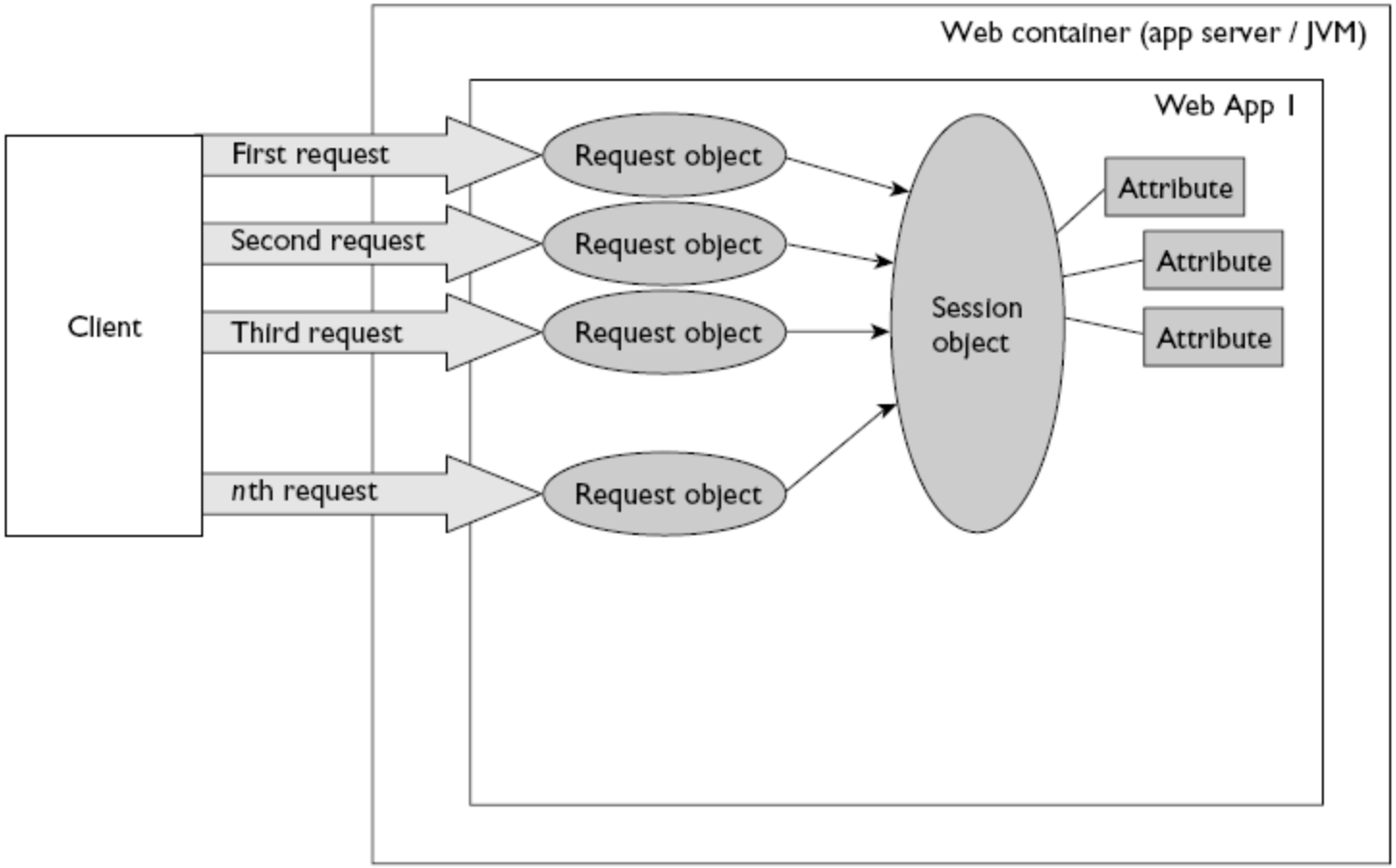
ServletConfig.getInitParameter()

ServletConfig Initialization Parameters…

- **Writing Code to Retrieve ServletConfig Initialization Parameters**

```
ServletConfig sc = getServletConfig();
String name = sc.getInitParameter(“name");
```

Session Life Cycle



Session Death

- HttpSession.invalidate() : Invalidates this session then unbinds any objects bound to it.
- Time-out : The time-out value is controlled in one of three ways:
  - **Application Server Global Default**
  - **Web Application Default**

```
<web-app>
   <session-config>
      <session-timeout>60</session-timeout>
   </session-config>
</web-app>
```

  - **Individual Session Setting**
    - On obtaining a session, your servlet code can set the time-out value for that individual session only using the **HttpSession.setMaxInactiveInterval(int seconds)** API.
    - An interval value of **zero or less** indicates that the session should never timeout.

Session Management

- There are two principal methods for session management “officially” recognized by the servlet API. One method is management by cookie exchange, and the other is management by rewriting URLs.

Session Management: General Principles

- Each of these requests needs to carry a unique ID, which identifies the session to which it belongs.
- The web application will allocate this unique ID on the first request from the client.
- The ID must be passed back to the client so that the client can pass it back again with its next request. In this way, the web application will know to which session the request belongs. This implies that the client must need to store the unique ID somewhere—and that's where session management mechanisms come in.

Cookies

- The preferred way for a web application to pass session IDs back to a client is via a cookie in the response. This is a small text file that the client stores somewhere. Storage can be on disk or memory.
- In the case of J2EE web applications, the cookie returned has a standard name **JSESSIONID** and store in memory.

URL Rewriting

- A “pseudo-parameter” called *jsessionid* is placed in the URL between the servlet name (with path information, if present) and the query string.

Example:

http://localhost:8080/examp0401/SessionExample;jsessionid=58112645388D9380808A726A27F92997?name=value

FPT UNIVERSITY
DREAM OF INNOVATION

Summary

- **Web Container Model**
  - **ServletContext**
- **Sessions and Listeners**
  - **Session Life Cycle**
  - **Session Management**