Scope and State Handling in JSP

CS 4640 Programming Languages for Web Applications

[Based in part on SWE432 and SWE632 materials by Jeff Offutt]
[Robert W. Sebesta, “Programming the World Wide Web]
Session State Information

The initial versions of the web suffered from a lack of state

If a web app needed multiple screens, there was no way for state to be accumulated or stored

• This is due to the stateless property of HTTP

In reality, we may want to keep track of the information (i.e., state)
Session Tracking

• Web sites that are service-oriented or e-commerce need to maintain user state

• This is called session tracking

• Session = a series of related interactions between a client and a web server

• Session tracking = keeping data between multiple HTTP requests

• The web brings in unique constraints:
  • HTTP is connectionless
  • Web apps are distributed
State on the Web

• Two assumptions (traditional software):
  1. The software components share physical memory
  2. The program runs to completion with active memory

• These assumptions are violated in web apps due to
  1. Distributed software components
  2. Connectionless nature of HTTP

• Need ways to store and access variables and objects to keep state in web apps

Public access and parameter passing are not enough for web apps
Session and Context Scope

Container Engine

Servlet S1
  session object 1
    JSP 1
    JSP 2

JSP 3

Servlet S2
  session object 2
    JSP 4

context object

Session 1

Session 2

Context (application)
JSP: Sharing Data with Session and Context Objects

• Using JSP Scriptlets
  • `getParameter();` // retrieves client form data
  • `session.getAttribute()` and `session.setAttribute();`
  • `context.getAttribute()` and `context.setAttribute();`

• For example:
  `<% session.setAttribute("ID", request.getParameter("ID")); %>`

Application scope

Predefined variable
Sharing Data with Scope

• The previous approach makes the code kind of clumsy
• Alternative approach – expanded use of JavaBean

A Java Bean is a Java class with 3 characteristics:

1. public class
2. public constructor with no arguments
3. public get and set methods (called getters and setters)

Property: A special, simple data object (that is, variable)

• getName () … <jsp:getProperty>
• setName (String name) … <jsp:setProperty>
• Note that a bean is not a Java language feature, but a design convention (pattern)
JSP Scope and State Management

• JSPs formalize this with **four** separate scopes
  1. **Page**: Within the same program component (web page)
  2. **Request**: Within the same request
  3. **Session**: Within all requests from the same session
  4. **Application**: Within all sessions for one servlet context

• Each can be accessed by **different sets** of program components
• Some exist for different periods of **time**
Sharing Data with Scope

[Scope and Handling State in JSPs, slides from SWE 642, reproduced with permission from J. Offutt]
useBean Action Tag

useBean

• Causes a JavaBean object to be instantiated
• Gives a name to the new object (id=)
• Defines the scope (scope=)
• Declares the location (bean details)
Java Bean Example

Syntax for using a bean

```jsp
<%@ page import="jspexamples.*" %>
<jsp:useBean id="letterColor" class="AlphabetCode" scope="page" />
```

- **Converting to Java import statement**: Java bean should always be in a package as required by most Web servers.
- **ID name to use for object**:
  ```java
  AlphabetCode letterColor = new …
  ```
- **Name of class**:
  ```java
  Class AlphabetCode
  ```
- **Defines accessibility**:
  JSPs offer several useful scopes for variables (page, request, session, application).

**Note**: `scope="application"` allows Beans to be shared among different servlets → lead to interactions among each other …
Properties of Beans

**setProperty** gives a value to a property in a bean

```jsp
<jsp:setProperty name="langBean" property="language" value="Java"/>
```

Equivalent to the call: `langBean.setLanguage("Java");`

```jsp
<jsp:setProperty name="langBean" property="*"/>
```

Sets all of the properties with values from HTML form

**getProperty** retrieves the value of a property

```jsp
<jsp:getProperty name="langBean" property="language"/>
```

Equivalent to the call: `langBean.getLanguage();`

- Property – begin with a **lower case** letter
- Getters and setters – **property name** start with a capital letter
Note on Java Beans

• Using Java Beans increases separation between the HTML and Java

• The Beans / Property pattern provides a convenient standard for implementing standard Java classes

• JSP’s useBean uses Java reflection to translate property names (for example, “language”) to method calls that are assumed to exist (“setLanguage()” and “getLanguage()”)

• The bean does not have to have an object with the name of the property, as long as it has a getter or setter
Summary
Sharing Data with Java Bean

• Use the **scope attribute** in the `<jsp:useBean>` action

```jsp
<jsp:useBean id="languageBean" class="lang.LanguageBean" scope="session">
  <jsp:getProperty name="languageBean" property="name">
```

• **Scoping keywords**:
  • `scope="request"` for the *request* object
  • `scope="session"` for the *session* object
  • `scope="application"` for the *context* object

• The page scope is default – local variables
Summary

• Programmers often get state management wrong
  • They learned “how” without learning “why” (the theory)
  • They don’t understand the differences in the various scopes
  • They forget to consider which scope to use as part of design

• State management is very different from traditional programming

• These scopes are quite powerful

• New frameworks beyond Java EE often add different scopes and different semantics on the same scopes