SERVLET BASICS

Giảng viên: Nguyễn Hoàng Anh

Email: nhanh@fit.hcmus.edu.vn

Bài giảng được trích từ

- http://www.javapassion.com/j2ee/
- http://courses.coreservlets.com/Course-Materials/csajsp2.html

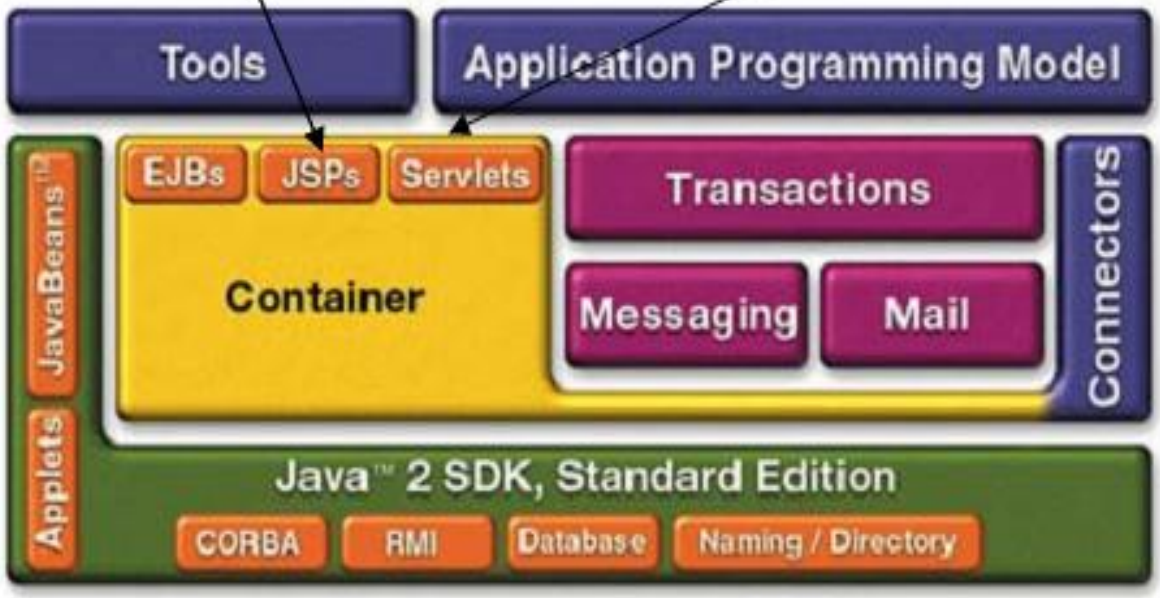
Contents

1. Servlet
2. Servlet Request & Response
3. Servlet Life Cycle
4. Scope Object

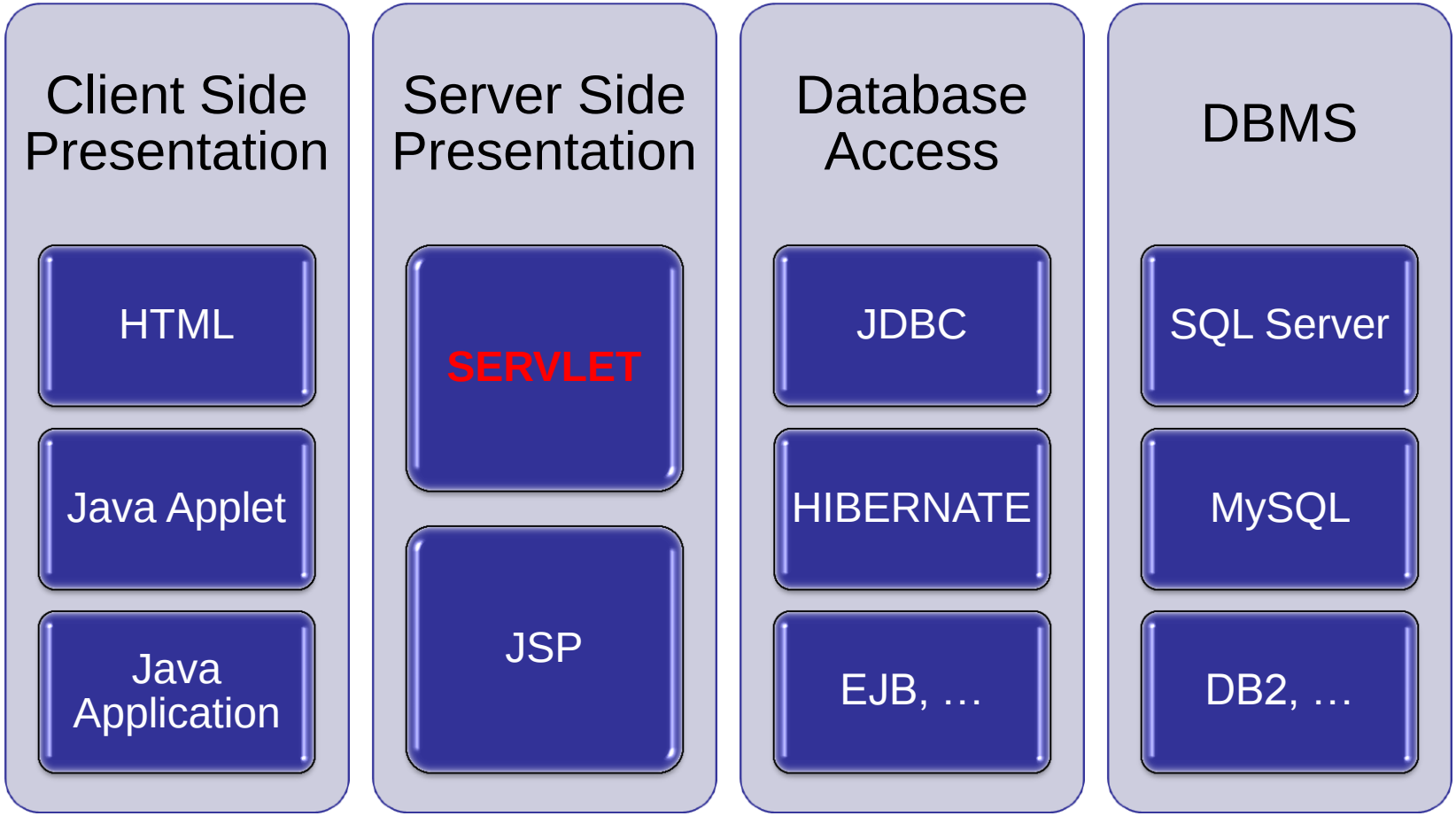
J2EE Architecture (1.2)

An extensible Web technology that uses template data, custom elements, scripting languages, and server-side Java objects to **return dynamic content to a client**. Typically the template data is HTML or XML elements. The client is often a **Web browser**.

Java Servlet A Java program that extends the functionality of a Web server, generating dynamic content and interacting with Web clients using a **request-response paradigm**.



Servlet

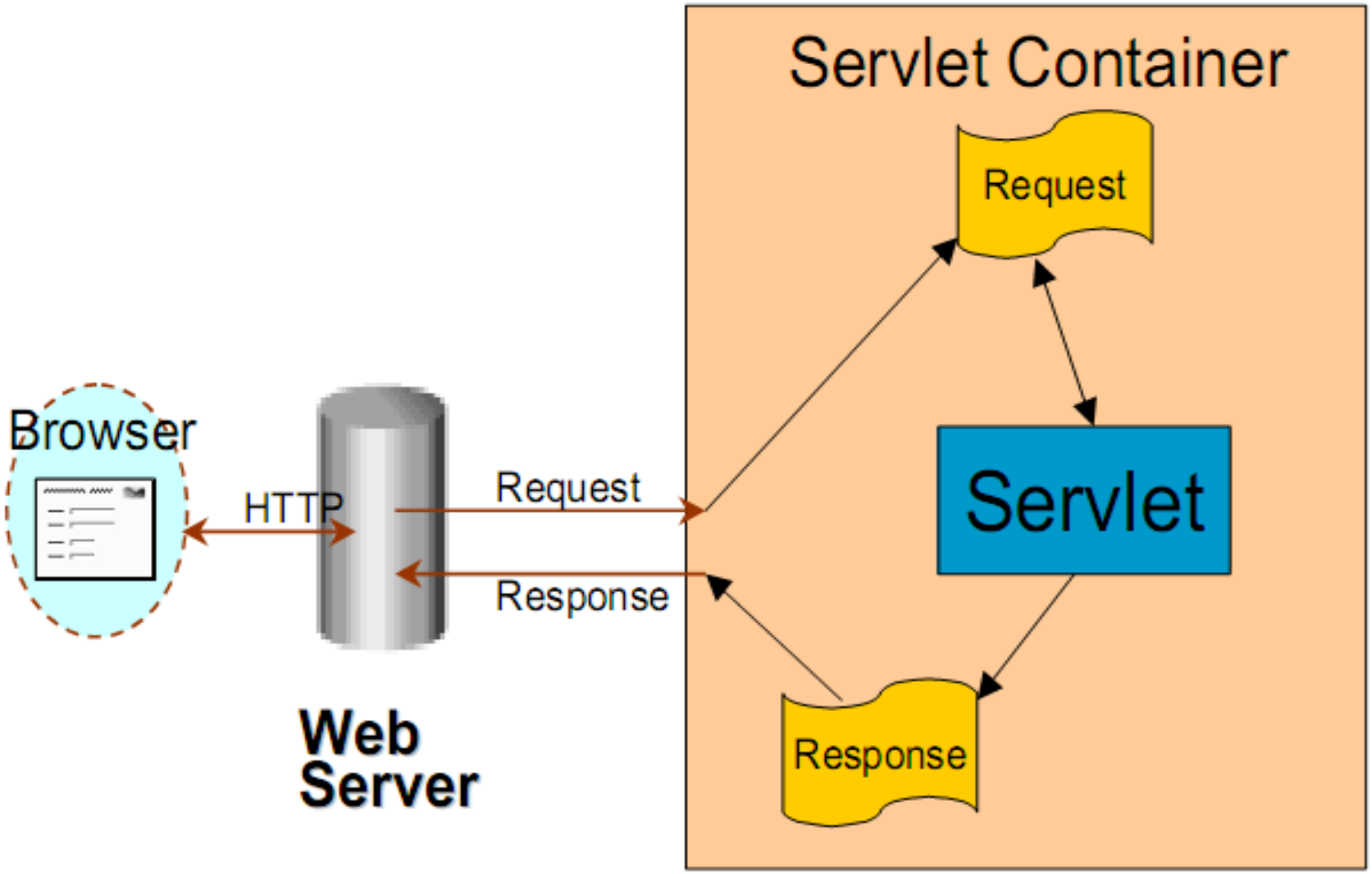

Client Side Presentation
HTML
Java Applet
Java Application
Server Side Presentation
SERVLET
JSP
Database Access
JDBC
HIBERNATE
EJB, …
DBMS
SQL Server
MySQL
DB2, …

Servlet Code

```
Public class HelloServlet extends HttpServlet {

  public void doGet(HttpServletRequest request,
                    HttpServletResponse
response){
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<title>Hello World!</title>");
  }
  ...
}
```

Servlet Request & Response



Request & Response

- What is a request?
  - Information that is sent from client to a server
    - Who made the request
    - What user-entered data is sent
    - Which HTTP headers are sent
- What is a response?
  - Information that is sent to client from a server
    - Text(html, plain) or binary(image) data
    - HTTP headers, cookies, etc

HTTP GET & HTTP POST

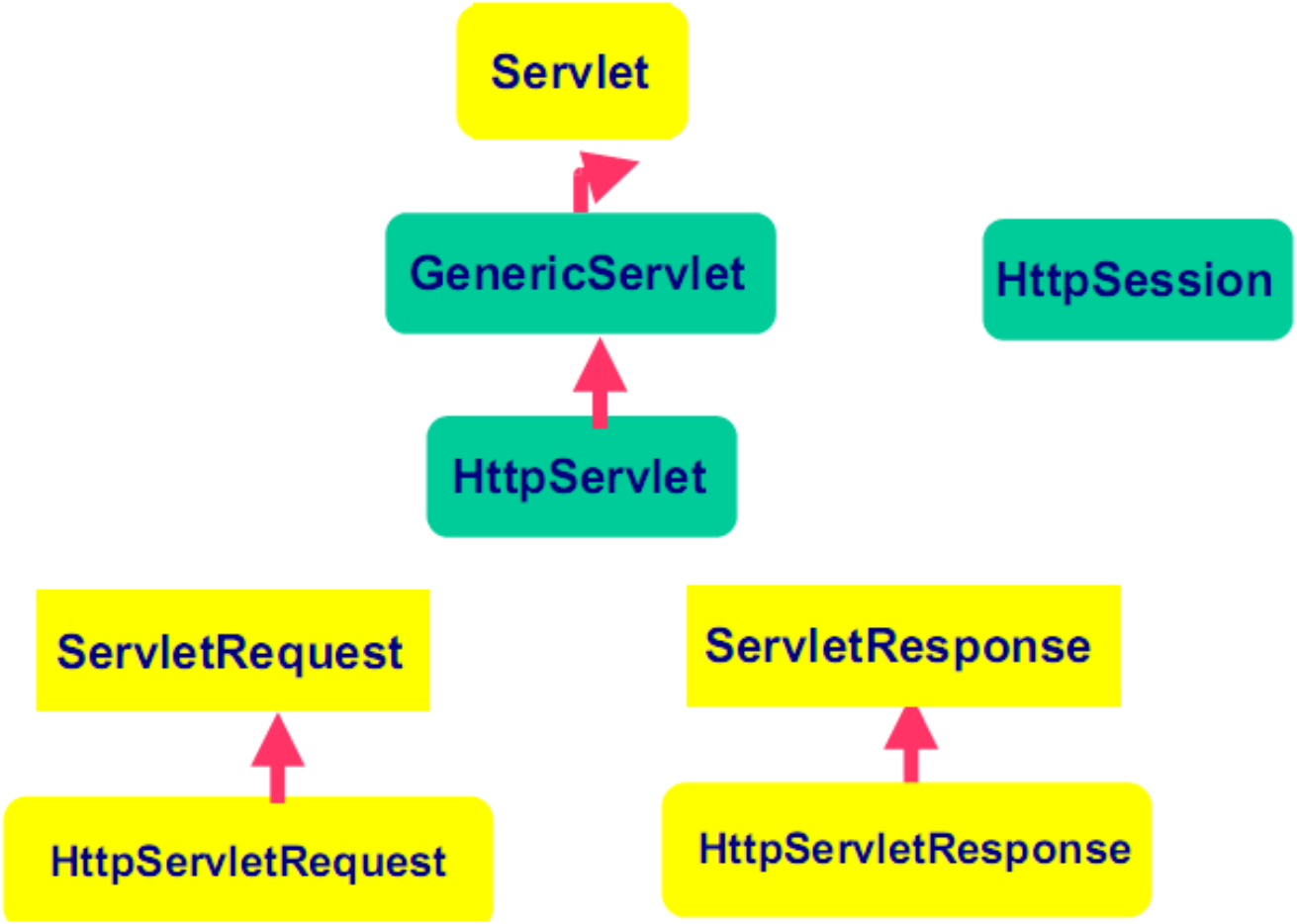
- The most common client requests
  - HTTP GET & HTTP POST
- GET requests:
  - User entered information is appended to the URL in a query string
  - Can only send limited amount of data
    - .../servlet/ViewCourse?FirstName=Sang&LastName=Shin
- POST requests:
  - User entered information is sent as data (not appended to URL)
  - Can send any amount of data

First Servlet

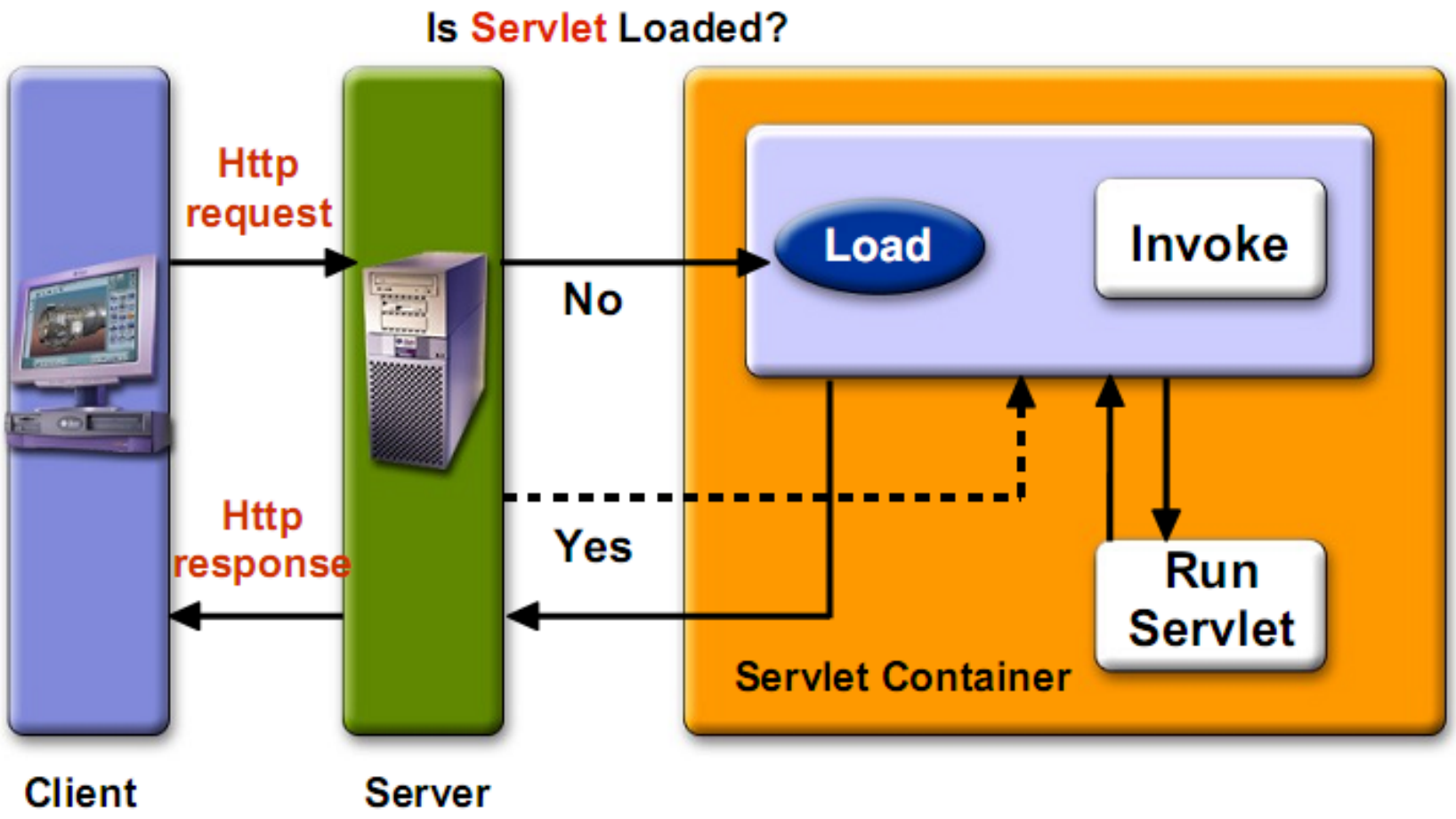
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

Public class HelloServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
          throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<title>First Servlet</title>");
    out.println("<big>Hello Code Camp!</big>");
  }
}
```

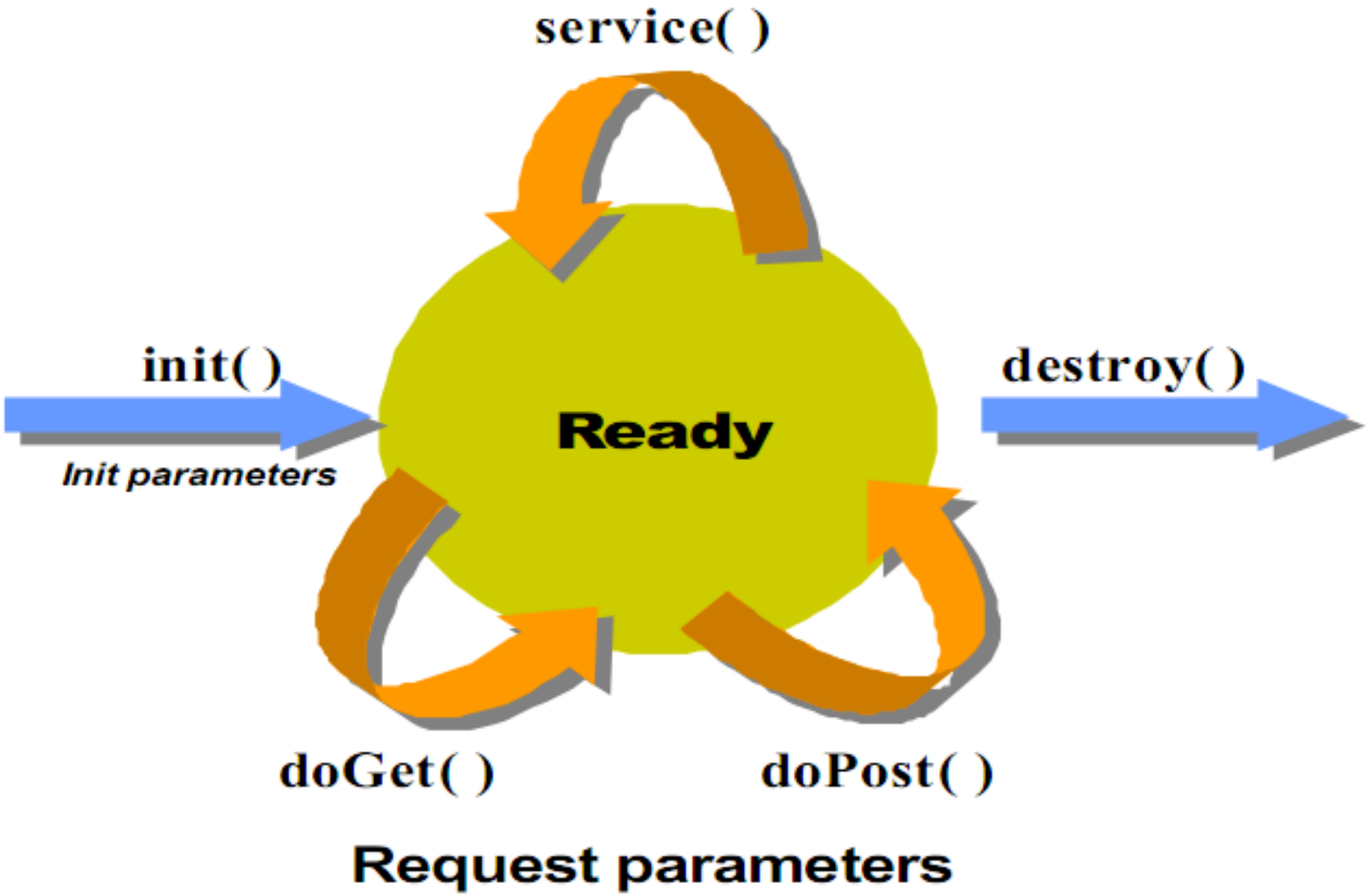
Servlet Interfaces & Classes



Servlet Life-Cycle



Servlet Life-Cycle Methods



Servlet Life-Cycle Methods

- Invoked by container
  - Container controls life cycle of a servlet
- Defined in
  - javax.servlet.GenericServlet class or
    - init()
    - destroy()
    - service() - this is an abstract method
  - javax.servlet.http.HttpServlet class
    - doGet(), doPost(), doXxx()
    - service() - implementation

Servlet Life-Cycle Methods

- init()
  - Invoked once when the servlet is first instantiated
  - Perform any set-up in this method
    - Setting up a database connection
- destroy()
  - Invoked before servlet instance is removed
  - Perform any clean-up
    - Closing a previously created database connection

init() from CatalogServlet.java

```
public class CatalogServlet extends HttpServlet {
  private BookDB bookDB;

  // Perform any one-time operation for the servlet,
  // like getting database connection object.

  // Note: In this example, database connection object is assumed
  // to be created via other means (via life cycle event mechanism)
  // and saved in ServletContext object. This is to share a same
  // database connection object among multiple servlets.
  public void init() throws ServletException {
    bookDB = (BookDB)getServletContext().
                     getAttribute("bookDB");
    if (bookDB == null) throw new
      UnavailableException("Couldn't get database.");
  }
```

init() reading Configuration parameters

```
public void init(ServletConfig config) throws
   ServletException {
        super.init(config);
        String driver = getInitParameter("driver");
        String fURL = getInitParameter("url");
        try {
          openDBConnection(driver, fURL);
        } catch (SQLException e) {
           e.printStackTrace();
        } catch (ClassNotFoundException e){
           e.printStackTrace();
        }
 }
```

Setting Init Parameters in web.xml

```
<web-app>
    <servlet>
        <servlet-name>chart</servlet-name>
        <servlet-class>ChartServlet</servlet-class>
        <init-param>
            <param-name>driver</param-name>
            <param-value>
              COM.cloudscape.core.RmiJdbcDriver
            </param-value>
        </init-param>

        <init-param>
            <param-name>url</param-name>
            <param-value>
              jdbc:cloudscape:rmi:CloudscapeDB
            </param-value>
        </init-param>
    </servlet>
</web-app>
```

destroy

```
public class CatalogServlet extends HttpServlet {
  private BookDB bookDB;

  public void init() throws ServletException {
    bookDB = (BookDB)getServletContext().
                     getAttribute("bookDB");
    if (bookDB == null) throw new
      UnavailableException("Couldn't get database.");
  }
  public void destroy() {
         bookDB = null;
  }
  ...
```

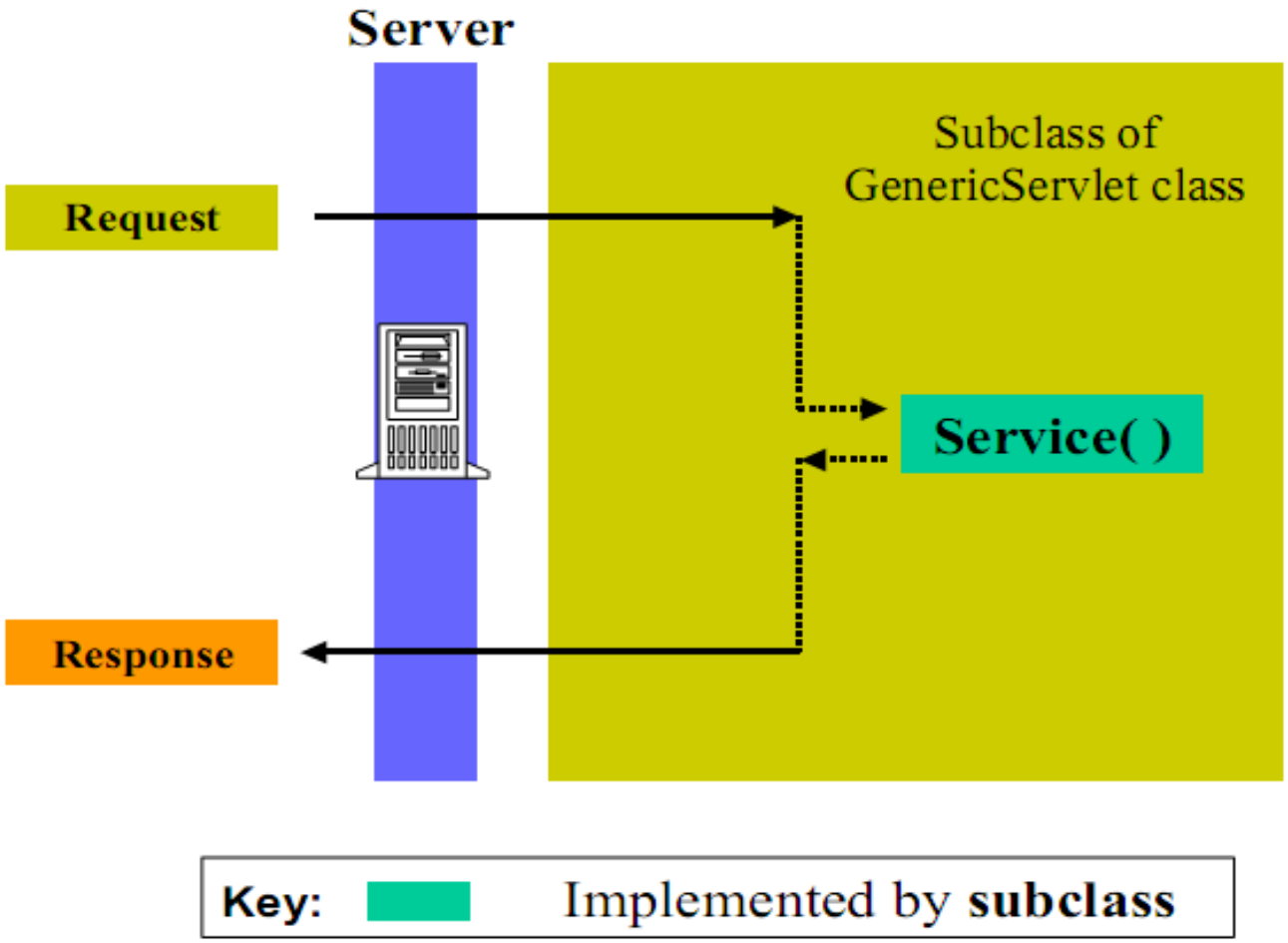
Servlet Life-Cycle Methods

- service() javax.servlet.GenericServlet class
  - Abstract method
- service() in javax.servlet.http.HttpServlet class
  - Concrete method (implementation)
  - Dispatches to doGet(), doPost(), etc
  - Do not override this method!
- doGet(), doPost(), doXxx() in in javax.servlet.http.HttpServlet
  - Handles HTTP GET, POST, etc. requests
  - Override these methods in your servlet to provide desired behavior

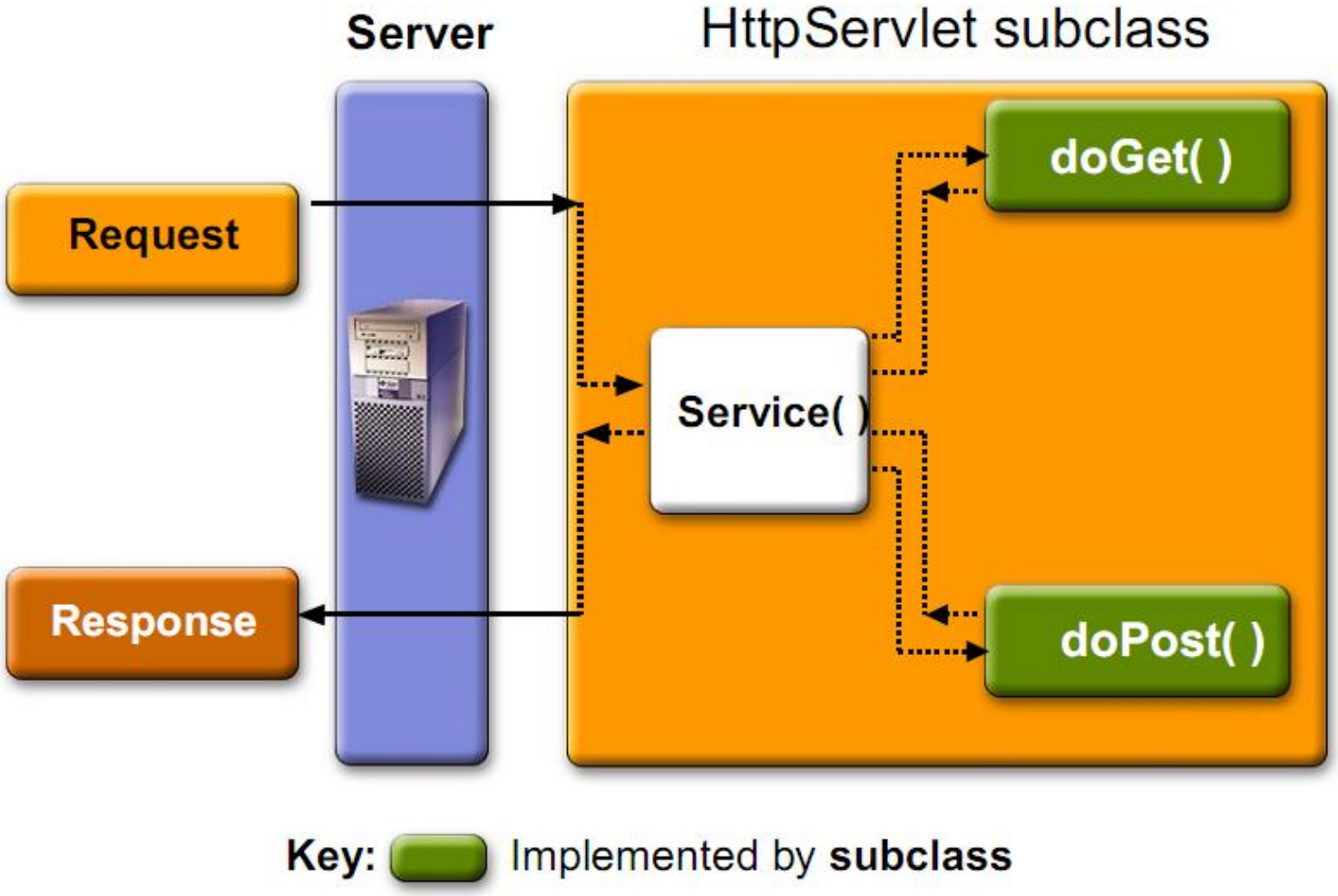
service & doGet , doPost

- service() methods take generic requests and responses:
  - service(ServletRequest request, ServletResponse response)
- doGet() or doPost() take HTTP requests and responses:
  - doGet(HttpServletRequest request, HttpServletResponse response)
  - doPost(HttpServletRequest request, HttpServletResponse response)

Service()


Server
Subclass of
GenericServlet class
Request
Service( )
Response
Key:
Implemented by subclass

doGet() & doPost()



doGet()

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

Public class HelloServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                 throws ServletException, IOException {

    // Just send back a simple HTTP response
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<title>First Servlet</title>");
    out.println("<big>Hello J2EE Programmers! </big>");
  }
}
```

doGet()

```
public void doGet (HttpServletRequest request,
                   HttpServletResponse response)
        throws ServletException, IOException {

        // Read session-scope attribute "message"
        HttpSession session = request.getSession(true);
        ResourceBundle messages = (ResourceBundle)session.getAttribute("messages")

        // Set headers and buffer size before accessing the Writer
        response.setContentType("text/html");
        response.setBufferSize(8192);
        PrintWriter out = response.getWriter();

        // Then write the response (Populate the header part of the response)
        out.println("<html>" +
                    "<head><title>" + messages.getString("TitleBookDescription") +
                    "</title></head>");

        // Get the dispatcher; it gets the banner to the user
        RequestDispatcher dispatcher =
               getServletContext().getRequestDispatcher("/banner");

        if (dispatcher != null)
               dispatcher.include(request, response);
```

doGet()

```
    // Get the identifier of the book to display (Get HTTP parameter)
    String bookId = request.getParameter("bookId");
    if (bookId != null) {

        // and the information about the book (Perform business logic)
        try {
            BookDetails bd = bookDB.getBookDetails(bookId);
            Currency c = (Currency)session.getAttribute("currency");
            if (c == null) {
                c = new Currency();
                c.setLocale(request.getLocale());
                session.setAttribute("currency", c);
            }
            c.setAmount(bd.getPrice());

            // Print out the information obtained
            out.println("...");
        } catch (BookNotFoundException ex) {
                response.resetBuffer();
                throw new ServletException(ex);
        }

    }
    out.println("</body></html>");
    out.close();
}
```

Steps of Populating HTTP Response

- Fill Response headers
- Set some properties of the response
  - Buffer size
- Get an output stream object from the response
- Write body content to the output stream

Response

```
Public class HelloServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                 throws ServletException, IOException {

    // Fill response headers
    response.setContentType("text/html");
    // Set buffer size
    response.setBufferSize(8192);
    // Get an output stream object from the response
    PrintWriter out = response.getWriter();
    // Write body content to output stream
    out.println("<title>First Servlet</title>");
    out.println("<big>Hello J2EE Programmers! </big>");
  }
}
```

Scope Objects

- Enables sharing information among collaborating web components via attributes maintained in Scope objects
  - Attributes are name/object pairs
- Attributes maintained in the Scope objects are accessed with
  - getAttribute() & setAttribute()
- 4 Scope objects are defined
  - Web context, session, request, page

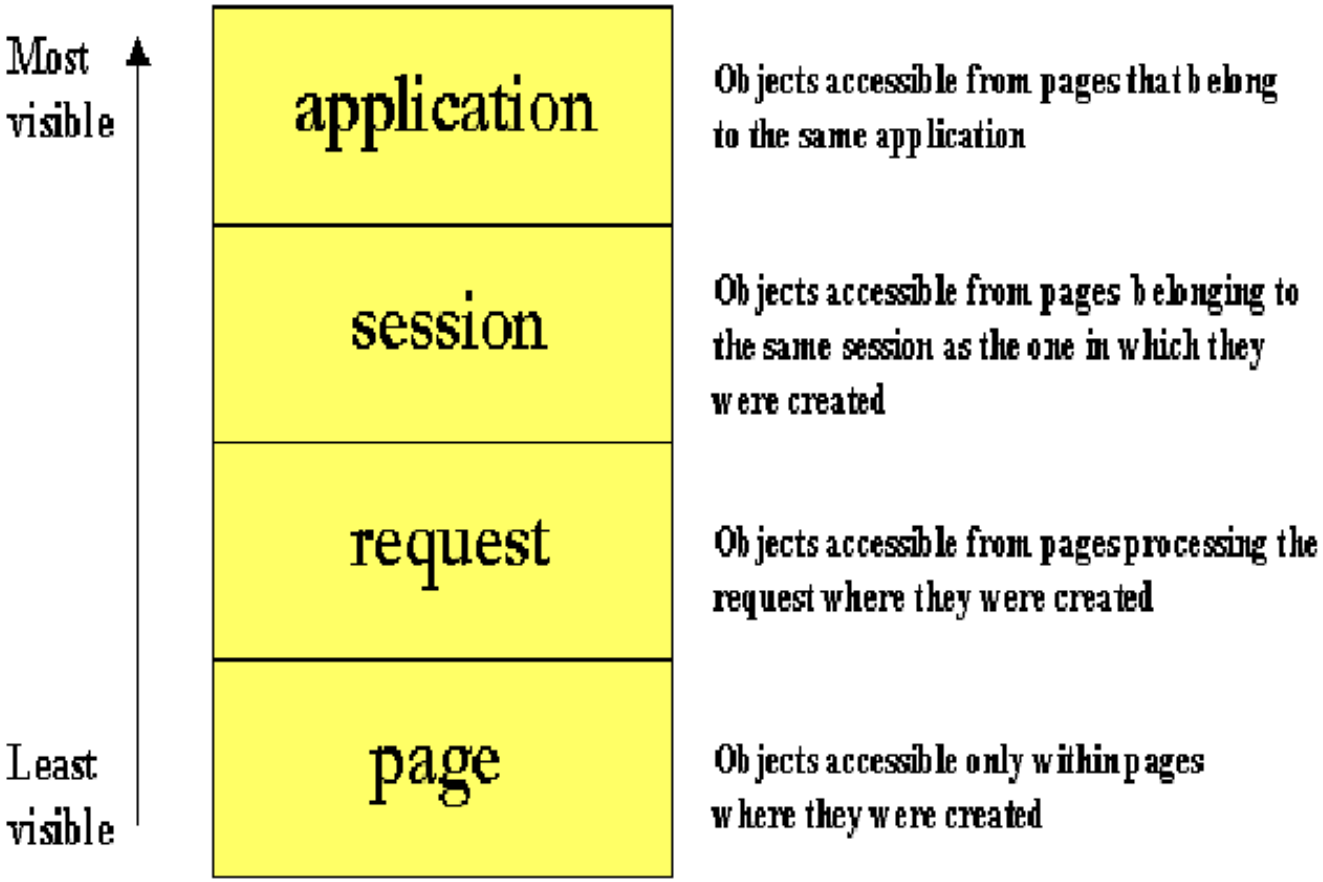
Four Scope Objects: Accessibility

- Web context (ServletConext)
  - Accessible from Web components within a Web context
- Session
  - Accessible from Web components handling a request that belongs to the session
- Request
  - Accessible from Web components handling the request
- Page
  - Accessible from JSP page that creates the object

Four Scope Objects: Class

- Web context
  - javax.servlet.ServletContext
- Session
  - javax.servlet.http.HttpSession
- Request
  - subtype of javax.servlet.ServletRequest: javax.servlet.http.HttpServletRequest
- Page
  - javax.servlet.jsp.PageContext

Scope Objects



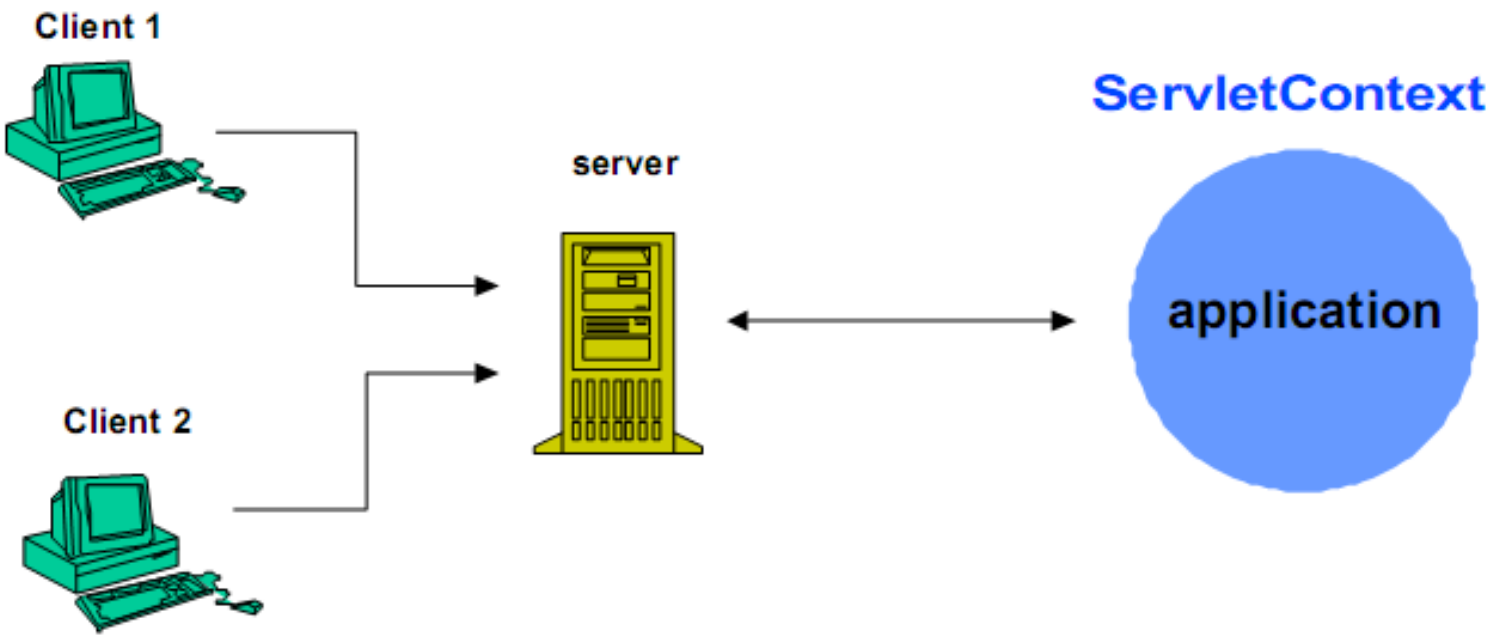
What is ServletContext For?

- Used by servets to
  - Set and get context-wide (application-wide) object-valued attributes
  - Get request dispatcher
    - To forward to or include web component
  - Access Web context-wide initialization parameters set in the web.xml file
  - Access Web resources associated with the Web context
  - Log
  - Access other misc. information

Scope of ServletContext

- Context-wide scope
  - Shared by all servlets and JSP pages within a "web application"
    - Why it is called “web application scope”
  - A "web application" is a collection of servlets and content installed under a specific subset of the server's URL namespace and possibly installed via a *.war file
    - All servlets in BookStore web application share same ServletContext object
  - There is one ServletContext object per "web application" per Java Virtual Machine

ServletContext: Web Application Scope



How to Access ServletContext Object?

- Within your servlet code, call getServletContext()
- Within your servlet filter code, call getServletContext()
- The ServletContext is contained in ServletConfig object, which the Web server provides to a servlet when the servlet is initialized
  - init (ServletConfig servletConfig) in Servlet interface

Example: Getting Attribute Value from ServletContext

```
public class CatalogServlet extends HttpServlet {
  private BookDB bookDB;
  public void init() throws ServletException {
    // Get context-wide attribute value from
    // ServletContext object
    bookDB = (BookDB)getServletContext().
                     getAttribute("bookDB");
    if (bookDB == null) throw new
      UnavailableException("Couldn't get database.");
  }
}
```

Example: Getting and Using RequestDispatcher Object

```
public void doGet (HttpServletRequest request,
                   HttpServletResponse response)
        throws ServletException, IOException {

        HttpSession session = request.getSession(true);
             ResourceBundle messages = (ResourceBundle)session.getAttribute("messages");

        // set headers and buffer size before accessing the Writer
        response.setContentType("text/html");
            response.setBufferSize(8192);
            PrintWriter out = response.getWriter();

        // then write the response
        out.println("<html>" +
                    "<head><title>" + messages.getString("TitleBookDescription") +
                    "</title></head>");

        // Get the dispatcher; it gets the banner to the user
        RequestDispatcher dispatcher =
               session.getServletContext().getRequestDispatcher("/banner");

        if (dispatcher != null)
               dispatcher.include(request, response);
```

Example: Logging

```
public void doGet (HttpServletRequest request,
                   HttpServletResponse response)
        throws ServletException, IOException {

        ...
        getServletContext().log("Life is good!");
        ...
        getServletContext().log("Life is bad!", someException);
```

Why HttpSession?

- Need a mechanism to maintain client state across a series of requests from a same user (or originating from the same browser) over some period of time
  - Example: Online shopping cart
- Yet, HTTP is stateless
- HttpSession maintains client state
  - Used by Servlets to set and get the values of session scope attributes

How to Get HttpSession?

- via getSession() method of a Request object (HttpServletRequest)

Example: HttpSession

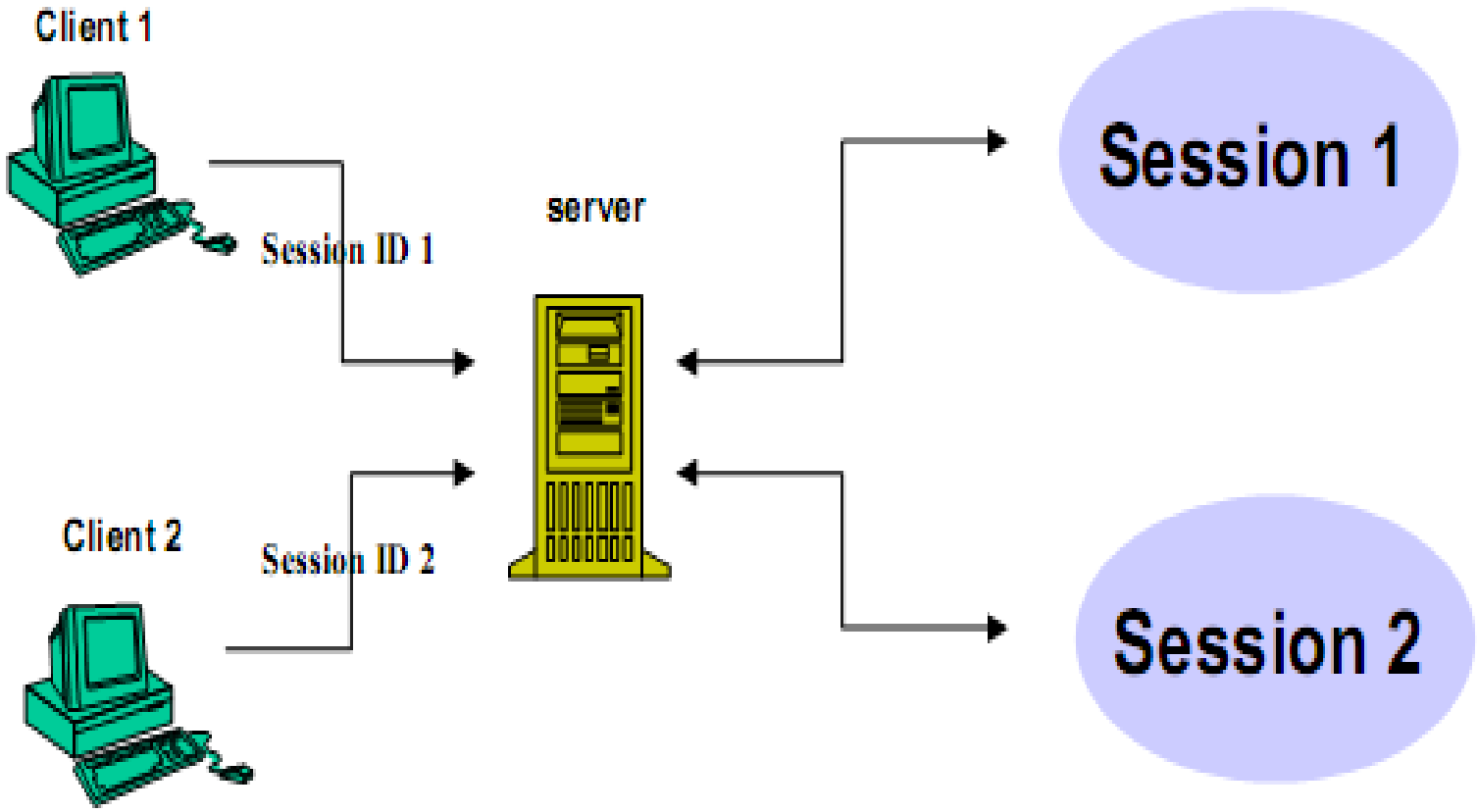
```
public class CashierServlet extends HttpServlet {
  public void doGet (HttpServletRequest request,
                     HttpServletResponse response)
              throws ServletException, IOException {

    // Get the user's session and shopping cart
    HttpSession session = request.getSession();
    ShoppingCart cart =
      (ShoppingCart)session.getAttribute("cart");
    ...
    // Determine the total price of the user's books
    double total = cart.getTotal();
```

What is Servlet Request?

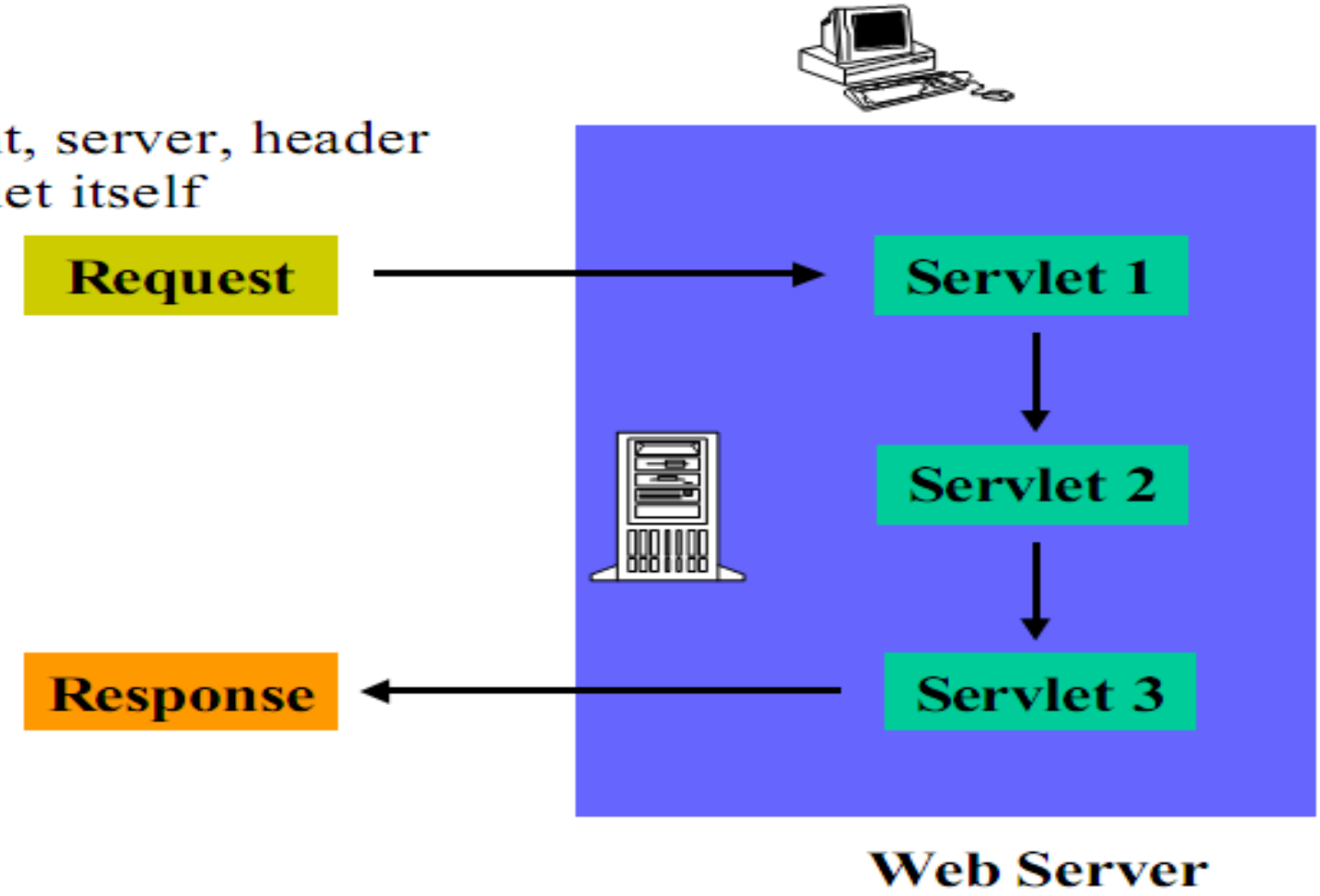
- Contains data passed from client to servlet
- All servlet requests implement ServletRequest interface which defines methods for accessing
  - Client sent parameters
  - Object-valued attributes
  - Locales
  - Client and server
  - Input stream
  - Protocol information
  - Content type
  - If request is made over secure channel (HTTPS)

Session


Client 1
Session ID 1
server
Session 1
Client 2
Session ID 2
Session 2

Request

data,
client, server, header
servlet itself



Getting Client Sent Parameters

- A request can come with any number of parameters
- Parameters are sent from HTML forms:
  - GET: as a query string, appended to a URL
  - POST: as encoded POST data, not appeared in the URL
- getParameter("paraName")
  - Returns the value of paraName
  - Returns null if no such parameter is present
  - Works identically for GET and POST requests

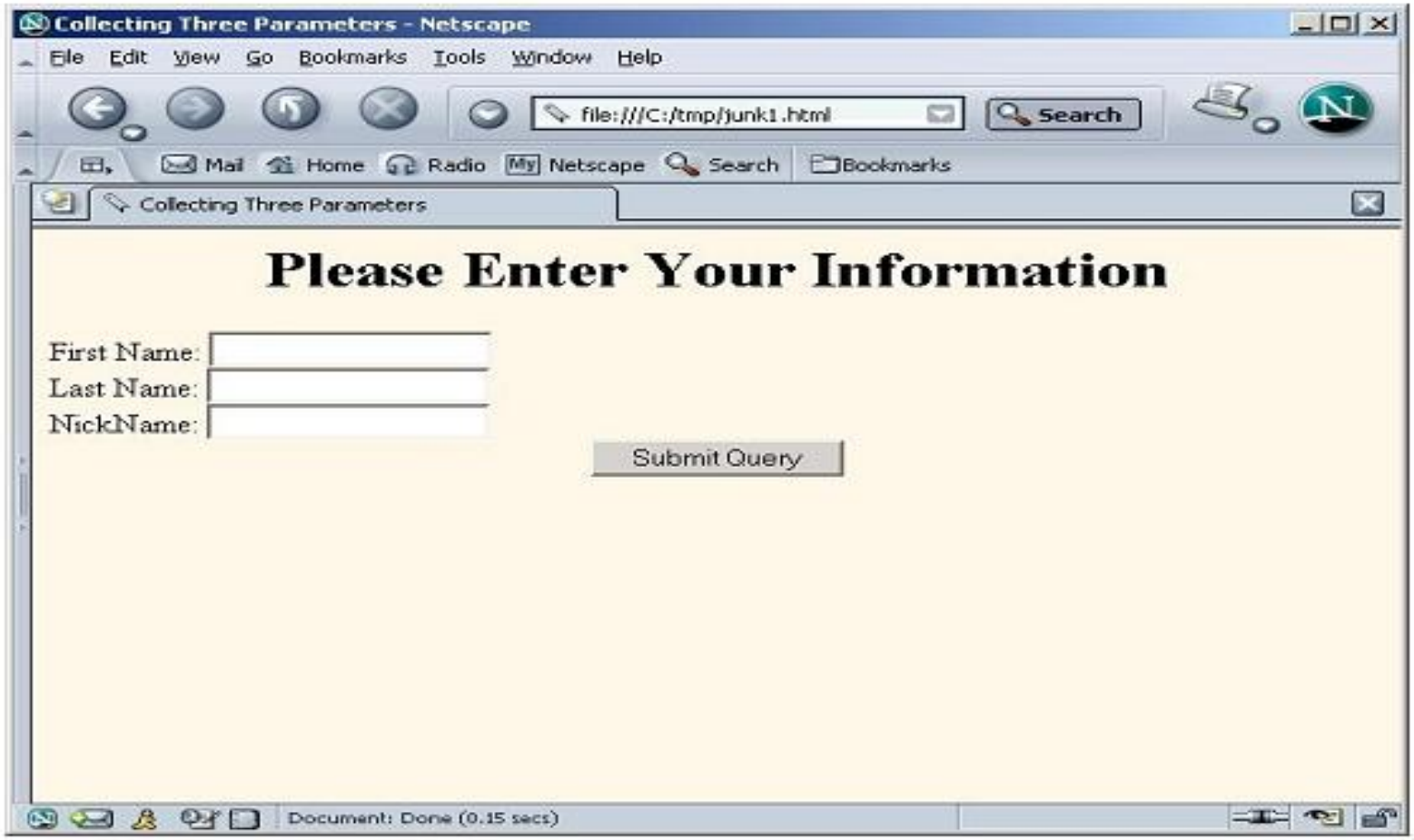
A Sample FORM using GET

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
 <HTML>
 <HEAD>
   <TITLE>Collecting Three Parameters</TITLE>
 </HEAD>
 <BODY BGCOLOR="#FDF5E6">
 <H1 ALIGN="CENTER">Please Enter Your Information</H1>

 <FORM ACTION="/sample/servlet/ThreeParams">
   First Name:  <INPUT TYPE="TEXT" NAME="param1"><BR>
   Last Name:   <INPUT TYPE="TEXT" NAME="param2"><BR>
   Class Name:  <INPUT TYPE="TEXT" NAME="param3"><BR>
   <CENTER>
     <INPUT TYPE="SUBMIT">
   </CENTER>
 </FORM>

 </BODY>
 </HTML>
```

A Sample FORM using GET



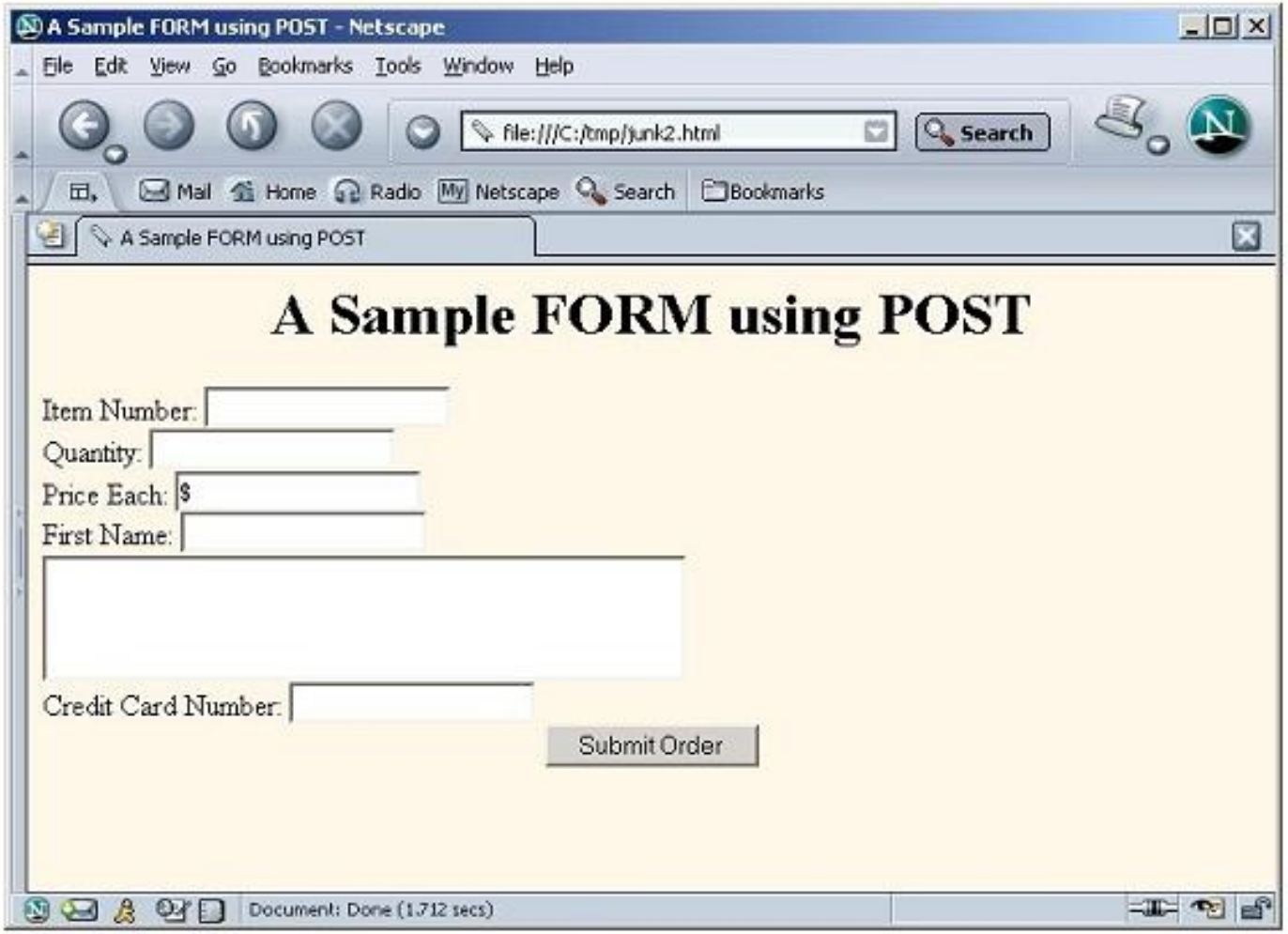
A FORM Based Servlet: Get

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
/** Simple servlet that reads three parameters from the html form */
public class ThreeParams extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Your Information";
    out.println("<HTML>" +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
                "<UL>\n" +
                "  <LI><B>First Name in Response</B>: "
                + request.getParameter("param1") + "\n" +
                "  <LI><B>Last Name in Response</B>: "
                + request.getParameter("param2") + "\n" +
                "  <LI><B>NickName in Response</B>: "
                + request.getParameter("param3") + "\n" +
                "</UL>\n" +
                "</BODY></HTML>");
  }
}
```

A Sample FORM using POST

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>A Sample FORM using POST</TITLE>
</HEAD>
<BODY BGCOLOR="#FDF5E6">
<H1 ALIGN="CENTER">A Sample FORM using POST</H1>
<FORM ACTION="/sample/servlet/ShowParameters" METHOD="POST">
  Item Number: <INPUT TYPE="TEXT" NAME="itemNum"><BR>
  Quantity: <INPUT TYPE="TEXT" NAME="quantity"><BR>
  Price Each: <INPUT TYPE="TEXT" NAME="price" VALUE="$"><BR>
  First Name: <INPUT TYPE="TEXT" NAME="firstName"><BR>
  <TEXTAREA NAME="address" ROWS=3 COLS=40></TEXTAREA><BR>
  Credit Card Number:
  <INPUT TYPE="PASSWORD" NAME="cardNum"><BR>
  <CENTER>
    <INPUT TYPE="SUBMIT" VALUE="Submit Order">
  </CENTER>
</FORM>
</BODY>
</HTML>
```

A Sample FORM using POST



A Form Based Servlet: POST

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

public class ShowParameters extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                    throws ServletException, IOException {
     ...
  }
  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
                    throws ServletException, IOException {

    doGet(request, response);

  }

}
```

What is HTTP Servlet Request?

- Contains data passed from HTTP client to HTTP servlet
- Created by servlet container and passed to servlet as a parameter of doGet() or doPost() methods
- HttpServletRequest is an extension of ServletRequest and provides additional methods for accessing
  - HTTP request URL
    - Context, servlet, path, query information
  - Misc. HTTP Request header information
  - Authentication type & User security information
  - Cookies
  - Session

HTTP Request URL

- Contains the following parts
  - http://[host]:[port]/[request path]?[query string]

HTTP Request URL:[requestpath]

- http://[host]:[port]/[request path]?[query string]
- [request path] is made of
  - Context: /<context of web app>
  - Servlet name: /<component alias>
  - Path information: the rest of it
- Examples
  - http://localhost:8080/hello1/greeting
  - http://localhost:8080/hello1/greeting.jsp
  - http://daydreamer/catalog/lawn/index.html

HTTP Request URL: [query string]

- http://[host]:[port]/[request path]?[query string]
- [query string] are composed of a set of parameters and values that are user entered
- Two ways query strings are generated
  - A query string can explicitly appear in a web page
    - Add To Cart
    - String bookId = request.getParameter("Add");
  - A query string is appended to a URL when a form with a GET HTTP method is submitted
    - http://localhost/hello1/greeting?username=Monica+Clinton
    - String userName=request.getParameter("username")

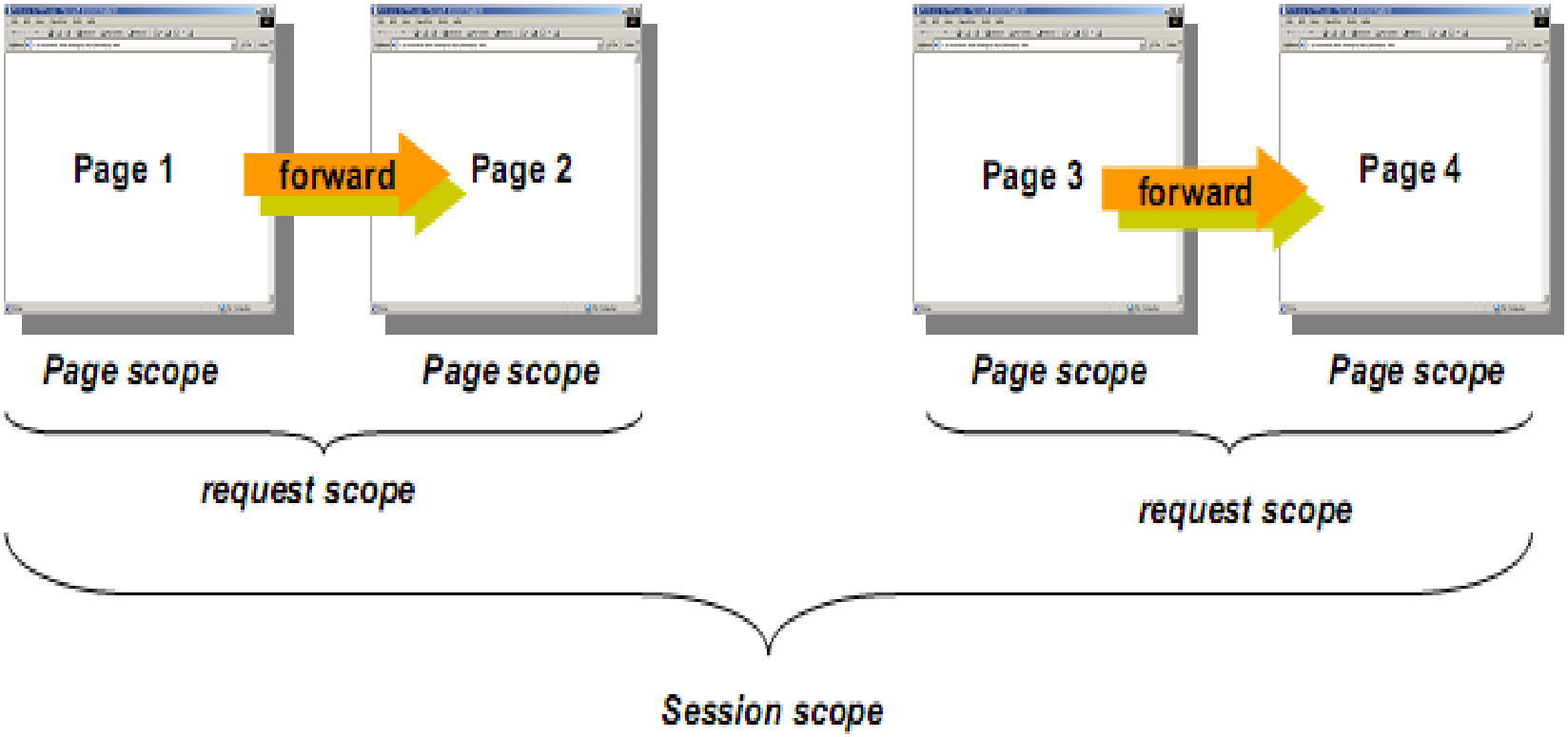
Context, Path, Query, Parameter

- String getContextPath()
- String getQueryString()
- String getPathInfo()
- String getPathTranslated()

Cookie Method (in HTTPServletRequest)

- Cookie[] getCookies()
  - an array containing all of the Cookie objects the client sent with this request

Page, Request



What is Servlet Response?

- Contains data passed from servlet to client
- All servlet responses implement ServletResponse interface
  - Retrieve an output stream
  - Indicate content type
  - Indicate whether to buffer output
  - Set localization information
- HttpServletResponse extends ServletResponse
  - HTTP response status code
  - Cookies

Responses

Request

Servlet 1

Servlet 2

Servlet 3

Response

Web Server

Response Structure:
status code , headers
and body.

Response Structure