1 Web App Architecture

1.1 Introduction

How do web clients and web servers talk to one another?

Our goal is to build a web application that clients around the globe can access.

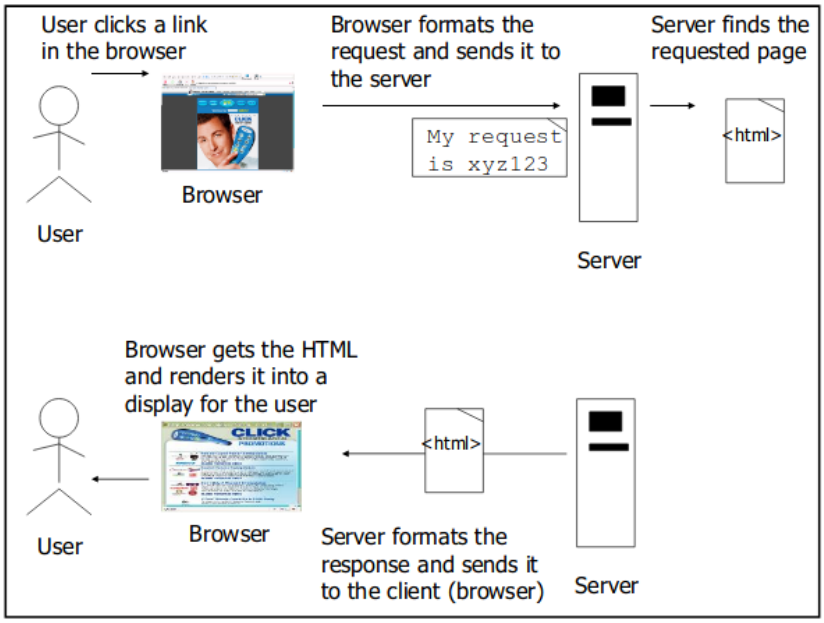
1.2 Client/Server

Web Server

Takes a client request and gives something back to the client.

Client

Lets the user request something on the server, and shows the user the result of the request.



1.3 Clients and servers know HTML anh HTTP

HTML

1. Server answers request
2. Server sends content to browser
3. Browser displays

Servers oftend send the browser a set of instructions written in HTML. All web browsers know what to do with html.

HTTP

1. Client sends HTTP request
2. Server answers with HTTP response

When a web server sends an HTML page to the client, it sends it using HTTP.

What is the HTTP protocol?

Key elements of the request stream

- HTTP method: the action to be performed
- The page to access: a URL
- Form parameters: like arguments to a method

Key elements of the response stream

- A status code: for whether the request was succesfull
- Content-type: text, picture, ...
- The content; the actual HTML, images , ...

What is the request?

First thing you will find is an HTTP method name.

The method name tells the server the kind of request that's being made, and how the rest of the message will be formatted.

The HTTP protocols has several methods, but the ones you'll use most often are **GET** and **POST**

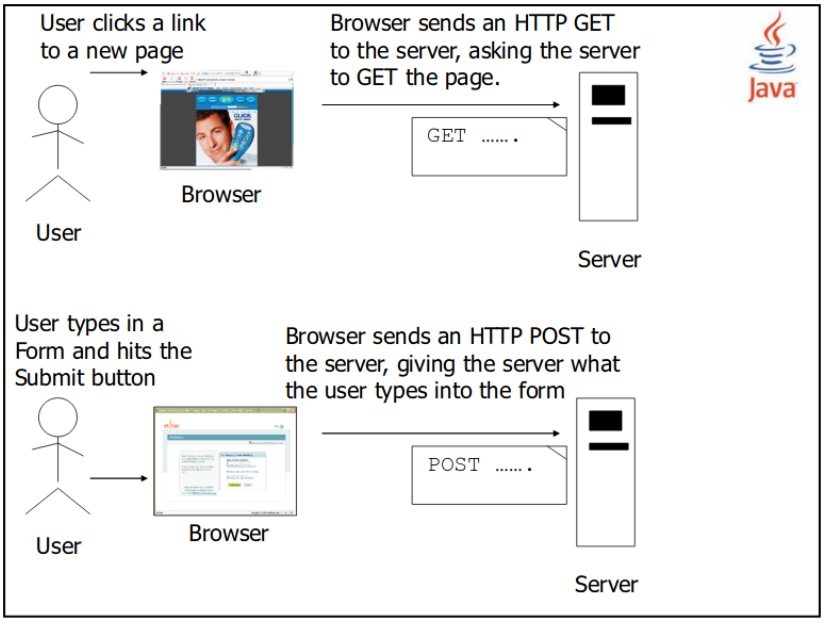
GET and POST

GET

The simplest HTTP method, the point is to get something back from the server.

POST

The more powerfull request, you can request something and at the same time send form data to the server.

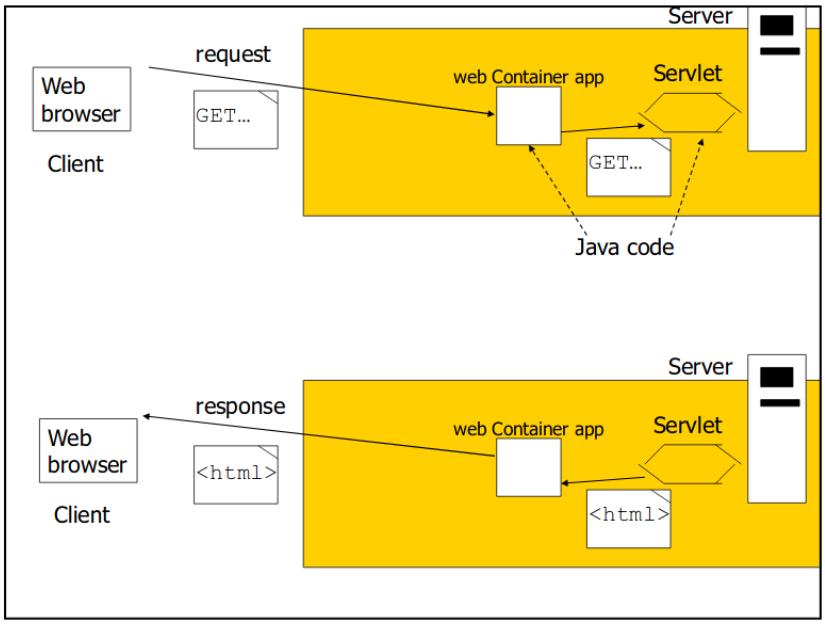


Container

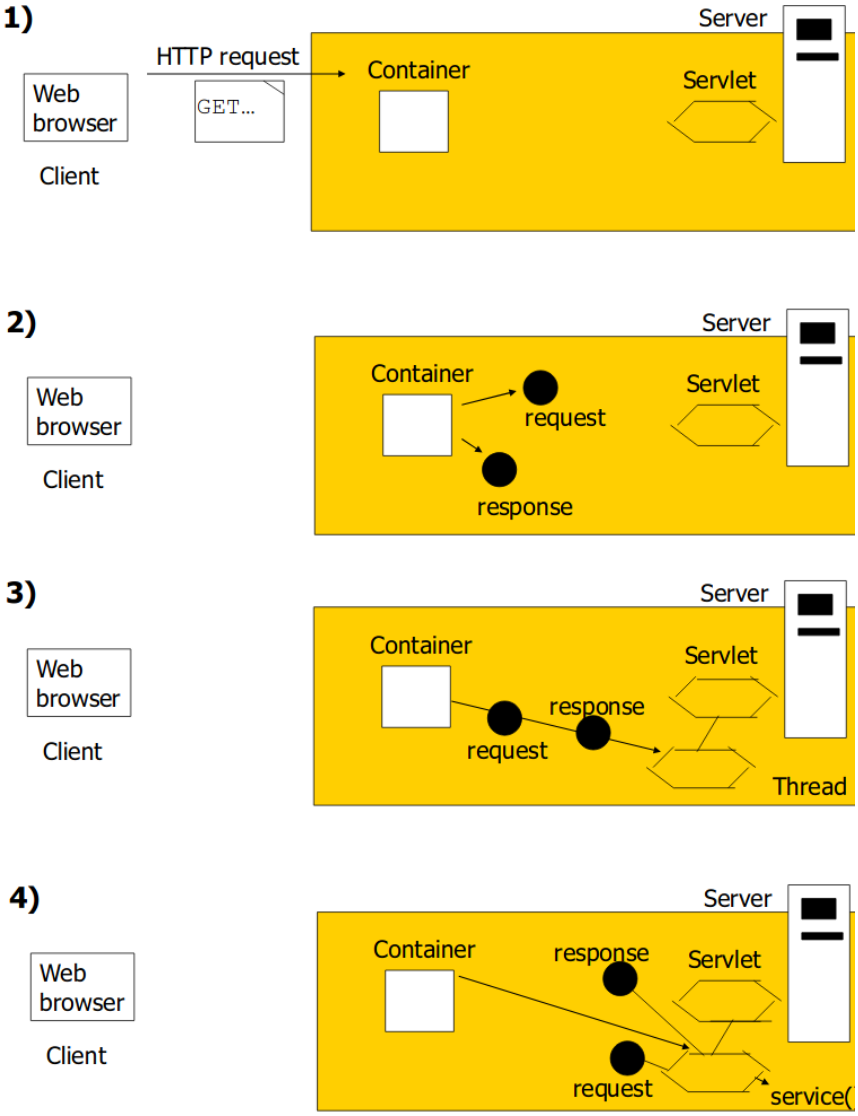
Servlets don't have a main() method. They are under the control of another Java application called a **container**.

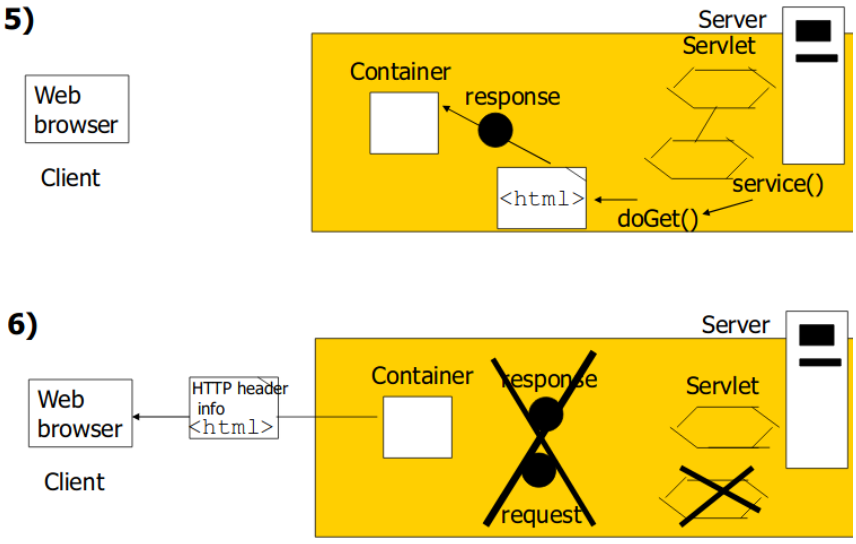
When your web server application gets a request for a servlet, the server hands the request not to the servlet itsel, but to the Container in which the servlet is deployed.

It is the Container that gives the servlet the HTTP request and response, and it is the Container that calls the servlet's method.



How the Container handles a request





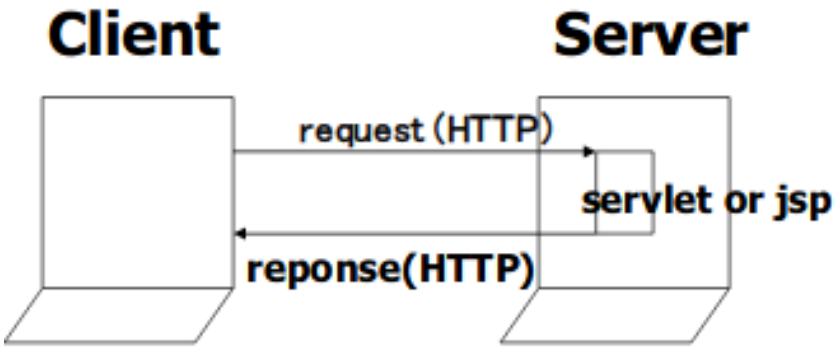
1. User click a link that has a URL to a servlet instead of a static page.
2. The container sees that the request is for a servlet, so the container creates two objects; HttpServletResponse and HttpServletRequest.
3. The container finds the correct servlet based on the URL in the request, creates or allocates a thread for that request, and passes the request and response objects to the servlet thread.
4. The container calls the servlet's service() method. Depending on the type of request, the service() method calls either the doGet() or doPost() method. For this example, we will assume the request was an HTTP GET.
5. The doGet() method generates the dynamic page and stuffs the page into the response object. Remember, the container still has a reference to the response object.
6. The thread completes, the container converts the response object into a HTTP response, sends it back to the client, then deletes the request and response objects.

2 Servlets

2.1 Introduction

The client requests that som action is performed, the server performs the action and responds to the client.

This request-response model of communication is the foundation for the highest-level views of networking in Java-Servlets and JavaServer Pages.



What is a SERVLET?

A servlet is a **Java programming language class** used to extend the capabilities of servers that host applications accessed via a **request-response programming model**.

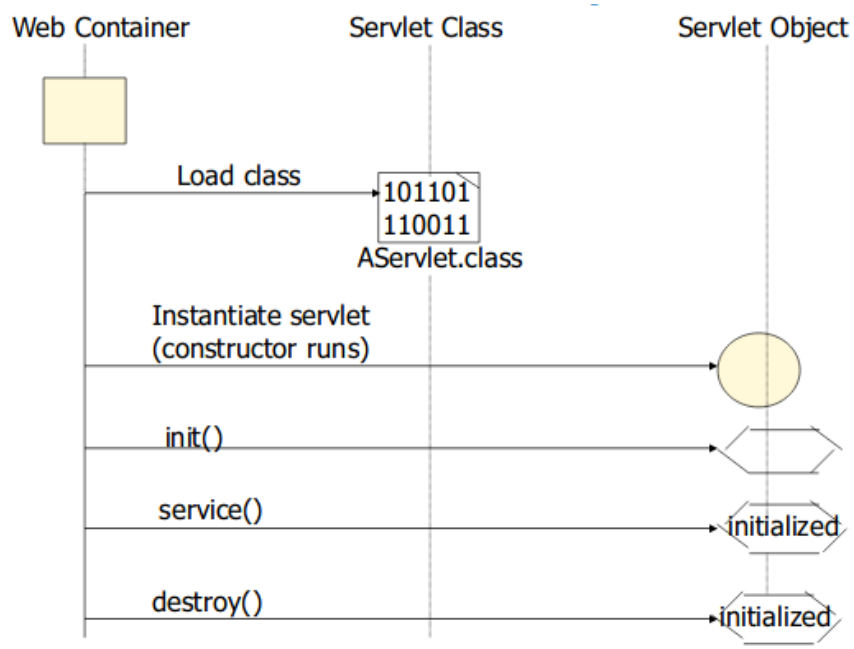
Java Servlet technology defines HTTP-specific servlet classes.

Servlet Life Cycle

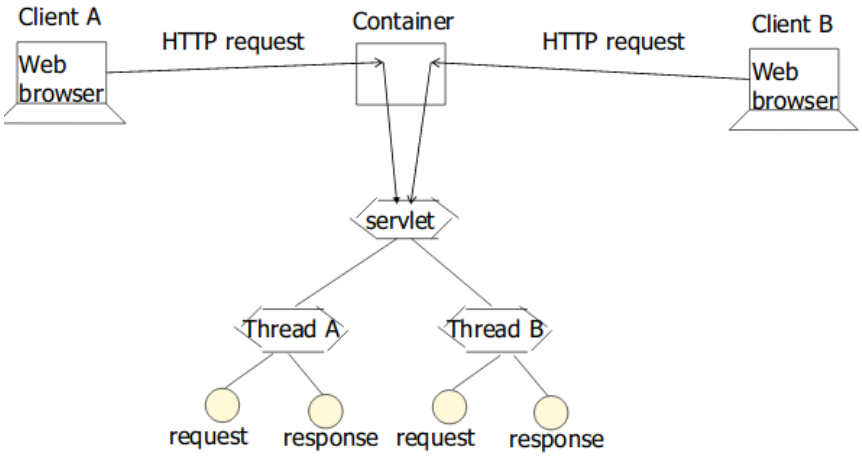
If an instance of a servlet does not exist, the container:

- Loads the servlet class
- Creates an instance of the servlet class
- Initializes the servlet instance by calling the init method
- Invokes the service method, passing a request and response object

If the container needs to remove the servlet, it finalizes the servlet by calling the servlet's destroy method.



Each request runs in a separate thread



GET and POST

The two most common HTTP request types

- GET: retrieves information from the server
- POST: sends data to server, such as authentication information or data from a form

HTTPServlet Class

Web-based servlets typically extend class HttpServlet

- method doGet() responds to GET requests
- method doPost() responds to POST requests

2.2 Handling http GET Requests

WelcomeServlet demonstration

The servlet and HTML document demonstrate a servlet that handles HTTP get requests

WelcomeServlet.java

```
@WebServlet("/welcome1")
public class WelcomeServlet extends HttpServlet
{
    //process "get" requests
    @Override
    protected void doGet(HttpServletRequest request, HttpServletRespon
se response)
        throws ServletException, IOException
    {
        response.setContentType("text/html");
        try(PrintWriter out = response.getWriter()){
            //Write HTML document
        }
    }
}
```

welcomeForm.htmm

```
<!DOCTYPE html>
<html>
    <head>
        <title>Handling an HTTP Get Request</title>
    </head>
    <body>
        <form action="welcome1" method="get">
            <p>
                <label>Click to invoke servlet</label>
                <input type="submit" value="Get HTML doc"/>
            </p>
        </form>
    </body>
</html>
```

The HTML document provides a form that invokes the servlet The form's action (welcome1) specifies the URL path that invokes the servlet, and the form's method indicates that the browser sends a get request to the server. This results in a call to the servlet's doGet method.

How the container found the servlet

A servlet can have 2 names

- Client-known URL name: e.g. the link to register/registerMe servlet
- Programmer-known file name: e.g. SignUpServlet.class

Mapping servlet names improves your app's flexibility and security

Gives you the flexibility to move things around without having to change the client code that refers to the old location of the servlet files.

Better security: the client doesn't know how things are structured on your server.

Handling http get requests containing data

In WelcomeServlet.java:

```
String firstname = request.getParameter("firstname");
```

In welcomeForm.html:

```
<input type="text" name="firstname"/>
```

2.3 Handling http POST requests

A http post request is often used to post data from an HTML form to a server-side form handler that processes the data.

Browsers often cache web pages so they can quickly reload the pages. The browser minimizes the amount of data that must be downloaded for you to view a web page.

Browsers typically do not cache the server's response to a post request, because the next post might not return the same result.

Differences between GET and POST

- GET requests can be bookmarked, POST requests cannot.
- GET is meant to be used for getting things; POST is meant to be used for sending data to be processed.

Demonstration

WelcomeServlet.java

```
protected void doPost (HttpServletRequest request, HttpServletResponse
 response)
    throws ServletException, IOException

    //instead of doGet()
```

welcomeForm.html

```
<form action="welcome1" method="post">
```

2.4 Redirecting Requests to Other Resources

The RedirectServlet.java recieves a page parameter as part of a get request, then uses that parameter to redirect the request to a different resource.

Demonstration

RedirectForm.html

```
<a href = "redirect?page=oracle">
```

RedirectServlet.java

```
@WebServlet("/redirect")
public class RedirectServlet extends HttpServlet
{
    //process "get" request from client
    @Override
    protected void doGet(HttpServletRequest request, HttpServletRespon
se response)
        throws ServletException, IOException
    {
        String location = request.getParameter("page");
        if (location != null)
        {
            if (location.equals("oracle"))
                response.sendRedirect("http://www.oracle ...");
            else
                if (location.equals("welcome"))
                    response.sendRedirect("welcome1");
        }
    }
}
```

Redirect vs Request Dispatch

Request Dispatch

The servlet calls:

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

When a servlet does a request dispatch it's like asking a co-worker to take over working with a client.

The co-worker ends up responding to the client. The user never knows someone else took over, because the URL in the browser bar doesn't change.

Redirect

When a servlet does a redirect it's like asking the client to call someone else instead.

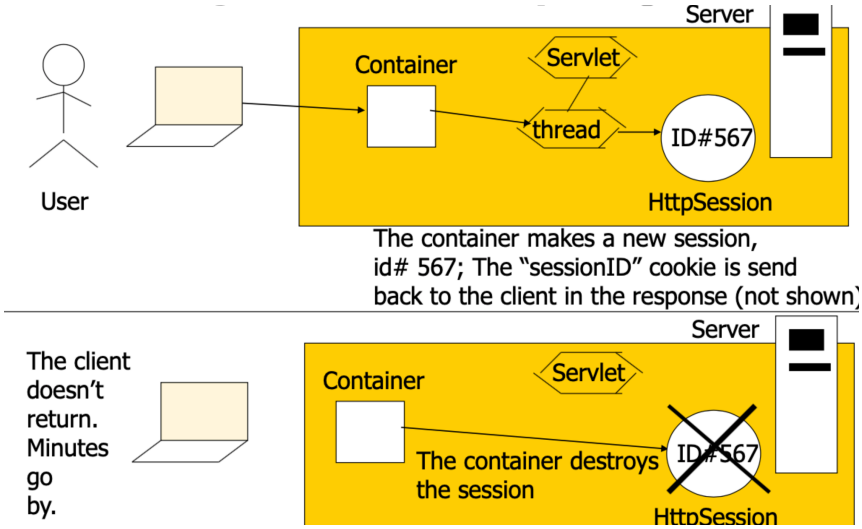
In this case, the client is the browser, not the user. The browser makes the new call on the users behalf. The user sees the new URL in the browser.

2.5 Session Tracking

- Personalization
- Privacy invasion
- HTTP-stateless protocol
  - does not support persistent information

- Track clients individually
  - Cookies
  - Session tracking
  - Hidden type input
  - URL rewriting

Session tracking with HttpSession



Three ways a session can die

1. It times out

Configuring session timeout in the Deployment Descriptor

```
<servlet>
    ...
</servlet>
<session-config>
    <session-timeout>15</session-timeout>
<session-config>
```

Setting session timeout for a specific session

```
session.setMaxInactiveInterval(20*60);
```

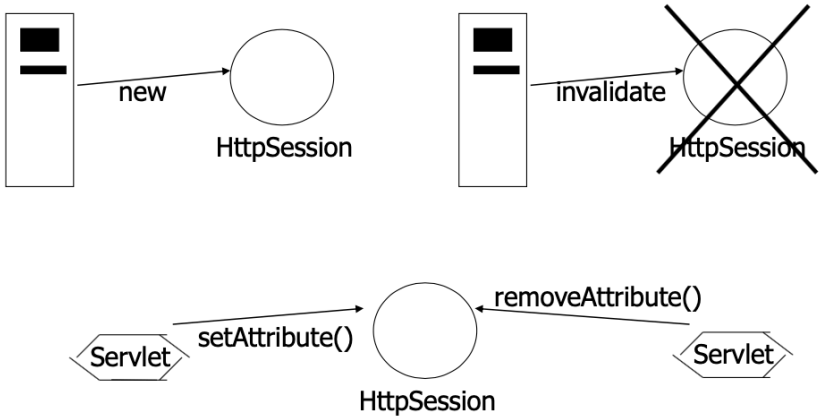
2. You can invalidate() on the session object

```
session.invalidate();
```

3. The application goes down

crashes or is undeployed

Important moments in a HttpSession object's life



Example: Servlet SessionServlet

- Uses HttpSession objects
- Handles both get and post requests

sessionSelectLanguage.html document for selecting a programming language and posting the data to the SessionServlet.

sessionSelectLanguage.html

```
<body>
    <form action="sessions" method="post" >
        Select a programming language: <br><br>
        <input type="radio" name="language" value="C++"/>C++ <br>
        <input type="radio" name="language" value="C#"/> C# <br>
        <input type="radio" name="language" value="Cobol" /> Cobol <br
>
        <!--this radio button checked by default -->
        <input type="radio" name="language" value="Java" checked="chec
ked"/> Java <br> <br>
        <input type="submit" value="Submit" />
    </form>
</body>
```

SessionServlet.java

```
package servlet;
import javax.servlet.*; import javax.servlet.http.*;
import java.io.*; import java.util.*;

@WebServlet("/sessions")
public classSessionServletextendsHttpServlet
{
    private final Map<String,String>books = new HashMap<>();
    // initialize Map books
    public voidinit()
    {
        books.put( "C++", "0130895725");
        books.put( "C#", "0130895717");
        books.put( “Cobol", "0130125075");
        books.put(“Java", "0134569555");
    }

    // receive language selection and create HttpSession object
    // containing recommended book for the client
```

```
    protected voiddoPost( HttpServletRequest request,HttpServletRespon
se response) throwsServletException, IOException
    {
        String language = request.getParameter("language")

        // Get the user's session object.
        // Uses method getSession of interface HttpServletRequest
        // Create a session (true) if one does not exist.
        HttpSession session = request.getSession( true );

        // add a value for user's choice to session
        session.setAttribute( language, books.get( language ) );

        response.setContentType( "text/html");

        try (PrintWriterout = response.getWriter())
        {
            // send HTML page to client
            out.println("<!DOCTYPE html>");
            out.println( "<html>" );

            // head section of document
            out.println("<head>" );
            out.println( "<title>Welcome to Sessions</title>" );
            out.println( "</head>" );

            // body section of document
            out.println( "<body>" );
            out.println( "<p>Welcome to Sessions! You selected "+langu
age + ".</p>" );

            // display information about the session
            out.println( "<p>Your unique session ID is: "+ session.get
Id()+ "<br>" );

            out.println( "This " + ( session.isNew()? "is": "is not")
+" a new session<br>");

            out.println("The session was created at: " + newDate( sess
ion.getCreationTime()) + "<br>" );
            out.println( "You last accessed the session at: "+ newDate
( session.getLastAccessedTime()) + "<br>");
            out.println( "The maximum inactive interval is: "+ session
.getMaxInactiveInterval()+ " seconds</p>");
            out.println( "<p><a href = "+"\"sessionSelectLanguage.html
\">"+"Click here to choose another language</a></p>");
            out.println("<p><a href = \"sessions\">"+"Click here to ge
t book recommendations</a></p>" );
```

```
            out.println( "</body>");
            // end HTML documentout.println( "</html>");
        }
    }

 // read session attributes and create HTML document
 // containing recommended books
 protected void doGet( HttpServletRequest request,HttpServletResponse
response )
 throwsServletException, IOException
 {
        // Get the user's session object.
        // Do not create a session (false) if one does not exist.
        HttpSession session = request.getSession( false );

        // get names of session object's values
        Enumeration<String> valueNames;

        if( session != null)
            valueNames = session.getAttributeNames();
        else
            valueNames = null;

        try (PrintWriter out = response.getWriter())
        {
            response.setContentType( "text/html");

            // start HTML document
            out.println("<!DOCTYPE html>");
            out.println( "<html>" );

            // head section of documen
            tout.println("<head>");
            out.println("<title>Recommendations</title>");
            out.println( "</head>" );

            // body section of document
            out.println( "<body>");
            if( valueNames != null&& valueNames.hasMoreElements() )
            {
                out.println( "<h1>Recommendations</h1>" );
                out.println( "<p>");
                String name, value;

                // get value for each name in valueNames
                while( valueNames.hasMoreElements() )
                {
                    name = valueNames.nextElement();
```

```
                    value = (String) session.getAttribute( name );
                    out.println( name + " How to Program. " +"ISBN#: "
+ value +"<br >");
                }//end-while
                out.println( "</p>");
            }//end-if
            else
            {
                out.println( "<h1>No Recommendations</h1>");
                out.println( "<p>You did not select a language.</p>");
            }
            out.println( "</body>" );

            // end HTML document
            out.println( "</html>");
        }
    }
}
```