Maintaining State

Greater Success with Greater Breadth of Awareness

Main point 1 Preview

1. HTTP request messages send input parameters as name/value pairs. Input parameters are text that must be accessed and converted by a servlet. This is the main mechanism web apps use to send information from the browser to the server. **Science of Consciousness**: Input parameters are an important communication mechanism between entities. Clearer awareness results in better communication between ourselves and others.

Post versus Get messages

GET /advisor/selectBeerTaste.do?color=dark&taste=malty HTTP/1.1
Host: www.wickedlysmart.com
Connection: keep-alive

POST /advisor/selectBeerTaste.do HTTP/1.1
Host: www.wickedlysmart.com
Connection: keep-alive

color=dark&taste=malty

- post has a body
- post more secure since parameters not visible in browser bar
- bookmarking POST requests does not work (well) in browsers
  - don't support since POST not idempotent
  - can bookmark, but loses body info and becomes a GET
- Intention of GET is to retrieve data; POST is to send data to be processed and stored
  - GET "should" be idempotent
  - POST generally not
  - GET requests cached for efficiency

Redirecting and forwarding requests

- servlets may internally pass (“forward”) the request processing to another local resource or to tell the client browser to issue another HTTP request to a specified URL

- forward (to another servlet or jsp in same website)

```
RequestDispatcher view =
request.getRequestDispatcher(“result.jsp”);
view.forward(request, response);
```

- redirect

```
response.sendRedirect(“http://www.cs.mum.edu”);  //to another site
or
response.sendRedirect(“result.jsp”);    //within same site
```

Difference between redirect and forward

- **forward**
  request.getRequestDispatcher("result.jsp").forward(request, response);
  - passes the request to another resource on the server
    - sometimes referred as "server side redirect"
  - request and response objects passed to destination servlet.
  - Browser is completely unaware of servlet forward and hence the URL in browser address bar will remain unchanged

- **redirect**
  response.sendRedirect("http://www.cs.mum.edu");
  - server sends HTTP status code 3xx to client along with the redirect URL (usually 302 temporary redirect)
  - client then sends a new request to the URL
  - extra round trip
  - address bar will change to new URL
  - only http message sent, request and response objects cannot be sent

HTTP input parameters: request.getParameter(…)

- send data from your HTML page to the servlet
- HTML
  - <input name="userName" type="text" />
- HTTP
  - GET /somepath/myservlet?userName=Fred HTTP/1.1
- Servlet
  - String input = request.getParameter("userName");
  - What if multiple values?
    - multiple selection list or check box.
  - GET /somepath/myservlet?colors=red&colors=blue&colors=green HTTP/1.1
  - String[] inputColors = request.getParameterValues("colors");
- Input parameters are always text/Strings
  - must validate and convert as needed
  - E.g., numbers, Dates, etc.

request.getParameterValues(…)

```
<input type="checkbox" name="habits" value="Reading">Reading
<input type="checkbox" name="habits" value="Movies">Movies
<input type="checkbox" name="habits" value="Writing">Writing
<input type="checkbox" name="habits" value="Singing">Singing
```

```
//suppose check Movies and Singing, then in doPost

String[] values=req.getParameterValues("habits");
// values[0] == “Movies”  true
// values[1] == “Singing” true
```

Main point 1

1. HTTP request messages send input parameters as name/value pairs. Input parameters are text that must be accessed and converted by a servlet. This is the main mechanism web apps use to send information from the browser to the server. **Science of Consciousness**: Input parameters are an important communication mechanism between entities. Clearer awareness results in better communication between ourselves and others.

Main point 2 Preview

Attributes are objects on the server. They promote communication and integration between components. **Science of Consciousness**: Our experience of transcending facilitates coherence and communication between different components of our brains and minds. Coherent and communication are critical to successful thought and action.

HTTP is Stateless

- HTTP is a stateless protocol.
  - Every request is new request to the server.
  - From the server's perspective, the request begins when the client opens a socket to the server, and it ends when the server sends the last packet back to the client and closes the connection.

Why is state necessary?

- Application often cannot function correctly being totally stateless.
  - A classic example is the online shopping website.
    - Somehow, between every request you made, the website knows those requests were coming for the same browser on the same computer and associates that with users shopping cart.
  - Remembering users throughout the *session*, is another use case.
  - Enabling Application Workflow
    - Often users need some form of workflow to complete a task
    - For e.g. checkout task involves
      - Filling billing address, shipping address
      - Filling payment details
      - Confirmation

How to maintain state?

- Container managed states
  - **Request scope**: life time is for a request-response cycle.
  - **Session scope**: life time is throughout the session.
  - **Application scope**: life time is for the life time of the application.

- Other ways
  - Cookies
    - temporary vs permanent
  - Hidden form fields
  - long term permanent or persistent info in an external database

Container managed state information

- different “scopes” of information a web app might need to manage
- request scope: short term computed results to pass from one servlet to another (i.e., “forward”)
- session scope: conversational state info across a series of sequential requests from a particular user
- application/context scope: global info available to any other users or servlets in this application

What is an ‘attribute’ in a web container?

- An object bound into one of the three servlet API objects
  - HttpServletRequest
  - HttpSession
  - ServletContext

- Is a name value pair
  - value has type Object
    - Vs String for input parameter
  - name is String

- Methods (on request, session, or context objects)
  - getAttribute(String)
    - must cast to specific type
    - E.g., suppose set(“manager”, bob) and bob is type Person
      - Person savedManager = (Person) request.getAttribute(“manager”);
  - setAttribute(String, Object)
  - removeAttribute(String)
    - To remove from scope
  - getAttributeNames()
    - To get an Enumeration of all attributes in scope

Request scope attributes

- only available for that request
  - E.g., set an attribute in servlet (controller) and then forward request to a view, jsp page (same request still)
  - request. setAttribute(“myAttr”, someInfo);
  - RequestDispatcher view = request.getRequestDispatcher(“result.jsp”);

    view.forward(request, response);

  Then, on the target JSP page
  - request. getAttribute(“myAttr”); //will retrieve the someInfo object

Context scope attributes

- Application level state
  - request.getServletContext().setAttribute(“myAttr”, test);
  - request.getServletContext().getAttribute(“myAttr”);

- Caution: global!!
  - Shared by every servlet and every request in the application
  - Not thread safe
    - Nor session attributes
    - Only request attributes thread safe
    - Any updates should be synchronized

Session scope attributes

- sessions are collections of objects (attributes) that are unique to a set of connected requests from a single browser

- Sessions are a critical state management service provided by the web container

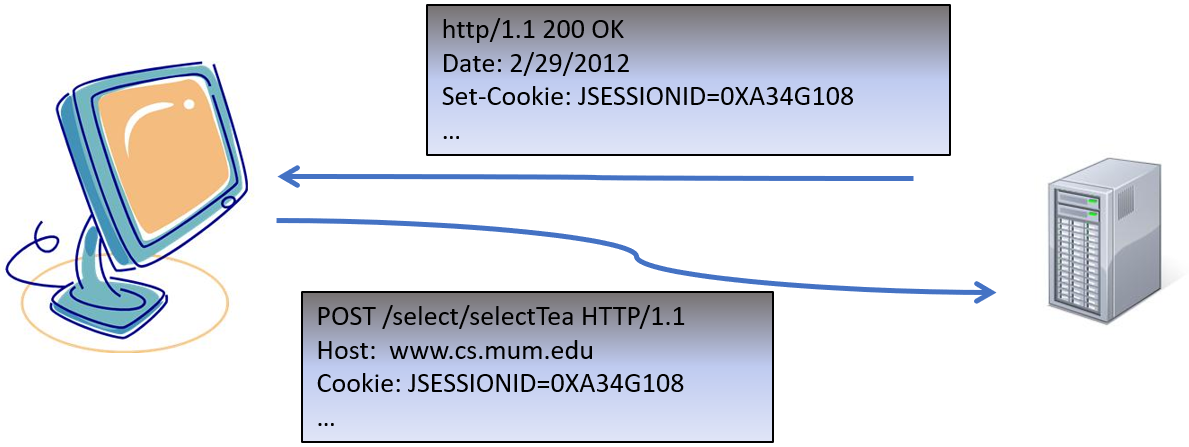
Main point 2

Attributes are objects on the server. They promote communication and integration between components. **Science of Consciousness**: Our experience of transcending facilitates coherence and communication between different components of our brains and minds. Coherent communication is critical to successful thought and action.

Main point 3 Preview

Web containers use cookies to create sessions. **Science of Consciousness**: HTTP cookies enable relationships and communication between client-server entities distributed across space and time. At the level of pure consciousness and the unified field everything is connected with everything else.

Session tracking via cookie exchange



- A web conversation, holds information across multiple requests.
- before container responds, saves the session info to some datastore
  - then sends an HTTP “cookie” with session id
  - Set-cookie header
- browser saves cookie
  - Places in Cookie header with the next request
  - All cookies from a given website returned when browser sends any request to that website
    - Suppose have multiple tabs open to the same website
- container reconstitutes session from storage before calling servlet

Example cookie exchange

- browser request for www.example.org homepage:

  GET /index.html HTTP/1.1
  Host: www.example.org
  ...

- server responds with two Set-Cookie headers:

  HTTP/1.0 200 OK
  Content-type: text/html
  Set-Cookie: theme=light
  Set-Cookie: sessionToken=abc123; Expires=Wed, 09 Jun 2021 10:18:14 GMT

- browser sends another request to visit the spec.html page:

  GET /spec.html HTTP/1.1
  Host: www.example.org
  Cookie: theme=light; sessionToken=abc123

Session lifetime

Client side

- Browser discards all “temporary” cookies when it closes
- Every tab or window of browser will have access to all cookies

Server side

- How to get a session
  - session = request.getSession(); //creates new session if none exists
    - session.isNew(); //checks whether is a new session
    - request.getSession(false); //returns null if none exists
- How to get rid of the session
  - sessions can become a memory resource issue
  - container can’t tell when browser is finished with session
  - 3 ways for container to remove sessions
    - session timeout in the DD
    - session.setMaxInactiveInterval(20*60); //seconds
    - session.invalidate(); //immediate

```
<session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
</web-app>
```

(HTTP) Cookies

- can be used for other things besides implementing sessions
  - temporary cookie
    - browser removes when it closes
    - this is default
    - session cookies are like this
  - permanent cookie
    - a cookie that has a max age set

- Sending a Cookie

```
Cookie cookie = new Cookie("Name", "Jack");
response.addCookie(cookie);
cookie.setMaxAge(seconds);
```

  - A positive value indicates that the cookie will expire after that many seconds have passed.
  - A zero value causes an existing permanent cookie to be deleted.

- Reading a cookie
  - Cookies come with the request
  - can only get all cookies, then search for the one you want.

```
for (Cookie cookie : request.getCookies()) {
if (cookie.getName().equals("Name")) {
String userName= cookie.getValue();  } }
```

Demo

5 ways to maintain state

Container managed state (3 scopes)

1. request scope: destroyed when servlet finishes processing request
2. session scope: destroyed when user closes browser
3. application scope destroyed when Tomcat container stopped.

4. Cookies saved on browser,

   temporary (deleted when the browser closes)

   permanent

5. Hidden fields on a form

- Answer questions 14,15,16 for today's quiz

Cross-site request forgery (CSRF)

- Confidential information should never be stored in HTTP Cookies

- Suppose you are on a site that takes user comments and they are not filtering HTML inputs.
  - Someone might insert an image link that is actually a request for money to be moved from your account.
  - If you are logged into your account in another tab then this request would go to your bank with your session cookie
  - (Your bank should have a lot more security and safeguards for sensitive actions)

<img src=http://mybank.com/withdraw?account=bob&amount=10000&for=sam alt="missing picture">

https://en.wikipedia.org/wiki/Cross-site_request_forgery

Main point 3

Web containers use cookies to create sessions. **Science of Consciousness**: HTTP cookies enable relationships and communication between client-server entities distributed across space and time. At the level of pure consciousness and the unified field everything is connected with everything else.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

Managing State: Continuum of Awareness

1. State information can be stored on the server in the session, and on the browser with cookies.

2. Sessions are managed by the container, which typically use cookies to store session identifiers on browsers.

3.Transcendental consciousness is the direct experience of the unified field. This experience brings orderliness and removes impurities and stresses from memory and thought, resulting in broader awareness and more powerful thoughts.

4. **Impulses within the Transcendental Field:** When one's awareness is connected to the transcendental field then even dynamic focused thoughts have the spontaneous support of the powerful unbounded awareness of pure consciousness.

5. **Wholeness moving within itself:** In unity consciousness, one not only experiences a continuum of unbounded awareness, pure consciousness, but one also appreciates all external entities as expressions of this continuum of awareness.