CS 416
Web Programming

Filters and Listeners

Dr. Williams
Central Connecticut State University

Topics for Lecture

- Web filters
- Web listeners

Web filters

- Web filter is an object that can dynamically intercept a request and manipulate the data or **perform other actions before** forwarding the request to the original servlet
- Essentially captures request before servlet receives it and performs some action then lets the servlet process it normally and then can **process afterward**
- Receiving servlet unaware of filter

Usefulness

- Filters are often used to log information about the request
  - Log the parameters being passed in that way if an error occurs can trace why. Filter can then be removed/added in production without changing the main business logic
  - Security auditing access to resources
  - Ability to capture before as well as after making it useful for measuring performance
  - Protecting secure pages

Registering Filter

- To create a Filter your class must extend the javax.servlet.Filter class
- The servlet then must be registered with the application server (Web annotation one way)

@WebFilter(filterName="MyFilter",urlPatterns={"/MyServlet"})

In addition to these basic parameters you can also add initial parameters in the same way as you did with the @WebServlet tag or web.xml

@WebInitParam(name="attrName",value="attrValue")

Filter processing

- Intercepting a request is done by overriding doFilter Method
- Method very similar to doGet(), doPost() the filter receives the request and response object as well as a **FilterChain**
- FilterChain contains the list of filters to run before the servlet processes

Filter syntax

```
public void doFilter(..request,..response,..filterChain)
    // do action before servlet
    String something request.getParameter(“something”);
    request.setAttribute(“attr”,value);
    request.getSession().setAttribute(“attr1”,value1);
    getContext().setAttribute(“attr2”,value2);

    filterChain.doFilter(request,response);

    // do action after servlet
}
```

Filter example

- Note to write to the application server log you use:

```
request.getServletContext().log(“my log entry”);
```

Web Listeners

- Web listeners allow you to track when changes are made to various objects that are part of server interactions
- The basic purpose is to listen for events such as creating/destroying request, session, or context or attributes on any of these

Usefulness

- Listeners are very useful for debugging, by being able to track when changes are made over the course of a server interaction
- Also useful for freeing up resources. For instance you could have code to make sure that all connections to a database have been closed on the destruction of a request as a safety precaution
- Also useful for how the application is being used, for example track how long a user is typically interacting with the site and adjust the life of the session to be more in line with that to free resources

Creating a Listener

- To create a Listener your class must implement the javax.servlet.XXX class
- Where the class depends on what you are listening for:
  - javax.servlet.**ServletContextListener** – create and destroy events
  - javax.servlet.**ServletContextAttributeListener** – setting and changing attribute values
  - javax.servlet.**ServletRequestListener**
  - javax.servlet.**ServletRequestAttributeListener**
  - javax.servlet.http.**HttpSessionListener**
  - javax.servlet.http.**HttpSessionAttributeListener**

Listener events

- The listeners fall into two categories a create and destroy listener
  - ServletContextListener
  - ServletRequestListener
  - HttpSessionListener
- Each of these have methods that can listen for whenever that type of object is initialized and when it is destroyed
  - Initialized means first time used
  - Destroyed means no longer in memory

Listener Attribute events

- The second type of listener is related to changes in the attributes on the object being listened to
- Events
  - Attribute created – attribute added the first time
  - Attribute removed – remove attribute called
  - Attribute changed – an attribute that already had a value is having its value set again

Registering a Listener

- For a listener to work it must be registered with the application server
- Since listeners listen to all events regardless of page their annotation is simple

```
@WebListener()
```

Listener example

Your turn: Creating a Performance logger

Create a filter to log the performance of our review servlet

- Create a filter that acts on calls to "/ReviewPersistingSolnServlet“
- The filter should:
    - Capture the start time (System.nanoTime())
    - Allow the servlet to run
    - Capture the end time
    - Log to the server the total time for the servlet

Your turn: Creating a session listener

- Create a listener that should track the time between changes in the context attribute “numNamesSoln”
  - Your code will need to listen for create and change events on the context
  - Check that the item that changed is the one we care about (i.e. name matches)
  - Store the time of the current change (context)
  - Output the difference in time to the log

Real application of web filter

- AuthorizationFilter – protecting secure pages