Servlets 2

Fatemeh Abbasinejad
abbasine@cs.ucdavis.edu

Why Session Tracking?

- "Personalization" is making it easier and more pleasant for people to surf the Internet.
- Consumers and companies benefit

1. Cookies

- **Idea**: associate a cookie with data on a server
- **Advantages**: excellent alternative, and is the most widely used approach
- **Disadvantages**:
  - Extracting the cookie that stores the session identifier from the other cookies
  - Setting an appropriate expiration time for the cookie

2. URL rewriting

- **Idea**: client append some extra data at the end of the URL
- **Advantage**: works when cookies are disabled
- **Disadvantage**:
  - the server-side program has a lot of straightforward but tedious processing to do
  - all pages must be dynamically generated.

3. Hidden Form Fields

- **Idea**:
  
  ```html
  <input type="hidden" name="session" value="">
  ```
  
  when user "submits", all form data including "hidden" fields are sent to the form handler on the server
- **Advantage**: works when cookies are disabled
- **Disadvantage**: all pages must be the result of form submissions
4. Sessions

- Session is an object that lives on the server.
- It's a high-level interface built on top of cookies or URL-rewriting.
- It's automatically associated with clients via cookies or URL-rewriting.
- Use request.getSession().

Behind the scenes

1. Create a new one.
2. Assign a cookie/URL as its key.
3. Send it back.

Session Tracking Basics

1. Access the session object

   ```java
   HttpSession session = request.getSession(true);
   if there is no session, it returns null and creates one (true)
   ```

Session Tracking Basics

2. Look up information associated with a session

   - Call `getAttribute()`
   - Cast the return value to an appropriate type

   ```java
   someClass value = (someClass)session.getAttribute("SomeId");
   if (value != null) {
      doSomethingWith(value);
   } else {
      value = new someClass(...);
      doSomethingElseWith(value);
   }
   ```

getAttribute()

- Session objects have a built-in data structure that let you store any number of keys and associated values.
- Return type is `Object`, so you have to do a `typecast` to whatever more specific type of data was associated with that key in the session.
- The return value is `null` if there is no such attribute.
getAttributeNames()

- Usually, you have a specific attribute name in mind, and want to find the value associated with it, but

- You can also get all the attribute names in a given session by calling `getAttributeNames()` which returns an `Enumeration`

```java
HttpSession session = request.getSession(true);
Enumeration valueNames;
valueNames = session.getAttributeNames();
String name, value;
...
// get a value for each name
While (valueNames.hasMoreElements()) {
    name = valueNames.nextElement().toString;
    value = session.getAttribute(name).toString()
}
```

### Session Tracking Basics

3. Store Information in a Session
   - Use `setAttribute()` with key and value
   ```java
   session.setAttribute("someID", value);
   ```

4. Discard Session Data
   - `removeAttribute()` discards a specific value
   - `invalidate()` discards an entire session

### HttpSession Methods

- `getId()`: This method returns the unique identifier generated for each session.
- `isNew()`: Determines if session is new
- `getLastAccessedTime()`: Returns the time, in milliseconds since the session was last sent from the client.
- `setMaxInactiveInterval()`: Sets the amount of time, in seconds, that a session should go without access before being automatically invalidated.