What is a Session?

- A session is defined as a sequence of client/server interactions initiated by a single client for the duration of some work unit.
- The term session is protocol specific and has different meanings/demarcations for different levels in the protocol stack (i.e. the OSI stack)
  - A TCP/IP session refers to the sequence of TCP/IP messages demarcated by the point of time when a TCP connection is open and when it’s closed.
  - An HTTP session refers to a sequence of HTTP request/response messages with the same session identifier.
  - An application session refers to a sequence of request/response messages demarcated by a user login and subsequent logout or server timeout.
February 20, 2008
Nick Duan

The OSI Protocol Stack

OSI Layers

Application Layer (HTTP & SOAP)
Presentation Layer (Socket APIs)
Session Layer (SSL)
Transport Layer (TCP)
Network Layer (IP)
Data Link Layer (Ethernet & Tw)
Physical Layer (Cable & Fiber, etc.)

Web Application

HTTP

Relationship among Sessions

- An HTTP session is composed of one or more TCP/IP sessions.
- A TCP/IP session usually consists of a single pair of HTTP request and response message.
- A Web application session is usually equivalent to an HTTP session.
Making HTTP Stateful

- To enable meaningful client/server conversations, the server has to maintain state information of the application (e.g. to determine when a session starts and ends)
- HTTP is stateless by nature. Additional conventions to be introduced to the protocol to make it stateful
  - Using a session ID to allow server keeping track of the conversation
  - A session ID is initiated by the server and kept by both client and server.
  - The same session ID is to be embedded in every request and response messages for the same session
- There are three basic methods to embed session ID in HTTP messages
  - Cookies, URL Rewriting and Hidden Input Fields

Enable HTTP Session via Cookies

- This is the most common way of doing session tracking
  1. Server to create a session and generate a unique session ID
     Set-Cookie: jsessionid=AAD870C41CF0DC96F405A1CFDA152E37
  2. Client to cache the ID and set it back with subsequent request messages
     Cookie: jsessionid=AAD870C41CF0DC96F405A1CFDA152E37
  3. Server and client use the same session ID for all subsequent message exchange
Enable HTTP session via URL Rewriting

- The URL string is being extended to include session information. Both client and the server are using the same session ID for all conversations of a single session
  http://ise.gmu.edu/~nduan;jsessionid=12345
- Widely used when browser is not set to accept cookies
- The length of the session ID is restricted by the GET request messages
- Session ID is visible on the browser

Enable HTTP session via Hidden Fields

- Session ID is defined in a hidden input field.
  `<input type=hidden name="jsessionid" value="12345">`
- Both client and server are sending the same hidden input field value back and forth for all conversations of the same session
- Useful only for HTML-based web applications
  - Application-specific coding required. No direct server support
- Session information visible on the browser in view source
HTTP Sessions in J2EE

- By default, the J2EE server uses HTTP Cookies to enable session tracking
  - A client makes the initial request to the server
  - Server creates a session object and sets the session id as a cookie with the name JSESSIONID in the response message
  - The client caches the session id and embeds it in subsequent request message
- If a browser does not support the use of cookies, the server automatically uses URL rewriting to perform session tracking
  - http://www.gmu.edu/sampleapps;jsessionid=12345
- The session ID may contain additional server-specific info

The HttpSession Object

- Created by the web container for all user sessions regardless of security requirement
- The session object is used to associate all related servlet objects
The HttpSession API

- `String getId()`
- `Object getAttribute(String name)`
- `void setAttribute(String name, Object value)`
- `void removeAttribute(String name)`
- `Enumeration getAttributeNames()`
- `boolean isNew()`
- `int getMaxInactiveInterval()`
- `void setMaxInactiveInterval(int time)`
- `ServletContext getServletContext()`

Attributes are name/value pairs stored in the session object

More about ServletContext object later

Usage Sample of HttpSession

```java
HttpSession session = request.getSession(true); // create a new session if the request does not have an existing one
Date created = new Date(session.getCreationTime());
Date accessed = new Date(session.getLastAccessedTime());
out.println("ID " + session.getId());
out.println("Created: " + created);
out.println("Last Accessed: " + accessed);
// set session info if needed
String dataName = request.getParameter("dataName");
if (dataName != null && dataName.length() > 0) {
    String dataValue = request.getParameter("dataValue");
    session.setAttribute(dataName, dataValue);
}
```
How to Configure Session Timeout

- The J2EE server defines the default session timeout setting in its configuration file for all web applications
  - vendor-specific, not a standard. In Tomcat, it is defined in server.xml file as part of the connector attributes
- Each web application may define its own session timeout parameter, overriding the one defined by the server
  - J2EE standard, defined in web.xml file
    - `<session-config>
      <session-timeout>1</session-timeout>
      </session-config>

Accessing Session Info in JSP

- Define a session attribute object in the page
  ```jsp
  <%@ page import = "num.NumberGuessBean" %>
  <jsp:useBean id="numguess" class="num.NumberGuessBean"
    scope="session"/>
  <jsp:setProperty name="numguess" property="*"/>
  ```
- Access the session attribute object
  - Try `<b>Try <%= numguess.getHint() %></b>`.
  - You have made `<%= numguess.getNumGuesses() %>` guesses.<p>
- Access the HttpSession object via the implicit reference `<i>session</i>`
  ```jsp
  <%=session.getId() %>
  ```
Data Sharing in Servlets & JSPs

- Servlets can share data at different scope levels. Data sharing is achieved via the get/setAttribute methods of the corresponding scope object
  - Request (HttpServletRequest object)
    - Data shared among servlets/JSPs in a single servlet chain
  - Session (HttpSession object)
    - Data shared among servlets/JSPs in a single HTTP session
  - Application (ServletContext object)
    - Data shared among servlets/JSPs in a single web app
  - Page (pageContext object, JSP only)
    - Data shared on a single JSP

The ServletContext Object

- An object created by the container accessible by all servlets/JSPs in a web application (defined in a single war file)
- Its parameters are defined in the deployment descriptor (web.xml)
  ```xml
  <context-param>
    <param-name>Webmaster</param-name>
    <param-value>webmaster@mycorp.com</param-value>
  </context-param>
  ```
- Accessing the context parameters via ServletConfig or ServletContext API
Sample Usage of ServletContext

- Getting the initialization parameters
  ```java
  public class SampleServlet extends HttpServlet {
      private String webmaster;
      public void init(ServletConfig config) {
          webmaster = config.getInitParameter("Webmaster");
          // or use: ServletContext ctx = config.getServletContext();
          //         webmaster = ctx.getInitParameter("Webmaster");
      } ....
  }
  ```

- Setting/Getting attributes
  ```java
  ServletContext ctx = this.getServletContext();
  ctx.setAttribute("name", myname);
  ....
  String name = ctx.getAttribute("name");
  ```

Online References

- Sun's J2EE Tutorial
  ```
  Sun's J2EE Tutorial
  ```

- Servlet and JSP specifications
  ```
  Servlet http://jcp.org/en/jsr/detail?id=154 (Chapter 7, Session)
  JSPs http://jcp.org/en/js/detail?id=245 (Chapter 2.8, Object)
  ```

- RFC 2965, HTTP State Management Mechanism
  ```
  RFC 2965, HTTP State Management Mechanism
  http://www.ietf.org/rfc/rfc2965.txt
  ```
Summary

- Session is an important concept in defining client/server interactions in a networked environment. It has different meanings for different layers in the protocol stack.
- An HTTP session is defined at the application layer and consists of one or more TCP/IP sessions, and it is defined with a unique session ID for all HTTP request/response messages of that session.
- There are three ways to embed session ID in HTTP messages: cookies, URL rewriting, and hidden input field in HTML form.
- HttpSession object is used to track all servlets involved in a HTTP session.
- Four types of scopes are defined in web container for inter-servlet data sharing: page, request, session and application.

Quiz

- Could multiple servlet objects be invoked to handle a single TCP/IP session (single pair of HTTP request/response message)?
- What is the method in HttpSession for creating a new session object if none exists?
- Are the methods in HttpSession thread-safe? If not, do I have to be concerned with synchronization when setting session attributes?
- Are the methods in ServletContext thread-safe? If not, do I have to be concerned with synchronization when setting context attributes?