Session Tracking in Java Servlets

James Baldo Jr.

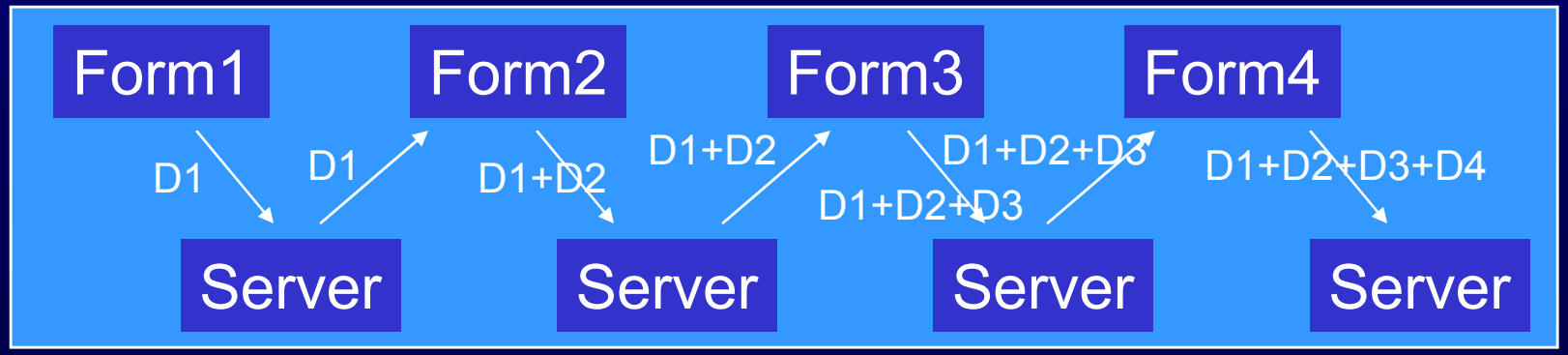
SWE 432

Design and Implementation of Software for the Web

Session State Information

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

- If you wanted multiple screens, there was no way for data to be accumulated or stored



Session Tracking

- Web sites that are service-oriented or e-commerce need to maintain user states
- This is called *session tracking*

Session Tracking (2)

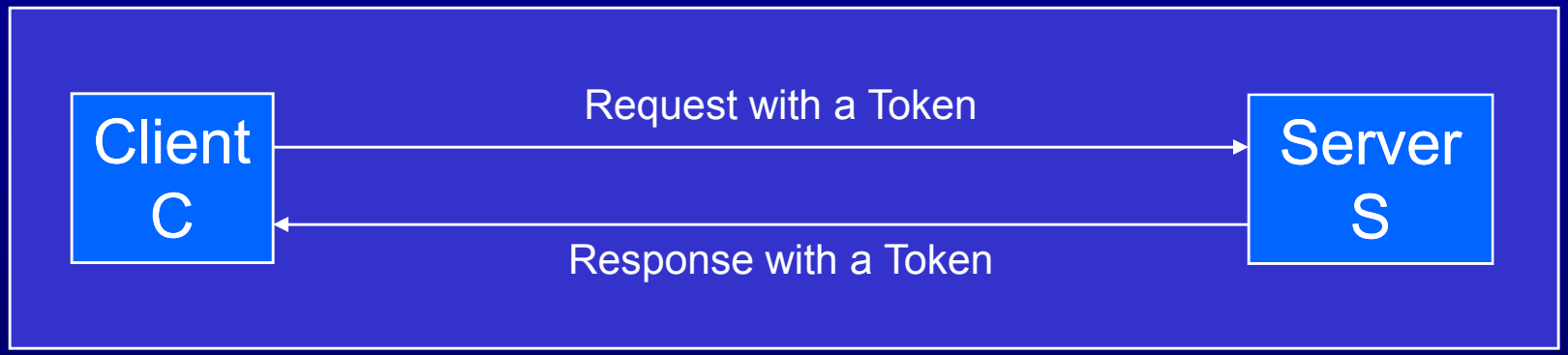
- Session tracking refers to passing data from one HTTP request to another
- Servlets can use several methods to do session tracking:
  1. Include data as extra parameters in URL (rewriting)
  2. Hidden form fields
  3. Cookies

     3.b) Cookies within Servlet API session tracking tools
  4. Sessions using the Secure Sockets Layer (SSL)

     (not discussed in 432)

Session: **A series of related interactions between a client and a web server (similar to a use case)**

Session Tracking (3)



All four work by exchanging a *token* between the client and the server

Non-servlet Methods (Stone Age)
1) URL Rewriting

- Forms usually add parameters

  URL ? P1=v1 & P2=v2 & P3=v3 & …
- You can add values in the URL as a parameter:

  HREF = "../servlet/X ? SneakyParam=42">

  or: User=george">
- This is used as a key to find the saved information about the user george.
  - Messy and clumsy
  - Long URLs
  - Information on URL is public
  - All HTML pages must be created dynamically

Non-servlet Methods
2) Hidden Form Fields

- Generate HTML pages with forms that store "hidden" information:

  ```
  <INPUT Type=hidden Name=USER Value=george>
  ```

- Somewhat clumsy
- Insecure
- All HTML pages must be created dynamically

Non-servlet Methods
3) Cookies

- ***Cookies*** are small files or text strings stored on the client's computer
- Created by the web browser
- Arbitrary strings stored on the client
- From the server's (Java) perspective: var=value pairs
- Java coding:

```
Cookie c = new Cookie ("user", "george");
c.setMaxAge (5*24*60*60); // expires in 5 days, in seconds
response.addCookie (c);      // sends cookie to client.
```

Non-servlet Methods
3) Cookies – cont.

- Cookies are very useful and simple
- Not stored with the HTML content
- Convenient way to solve a real problem
- But cookies are scary!
  - It's as if I stored my files at your house
  - Cookies go way beyond session tracking
  - Cookies provide a way to do **behavior tracking**

Bronze-age method
3.b) Servlet API

The servlet API uses cookies to provide a *simple*, *safe*, *flexible* method for session tracking

- Cookies are handled automatically
- HttpSession stores data in the current active object
- Data disappears when the object is destroyed
- Object is destroyed after the session ends, by default 30 minutes after the last request

Servlet API (2)

- void setAttribute (String name, Object attribute) : Adds an item to the session
- Object getAttribute (String name) : Returns the value stored for the given name
- void removeAttribute (String name) : Removes an item from the session
- Enumeration getAttributeNames() : Returns an enumeration of all the value names that are stored for this session
- String getID() : Returns the session ID
- void invalidate() : Removes the current session

Servlet API (3)

- These methods are **not** synchronized
- Multiple servlets can access the same session object at the same time
- If this can happen, your program should synchronize the code that modifies the shared session attributes

Using Session Objects

- Get a session object:

  HttpSession s = request.getSession (true);
  - true: create if it does not exist.
  - false: return null if it does not exist.
- Put objects into the session object (cannot put primitive types):

  s.setAttribute ("answer", 42); // does not work

  s. setAttribute ("answer", new Integer (42));
- Getting primitive values from session objects:

  Integer ansobj = (Integer) s.getAttribute ("answer");

  int ans = ansobj.intValue ();
- Deleting session:

  s.invalidate (); // Information is thrown away

Session Definition

A session is defined by

- The web server
  - Servlet container
  - Servlet context
- The client
  - IP address
  - Browser
- Session objects are kept on the server
- Each session object uses different parts of memory (instances of data values) on the server

Session Objects in General

- Generally speaking, session handling is really about sharing data
- A Web application is comprised of several software components
- The characteristics of a Web app means that the components do not communicate directly
  - Independent processes (really, threads)
  - Stateless protocol
  - Client-server or N-tier architecture
  - Execution flow always goes through a client

How can these independent components share data?

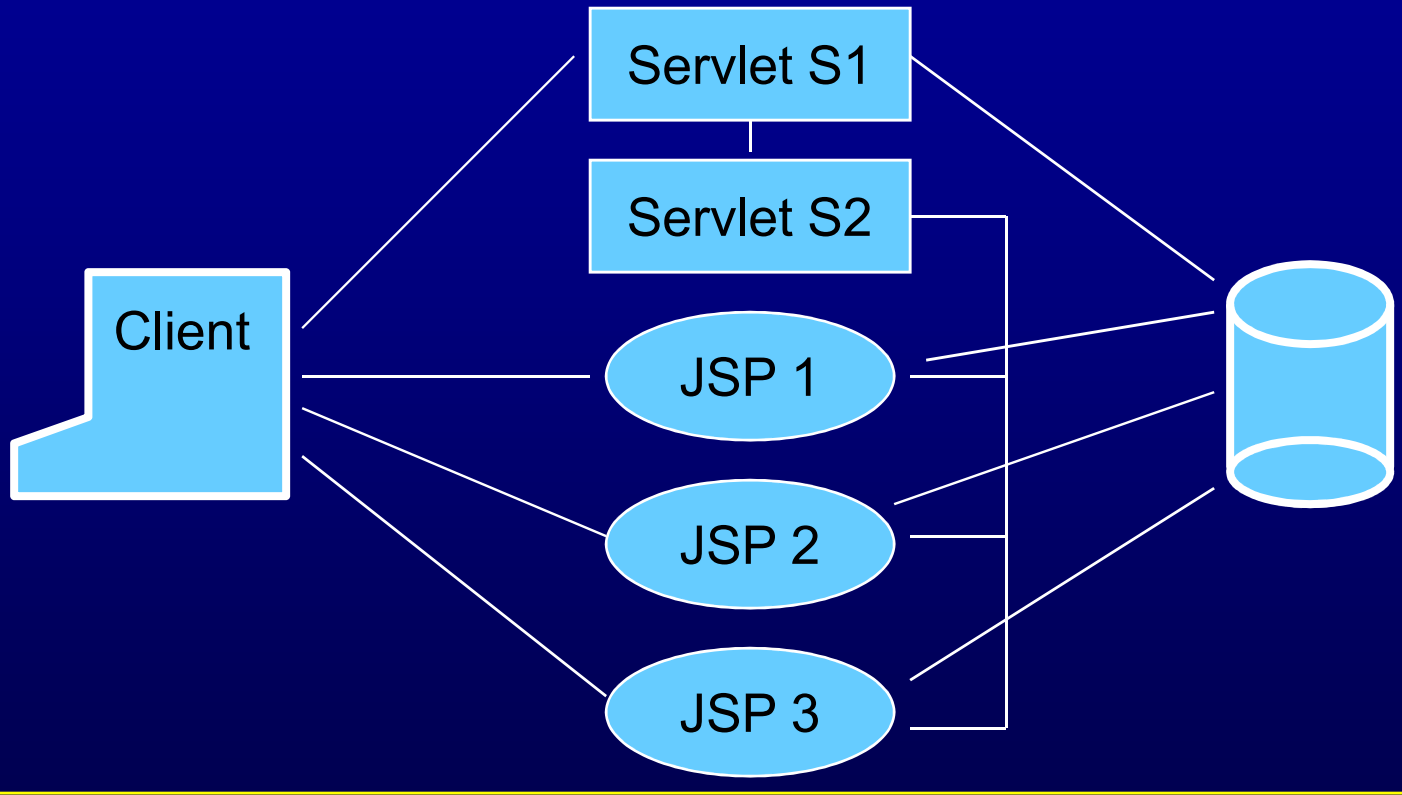
Data Scope

- Access levels (scope) in Java:
  - private (within a class)
  - protected (within package and through inheritance)
  - package (inheritance within the package)
  - public (entire application)
- Data sharing in Java:
  - Two components can share data if they are in the same scope
  - Two components can share data by passing parameters

BUT … Public access and parameter passing are not possible in Web applications!

Example

Consider a small Web app with 2 servlets and 3 JSPs



How can the servlets and JSPs share data?

Sharing Data : Hidden Form Fields

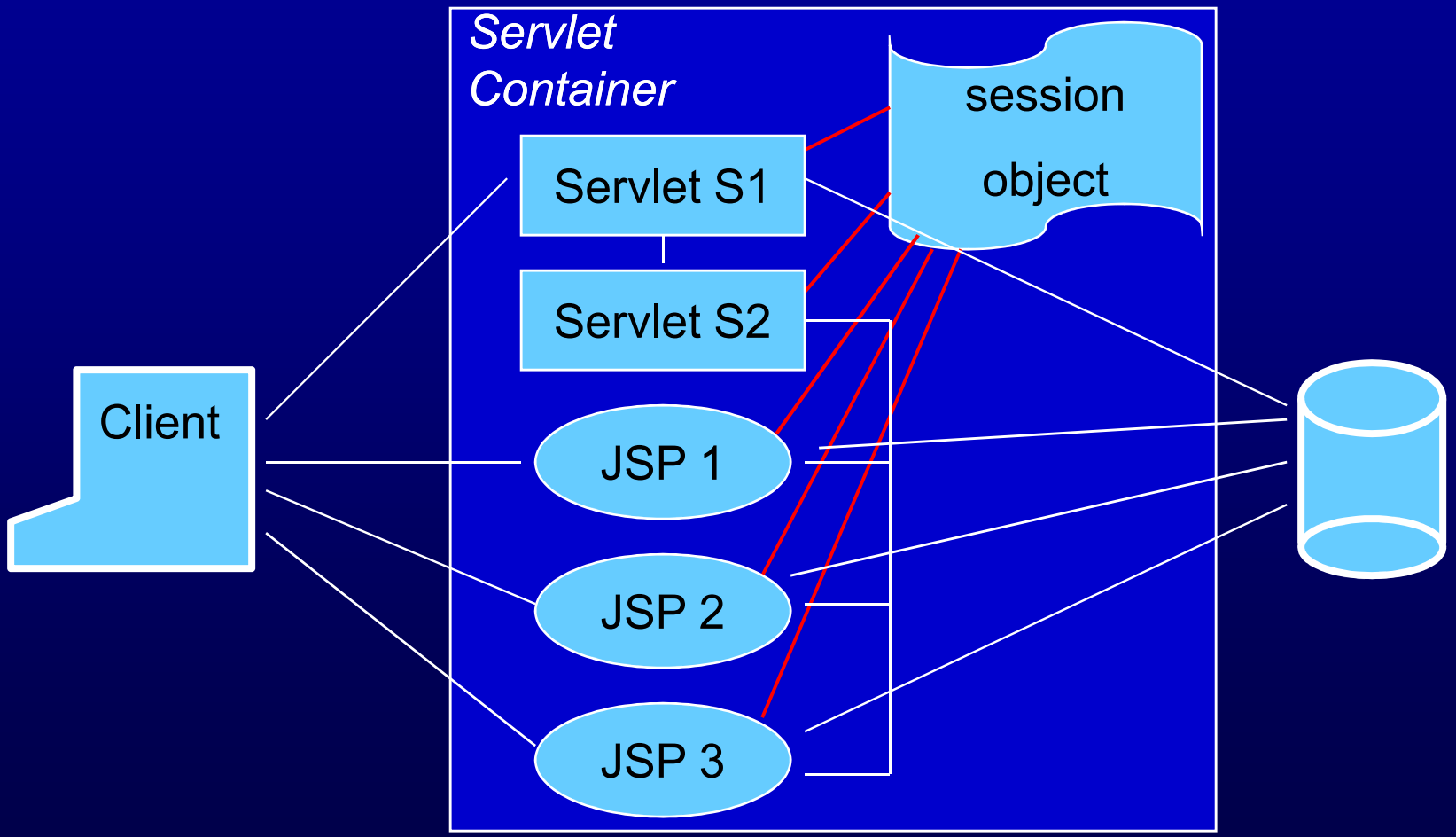
- Flows of control go through the client
- Data that must be passed from one software component to another can be stored in hidden form fields in the HTML pages
- Several problems:
  - Insecure – users can see the data
  - Unreliable – users can change the data
  - Undependable – users can use the back button, direct URL entry, and URL rewriting to skip some hidden form fields
- Still useful in limited situations

Sharing Data : Session Object

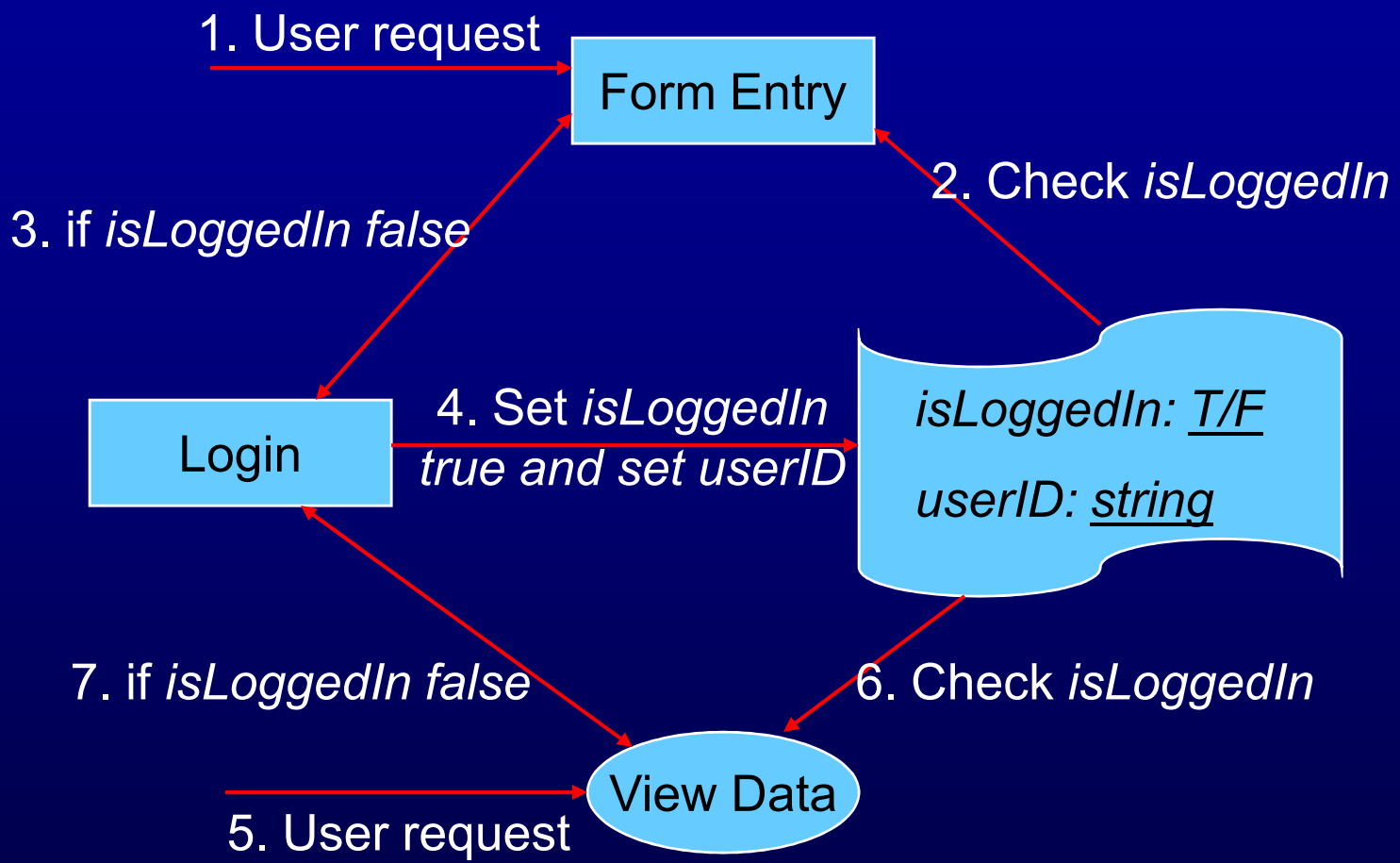
- One program component can store a value in the session object
- Another component can retrieve, use, and modify the value
- Depends on the servlet container:
  - Software components are threads, not processes
  - Servlet container stays resident and can keep shared memory

Session Data Example

Software components share “container” access data



Login Example



More on Maintaining State

Sometimes we want to share session data among multiple clients

1. User session state

   Cookies and session object

2. Multi-user session state

   Servlet-context object

Why do we need them?

- Chat rooms: Allow multiple users to interact
- Group working: Online meeting
- Online bidding
- Reservation systems

Servlet Context Object

The servlet context object supports resources that can be shared by groups of users:

- Information about servlet's environment:
  - Server name
  - MIME type
- Method to write to a log file (log())
- Share information through context attributes
  1. getAttribute()
  2. setAttribute()
  3. removeAttribute()

Session Summary

- A *session* is a single coherent use of the system by the same user
- Sessions need to *maintain state*
- Maintaining state is difficult because HTTP is *stateless*
- J2SE applications keep track of state within the *session object*
  - The session object is based on *cookies*
  - Cookies are handled by the software libraries, giving a useful *abstraction* for programmers