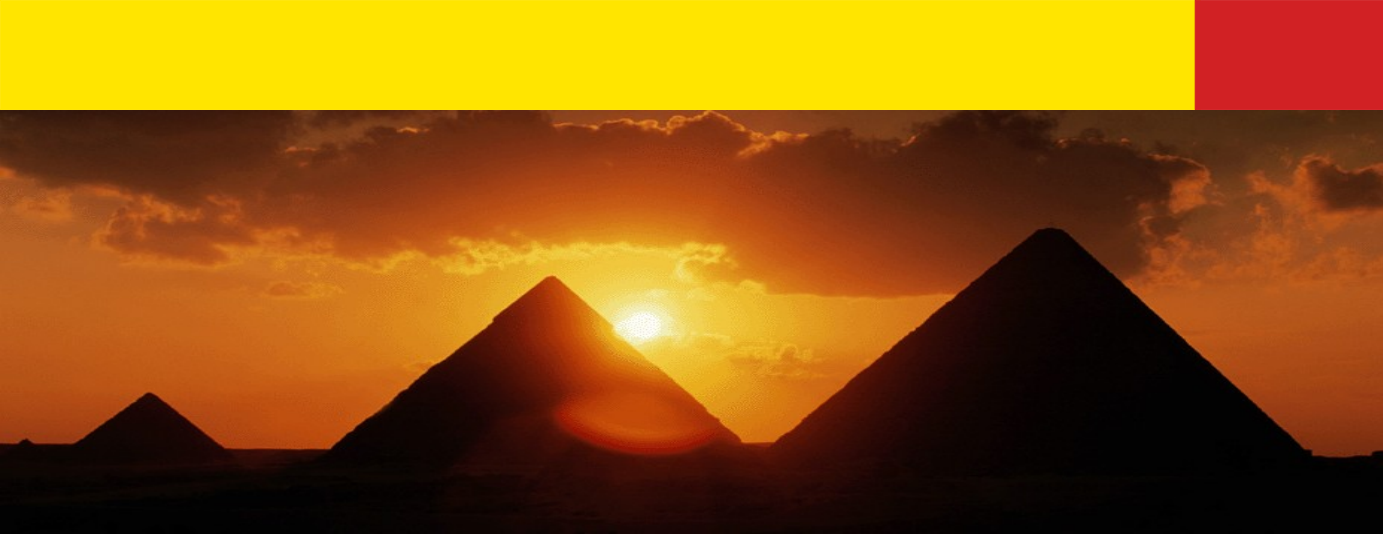




شعبان 1429

الإصدار 1.0

Servlet Basics

أساسيات Servlet

Disclaimer & Acknowledgments إخلاء مسؤولية و عرفان

- Even though Sang Shin is a full-time employee of Sun Microsystems, the contents here are created as his own personal endeavor and thus does not reflect any official stance of Sun Microsystems.
على الرغم من أن Sang Shin يعمل في شركة SUN فإن محتوى هذه الدروس تم إنجازها بمجهوده الشخصي وبالتالي لا تعبر بأي حال عن الموقف الرسمي لشركة SUN
- Sun Microsystems is not responsible for any inaccuracies in the contents. لا تتحمل شركة Sun أي مسؤولية عن الأخطاء الموجودة في محتوى هذه الدروس
- Acknowledgements عرفان
 - The slides and example code of this presentation are from “Servlet” section of Java WSDP tutorial written by [Stephanie Bodoff](#) of Sun Microsystems
محتوى هذه الشريحة و الأمثلة مقتبسة من فصل “Servlet” للكتاب “التعليم Java WSDP” من Sun Microsystems
الذي كتبه [Stephanie Bodoff](#) لصالح Sun Microsystems
 - Some slides are borrowed from “Servlet” codecamp material authored by [Doris Chen](#) of Sun Microsystems
 - Some example codes are borrowed from “Core Servlets and JavaServer Pages” book written by [Marty Hall](#)

Revision History

تواريخ المراجعات

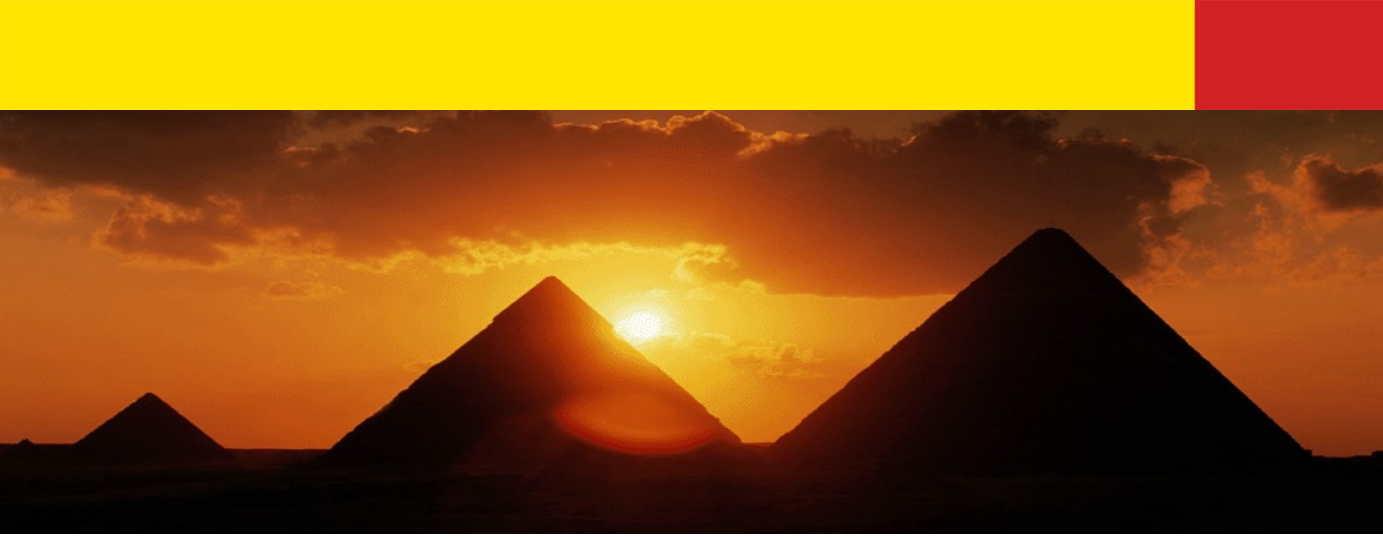
- 12/24/2002: version 1 (without speaker notes) by Sang Shin
- 01/04/2003: version 2 (with partially done speaker notes) by Sang Shin
- 01/13/2003: version 3 (screen shots of installing, configuring, running BookStore1 are added) by Sang Shin
- 04/22/2003: version 4:
 - Original Servlet presentation is divided into “Servlet Basics” and “Servlet Advanced”
 - speaker notes are added for the slides that did not have them, editing and typo checking are done via spellchecker (Sang Shin)

Topics المواضيع

- Servlet in big picture of J2EE دور Servlet البارز في J2EE
- Servlet request & response model نموذج الطلب و الإجابة لل Servlet
- Servlet life cycle دورة حياة Servlet
- Servlet scope objects نطاق كائنات Servlet
- Servlet request طلب Servlet
- Servlet response: Status, Header, Body إجابة Servlet: الحالة و الترويسة و الجسم
- Error Handling معالجة الأخطاء

Advanced Topics: المواضيع المتقدمة

- Session Tracking متابعة ال Session
- Servlet Filters مرشحات ال Servlet
- Servlet life-cycle events أحداث دورة حياة ال Servlet
- Including, forwarding to, and redirecting to other web resources تضمين و إرسال و تحويل إلى موارد أخرى للشبكة
- Concurrency Issues مسألة التزامن
- Invoker Servlet المستدعي لل Servlet



Servlet in a Big Picture of J2EE

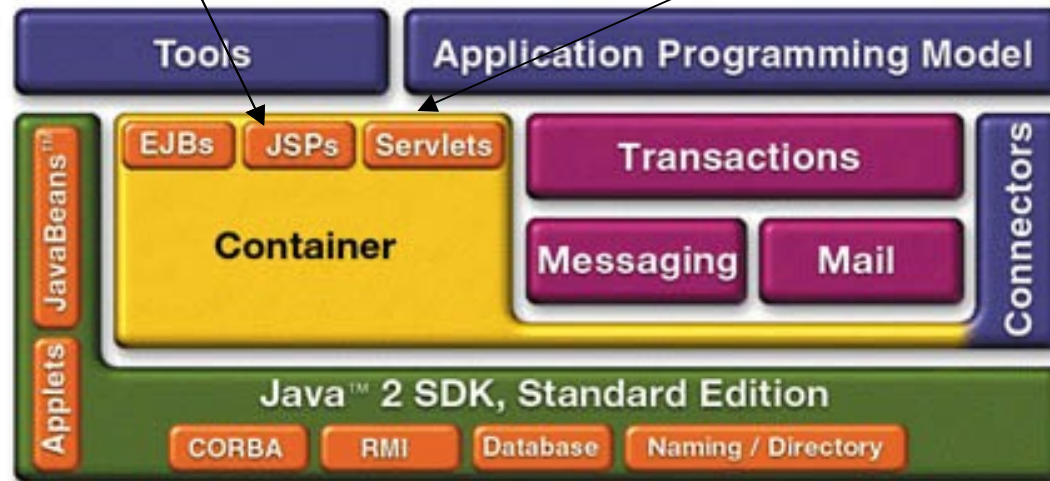
دور Servlet البارز في J2EE

J2EE 1.2 Architecture

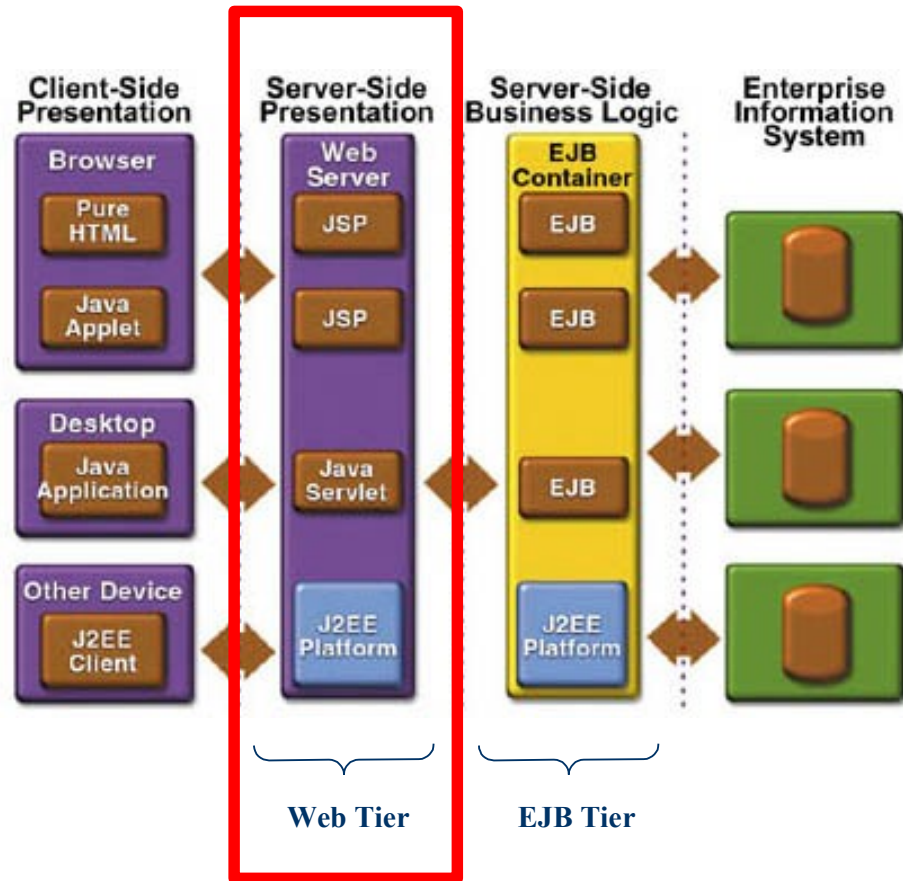
J2EE 1.2 بنية

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**.



Where are Servlet and JSP?



What is Servlet? ما هي Servlet

- Java™ objects which are based on servlet framework and APIs and extend the functionality of a HTTP server.
كائنات الجافا المبنية على منصة servlet و API و تقوم بتوسيع وظائف خادم HTTP
- Mapped to URLs and managed by container with a simple architecture
يقع تعيينها للURL كما يتم إدارتها من قبل الحاوية مع بنيتها البسيطة
- Available and running on all major web servers and app servers
متوفرة و تشغل على جميع خادמות الشبكة و خادמות التطبيقات الهامة
- Platform and server independent
إستقلالية المنصة والخادم

First Servlet Code

أول مصدر Servlet

```
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>");  
    }  
    ...  
}
```

CGI versus Servlet

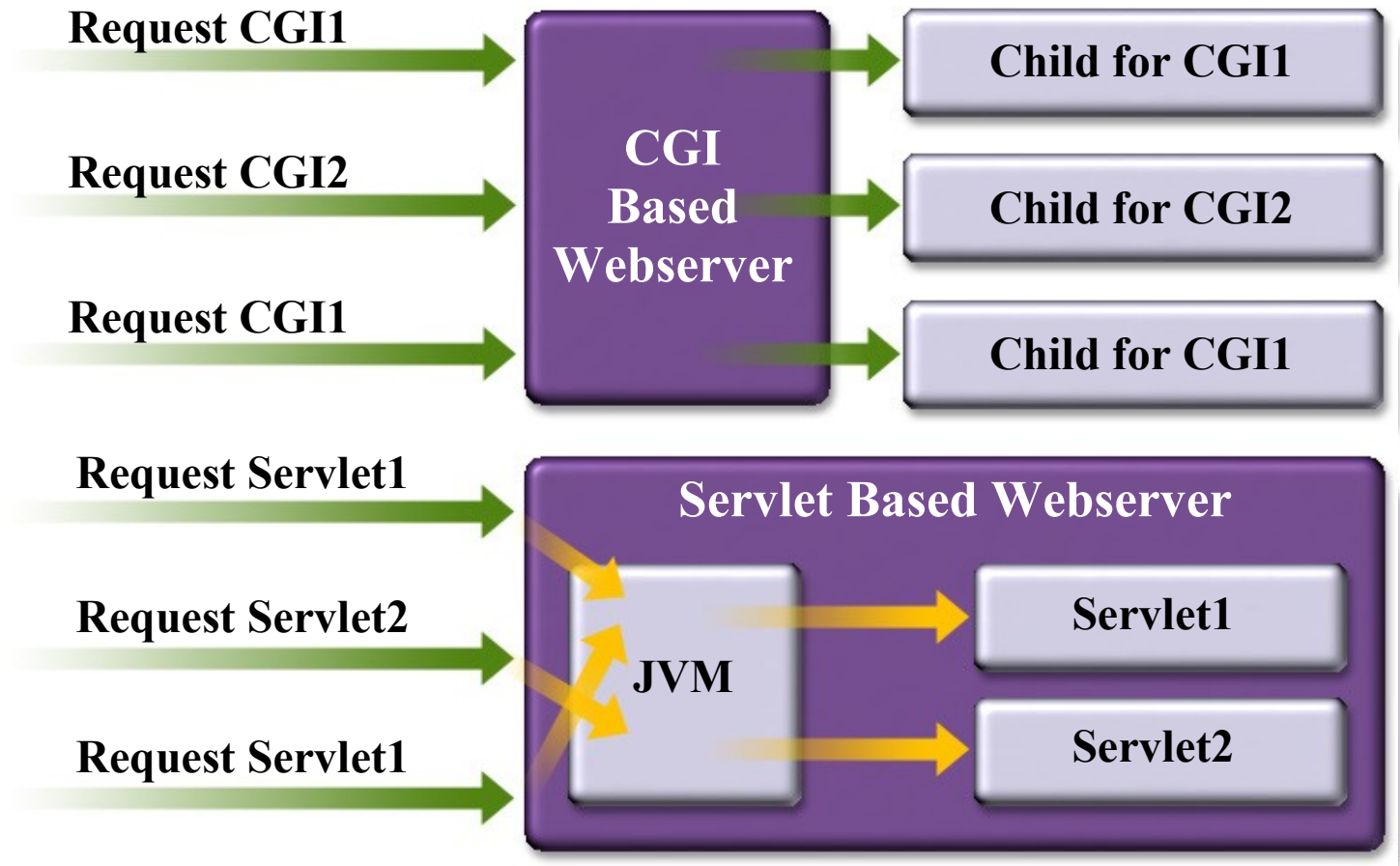
CGI

- Written in C, C++, Visual Basic and Perl
- Difficult to maintain, non-scalable, non-manageable
- Prone to security problems of programming language
- Resource intensive and inefficient
- Platform and application-specific

Servlet

- Written in Java
- Powerful, reliable, and efficient
- Improves scalability, reusability (component based)
- Leverages built-in security of Java programming language
- Platform independent and portable

Servlet vs. CGI



Advantages of Servlet إيجابيات Servlet

- No CGI limitations ليست محدودة مثل CGI
- Abundant third-party tools and Web servers supporting Servlet وفرة أدوات الطبقة الثالثة ودعم الخادومات لل Servlet
- Access to entire family of Java API الوصول إلى عائلة جافا API كلها
- Reliable, better performance and scalability ثابت مع أداء أفضل والقابلية للتوسع
- Platform and server independent إستقلالية المنصة و الخادم
- Secure محمي
- Most servers allow automatic reloading of Servlet's by administrative action أكثر الخادومات يسمحون بإعادة الشحن التلقائي لل Servlet عن طريق الإدارة

What is JSP Technology? ما هي تقنية JSP

- Enables **separation** of **business logic** from **presentation** يمكن فصل business logic عن presentation
 - Presentation is in the form of HTML or XML/XSLT
 - Business logic is implemented as **Java Beans** or **custom tags**
 - Better maintainability, reusability أكثر قابلية للصيانة و لإعادة الإستعمال
- Extensible via custom tags قابل للإمتداد من خلال الأوسمة الخاصة
- Builds on Servlet technology مبني على تقنية Servlet

What is JSP page? ما هي تقنية JSP

- A **text-based document** capable of returning dynamic content to a client browser
وثيقة نصية قابلة على إعادة محتوى حيوي* لمتصفح الحريف
- Contains both static and dynamic content
يحتوي على محتوى ثابت و حيوي
 - Static content: HTML, XML
محتوى ثابت
 - Dynamic content: programming code, and JavaBeans, custom tags
محتوى ديناميكي : المصدر البرمجي و الأوسمة الخاصة و JavaBeans

JSP Sample Code

جينة لمصدر JSP

- `<html>`
- `Hello World!`
`<br>`
`<jsp:useBean id="clock"`
`class="calendar.JspCalendar" />`
`Today is`
`<ul>`
`<li>Day of month: <%= clock.getDayOfMonth() %>`
`<li>Year: <%= clock.getYear() %>`
`</ul>`
- `</html>`

Servlets and JSP - Comparison

مقارنة بين JSP و Servlet

Servlets

- HTML code in Java
- Any form of Data
- Not easy to author a web page

JSP

- Java-like code in HTML
- Structured Text
- Very easy to author a web page
- Code is compiled into a servlet

JSP Benefits

فوائد JSP

- Content and display logic are separated
- Simplify development with JSP, JavaBeans and custom tags
تبسيط التطوير مع JSP و JavaBeans و الأوسمة الخاصة
- Supports software reuse through the use of components
دعم إعادة إستعمال البرامج من خلال إستعمال المكونات
- Recompile automatically when changes are made to the source file
إعادة تجميع تلقائي عندما يتم إدخال تغيير على الملف المصدر
- Easier to author web pages
أسهل لمؤلفي صفحات الويب
- Platform-independent
منصة مستقلة

When to use Servlet over JSP

متى نستعمل Servlet فوق JSP

- Extend the functionality of a Web server such as supporting a new file format
توسيع وظائف خادم الشبكة مثل دعم صيغة ملف جديد
- Generate objects that do not contain HTML such as graphs or pie charts
يقوم بتوليد كائنات لا تحتوي على HTML مثل الرسوم البيانية أو الخرائط الفطيرة
- **Avoid** returning HTML directly from your servlets whenever possible
تجنب إعادة HTML مباشرة من الـ Servlet قدر الإمكان

Should I Use Servlet or JSP?

هل يجب أن أستعمل JSP أو Servlet

- In practice, servlet and JSP are used together
 - عند الممارسة فإن Servlet و JSP يستعملان معا
 - via MVC (Model, View, Controller) architecture
 - من خلال بنية MVC
 - Servlet handles Controller
 - Servlet تعالج طبقة التحكم
 - JSP handles View
 - JSP يعالج طبقة العرض

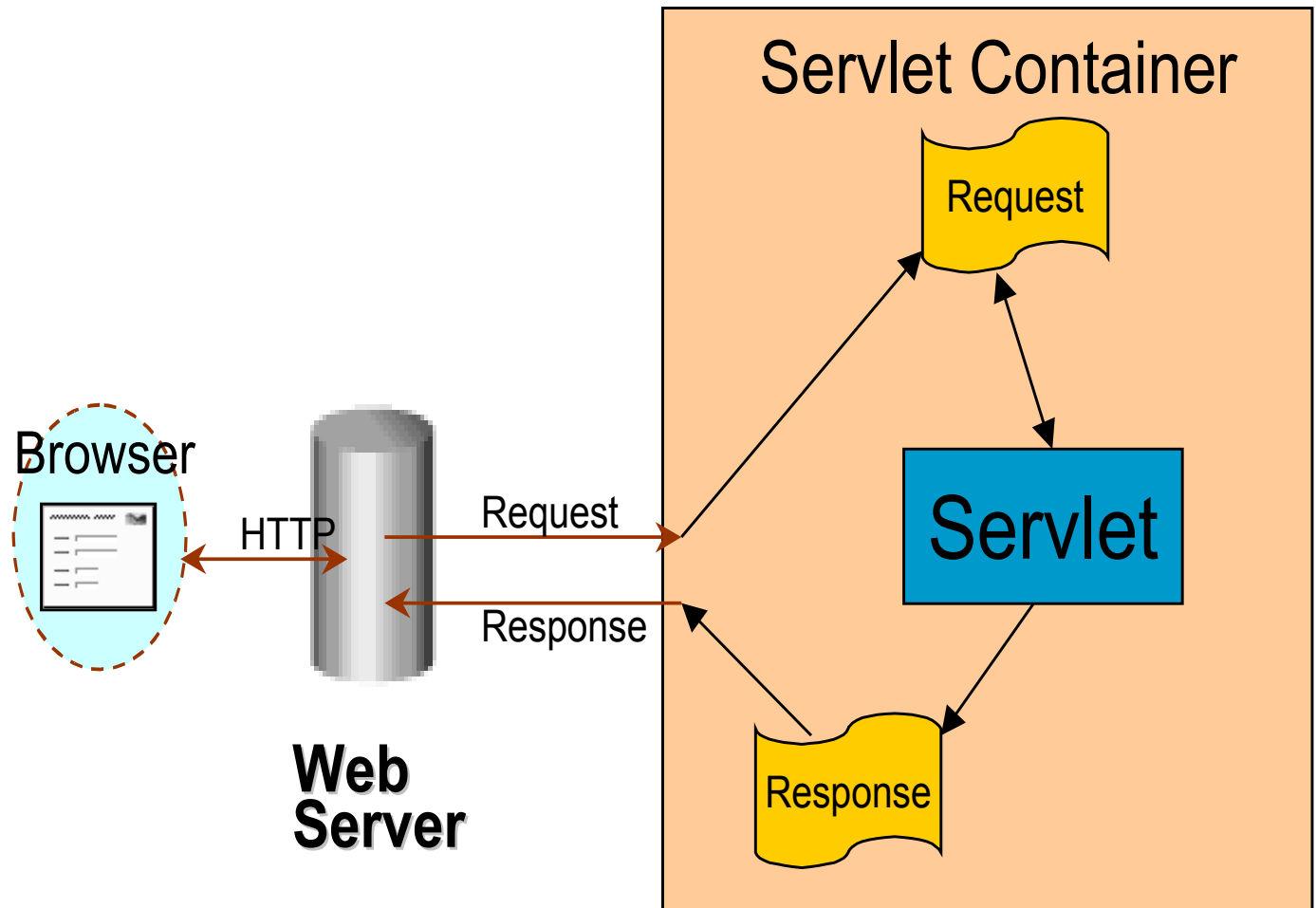


Servlet Request & Response Model

طلب Servlet نموذج إجابة

Servlet Request and Response Model

طلب Servlet و نموذج إجابة



What does Servlet Do? ماذا تفعل servlet

- Receives client request (mostly in the form of HTTP request)
يستقبل طلبات الحريف (غالبا ما تكون على هيئة طلب HTTP)
- Extract some information from the request
يقتطف بعض المعلومات من الطلب
- Do content generation or business logic process (possibly by accessing database, invoking EJBs, etc)
يقوم بتوليد المحتوى أو عمليات business logic
- Create and send response to client (mostly in the form of HTTP response) or forward the request to another servlet or JSP page
إنشاء و بعث إجابة إلى الحريف أو إحالة الطلب إلى Servlet أو صفحة JSP أخرى

Requests and Responses طلب و إجابة

- What is a request? ما هو الطلب؟
 - Information that is sent from client to a server معلومات يتم تعثها من الحريف إلى الخادم
 - Who made the request من صاحب الطلب
 - What user-entered data is sent
 - Which HTTP headers are sent
- What is a response? ما هي الإجابة؟
 - Information that is sent to client from a server معلومات يتم بعثها إلى الحريف من قبل الخادم
 - Text(html, plain) or binary(image) data
 - HTTP headers, cookies, etc

HTTP

- HTTP request contains يتضمن طلب HTTP
 - Header الترويسة
 - a method الطريقة
 - Get: Input form data is passed as part of URL مدخلات للبيانات تمر عبر URL
 - Post: Input form data is passed within message body مدخلات للبيانات تمر ضمن رسالة الجسم Body
 - Put
 - Header
 - request data طلب البيانات

HTTP GET and POST

- The most common client requests أكثر الطلبات المشتركة بين الحرفاء
 - 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 لا يستطيع إرسال إلا كمية محدودة من البيانات
 - [.../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 يستطيع إرسال أي كمية من البيانات

First Servlet

Servlet أول

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

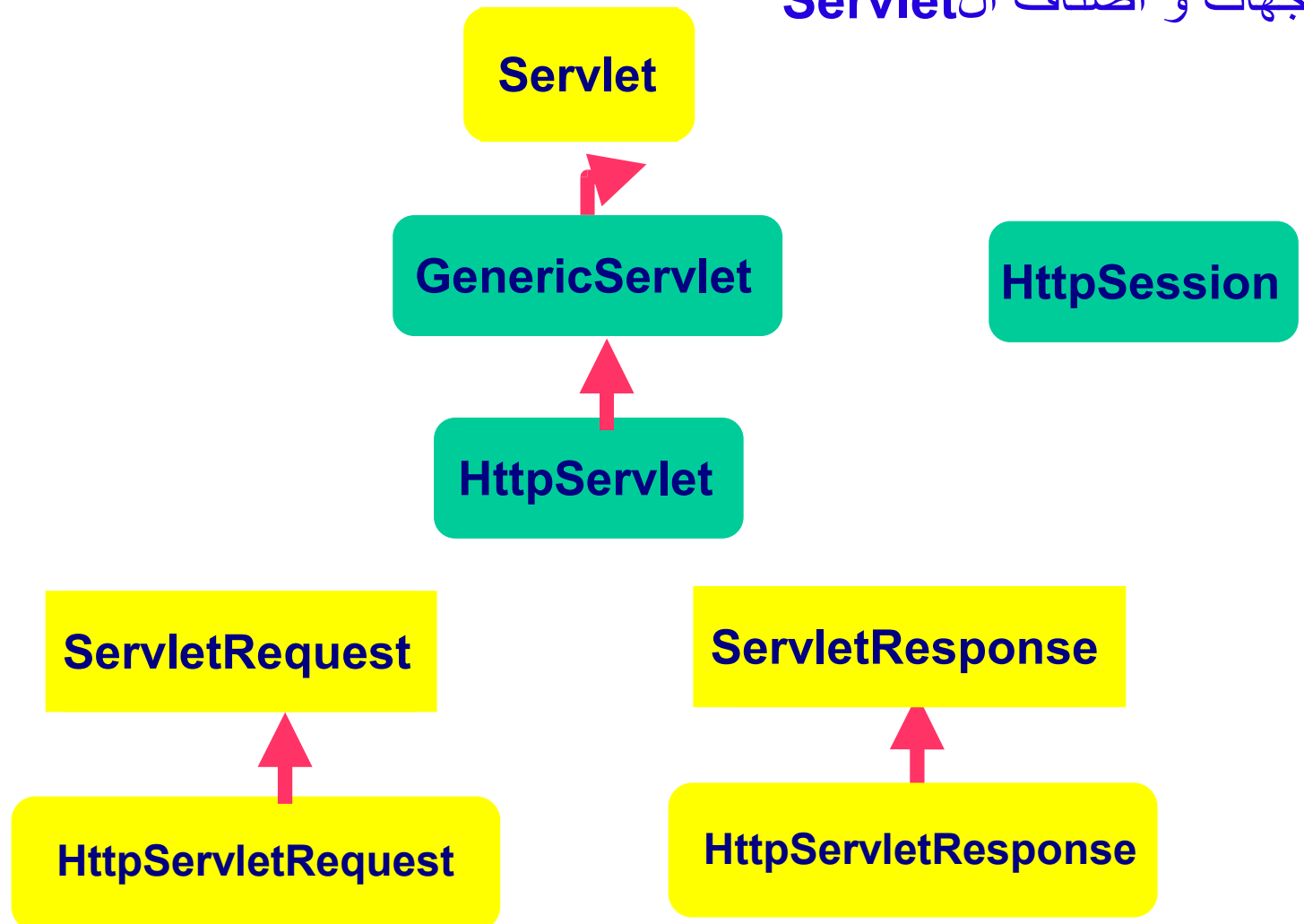
```
Public class HelloServlet extends HttpServlet {  
    public void doGet(HttpServletRequest request,  
                       HttpServletResponse response)  
        throws ServletException, IOException {  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();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

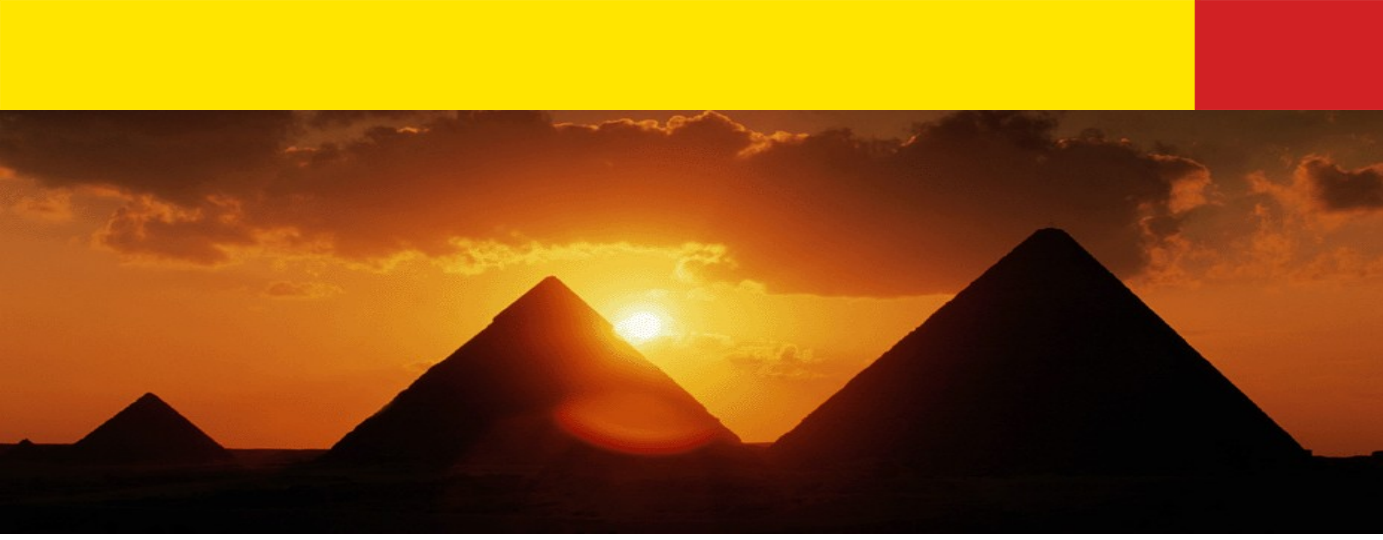


Interfaces واجهات & Classes of Servlet أصناف الServlet

Servlet Interfaces & Classes

واجهات و أصناف الServlet



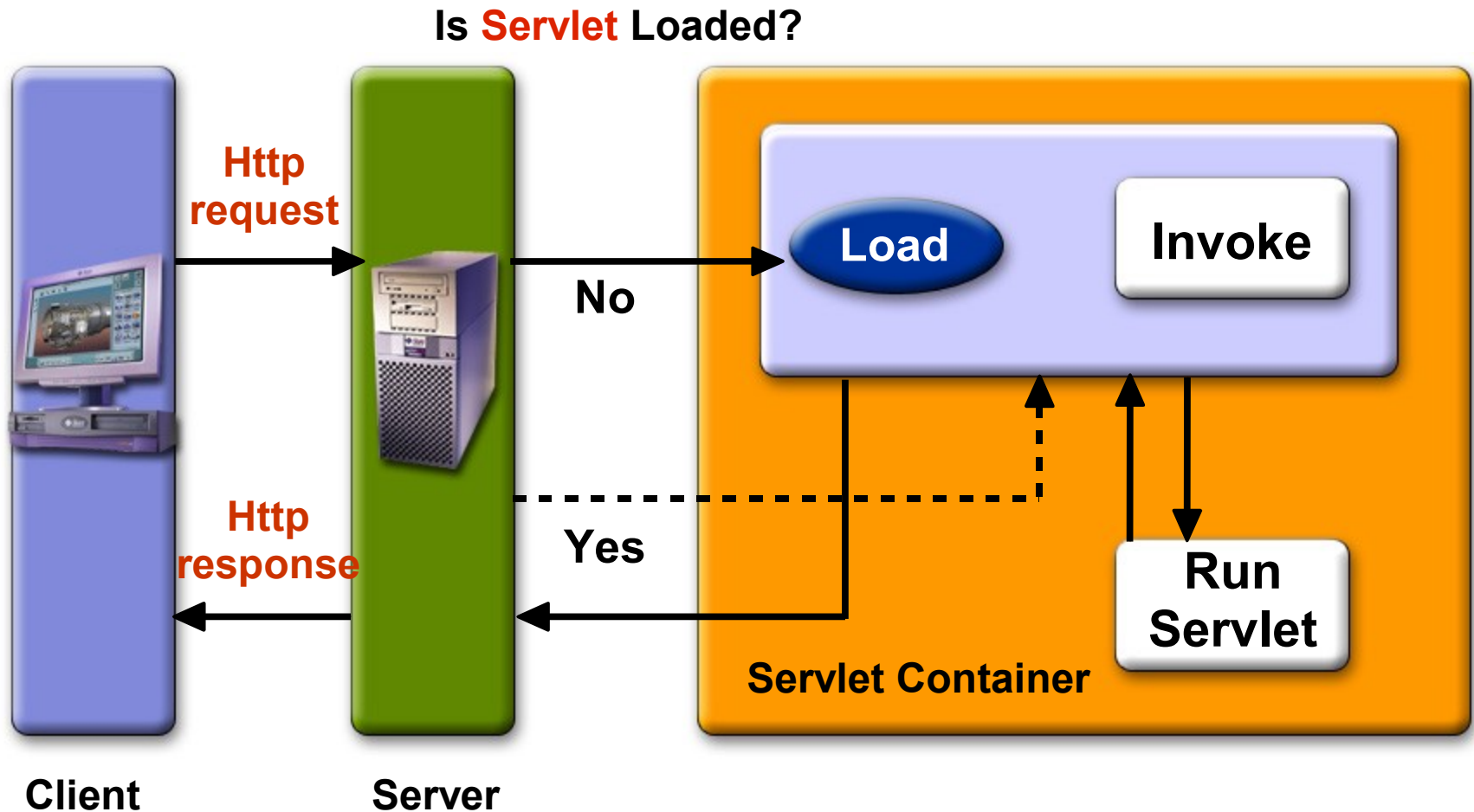


Servlet Life-Cycle

دورة حياة Servlet

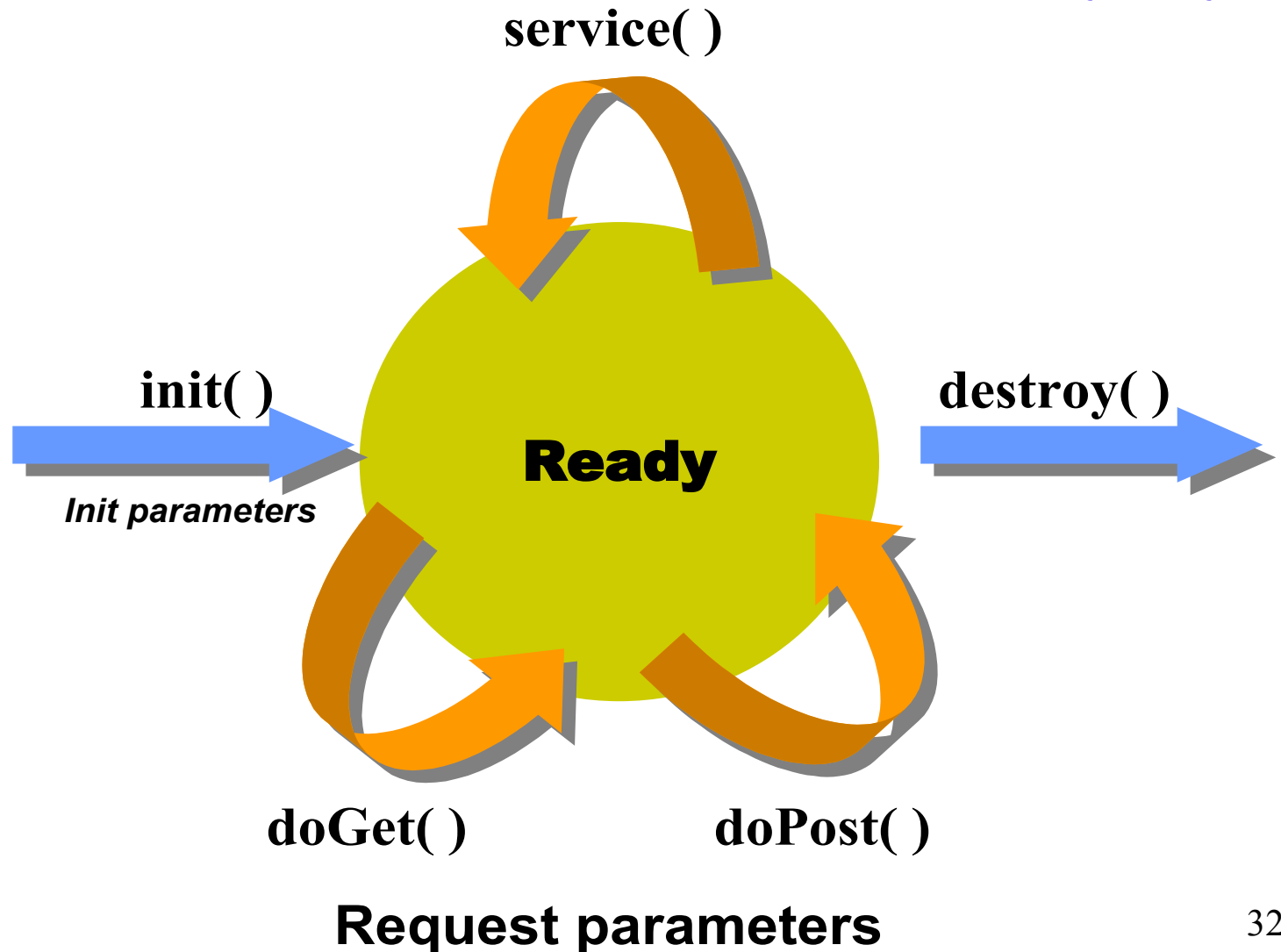
Servlet Life-Cycle

دورة حياة Servlet



Servlet Life Cycle Methods

طرق دورة حياة Servlet



Servlet Life Cycle Methods

طرق دورة حياة Servlet

- Invoked by container مستدعات من قبل الحاوية
 - Container controls life cycle of a servlet الحاوية تسيطر على دورى حياة الServlet
- Defined in عرفت في
 - `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

طرق دورة حياة Servlet

- **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

Example: 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.");  
    }  
    ...  
}
```

Example: init() reading قراءة Configuration parameters تشكيل المتوسطات

```
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 ^{إعدادات} Init Parameters ^{موسطات} 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>
```

Example: 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;
    }
    ...
}
```

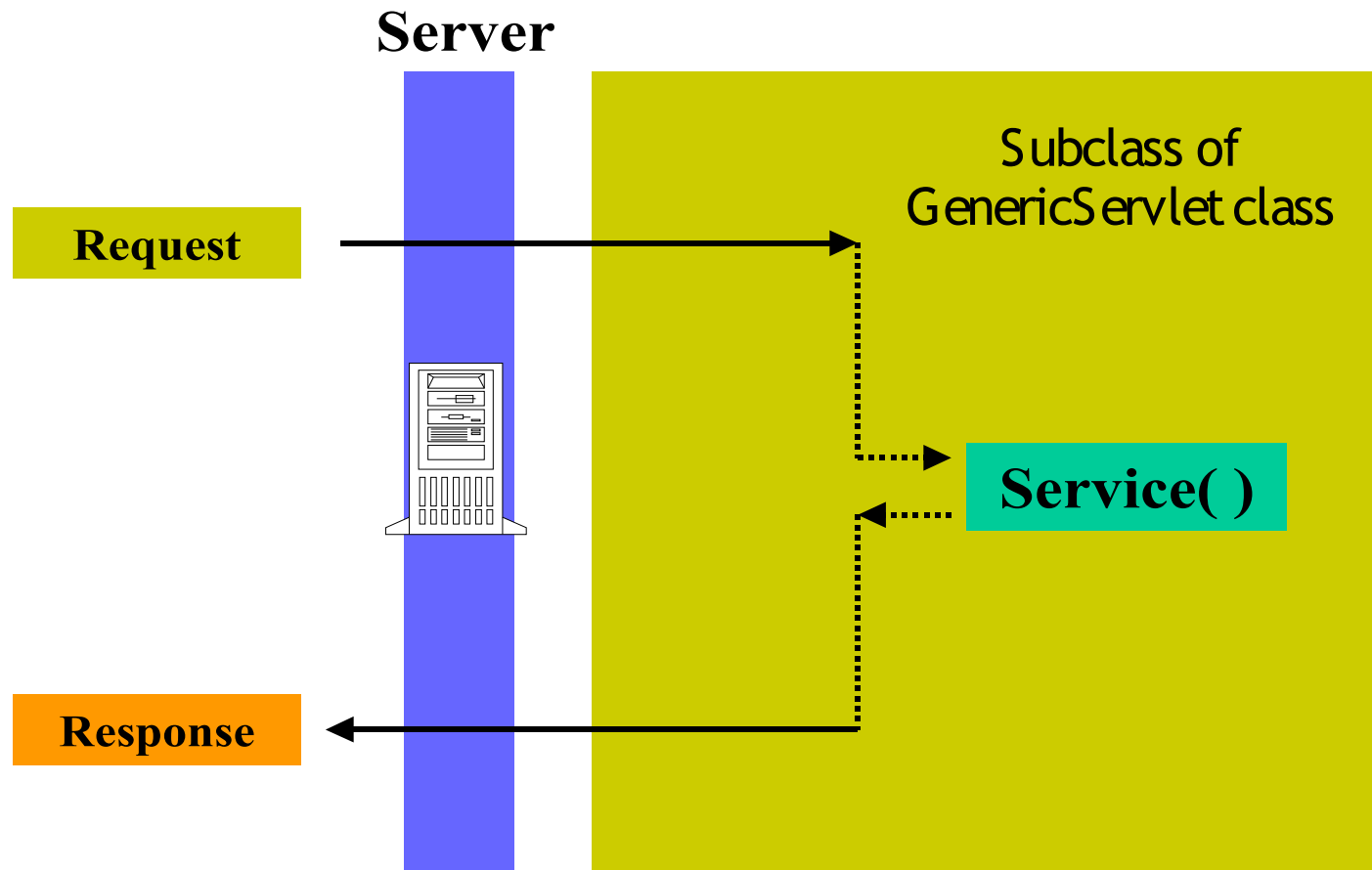
Servlet Life Cycle Methods طرق دورة حياة Servlet

- 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 `javax.servlet.http.HttpServlet`
 - Handles HTTP GET, POST, etc. requests
 - **Override these methods** in your servlet to provide desired behavior

Service() & doGet()/doPost()

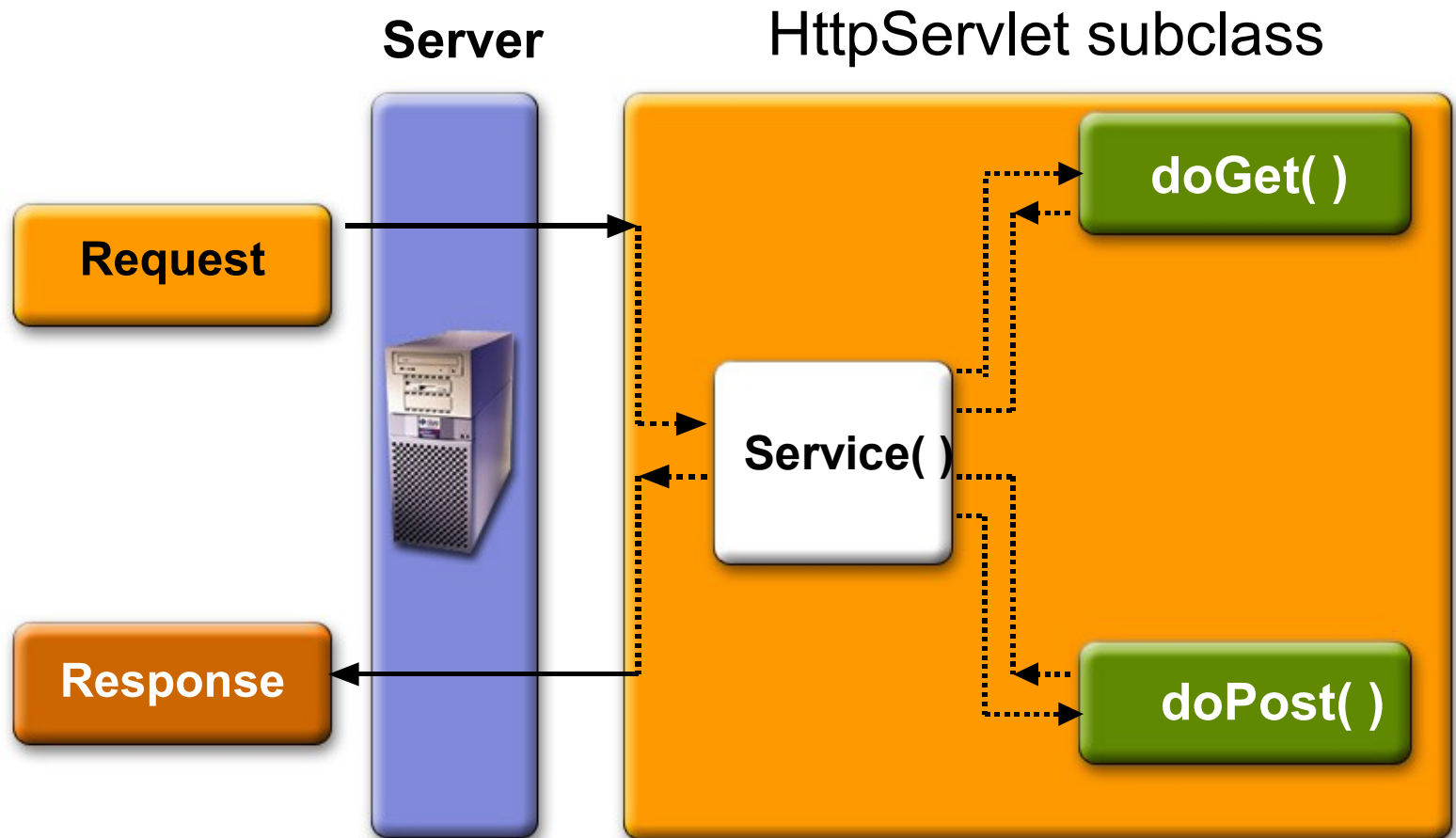
- **service()** methods take generic requests and responses: طرق للقيام بطلبات و إجابات عامة
 - `service(ServletRequest request, ServletResponse response)`
- **doGet()** or **doPost()** take HTTP requests and responses: تقوم بطلبات و إجابات HTTP
 - `doGet(HttpServletRequest request, HttpServletResponse response)`
 - `doPost(HttpServletRequest request, HttpServletResponse response)`

Service() Method طريقة



Key: Implemented by **subclass**

doGet() and doPost() Methods



Key:  Implemented by **subclass**

Things You Do in أعمال تقوم بها في doGet() & doPost()

- Extract client-sent information (HTTP parameter) from HTTP request إستخراج المعلومات المرسلة من الحريف من طلبات HTTP
- Set (Save) and get (read) attributes to/from Scope objects يتم تعيين (حفظ) و الحصول (قراءة) على الخصائص في و من كائنات النطاق
- Perform some business logic or access database إنجاز بعض business logic أو الإعتناء بالنفاز إلى قاعدة البيانات
- Optionally forward the request to other Web components (Servlet or JSP) ترسل الطلبات خياريا إلى مكونات شبكة أخرى (Servlet أو JSP)
- Populate HTTP response message and send it to client تقوم بتأهيل رسائل إجابات HTTP و ترسلها إلى الحريف

Example: Simple 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>");  
    }  
}
```

Example: Sophisticated ^{محنك} 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);
}
```

Example: Sophisticated ^{محنك} 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();
}
```

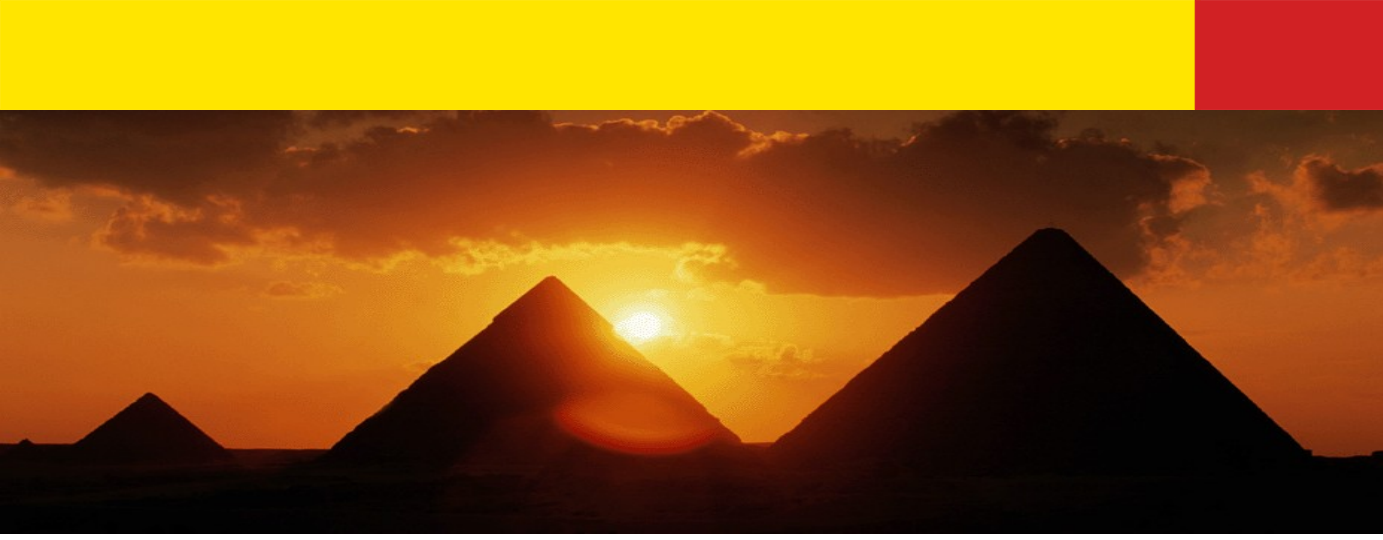
Steps of Populating HTTP Response

خطوات لتأهيل إجابة HTTP

- Fill Response headers إملاً جواب الترويسة
- Set some properties of the response تعيين بعض خصائص الإجابة
 - Buffer size
- Get an output stream object from the response
- Write body content to the output stream

Example: Simple Response إجابة بسيطة

```
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

نطاق الكائنات

Scope Objects نطاق الأهداف

- Enables **sharing information** among collaborating web components via attributes maintained in Scope objects
تفعيل الإشتراك في المعلومات بين مكونات الشبكة المتعاونة من خلال المحافظة على السمات في نطاق الكائنات
 - Attributes are name/object pairs
الخصائص عبارة عن أزواج إسم و كائن
- Attributes maintained in the Scope objects are accessed with
يتم الحفاظ على الخصائص في نطاق الكائنات و يتم النفاذ عن طريق
 - `getAttribute()` & `setAttribute()`
- 4 Scope objects are defined
أربع نطاقات تم تعريفها
 - Web context, session, request, page

Four Scope Objects: Accessibility إتاحة

- Web context (ServletContext)
 - Accessible from Web components within a Web context
متاح عن طريق كائنات الشبكة ضمن سياق الشبكة
- Session
 - Accessible from Web components handling a request that belongs to the session
متاح عن طريق كائنات الشبكة لمعالجة الطلب الذي ينتمي إلى الجلسة
- Request
 - Accessible from Web components handling the request
متاح عن طريق كائنات الشبكة لمعالجة الطلبات
- Page
 - Accessible from JSP page that creates the object
51 متاح عن طريق صفحة JSP التي تقوم بإنشاء الكائن

Four Scope Objects: Class صنف

- 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`

Web Context سياق الشبكة (ServletContext)

What is ServletContext For?

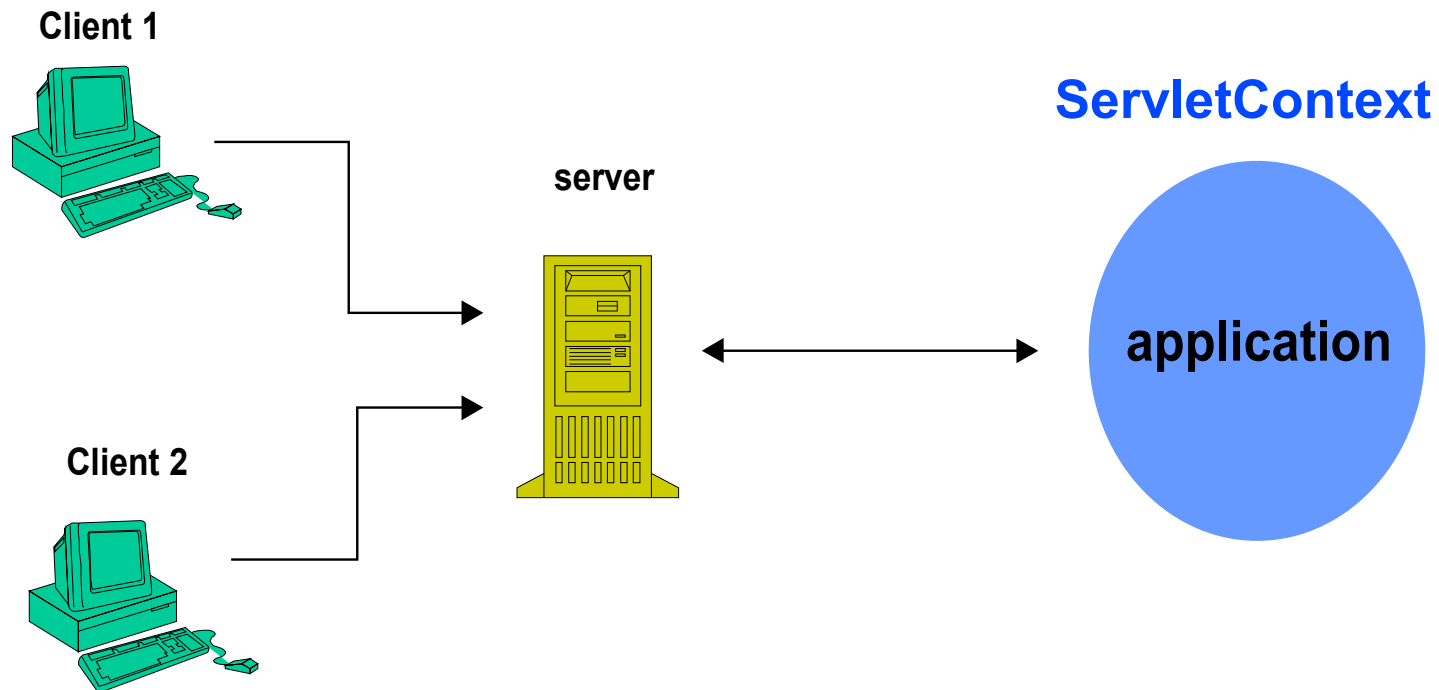
- Used by servets to تستعمل من قبل Servlet لغرض
 - Set and get context-wide (application-wide) object-valued attributes يتم التحصل و تعيين السياق العام عن طريق قيمة كائن الخصائص *
 - Get request dispatcher الحصول على الطلب المرسل
 - To forward to or include web component
 - Access Web context-wide initialization parameters set in the web.xml file web.xml النفاذ إلى السياق العام لتمهيد المتوسطات المعينة في ملف
 - Access Web resources associated with the Web context النفاذ إلى موارد الشبكة المرتبطة بسياق الشبكة
 - Log السجل
 - Access other misc. information النفاذ إلى المعلومات المتنوعة الأخرى

Scope of ServletContext

نطاق ServletContext

- Context-wide scope
 - * نطاق السياق العام
 - Shared by all servlets and JSP pages within a "web application"
مشتركة مع جميع servlets و صفحات JSP من خلال تطبيق الشبكة
 - لماذا سميت بنطاق تطبيق الشبكة "web application scope"
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
تطبيق الشبكة هي مجموعة servlets و محتويات تم تثبيتها تحت زمرة فرعية خاصة لمساحة اسم مسار الخادم و الذي يمكن تثبيته عن طريق ملف war
 - All servlets in BookStore web application share same ServletContext object
جميع servlets في تطبيق الشبكة BookStore يشتركون في نفس كائن ServletContext
 - There is one ServletContext object per "web application" per Java Virtual Machine
هناك كائن ServletContext واحد لكل تطبيق شبكة لكل JVM

ServletContext: نطاق تطبيق الشبكة



How to Access ServletContext Object?

كيف تنفذ إلى ServletContext

- Within your servlet code `call` `getServletContext()` إستدع من خلال مصدر `servlet`
- Within your servlet filter code `call` `getServletContext()` إستدع من خلال مصدر مرشح `servlet`
- The ServletContext is contained in `ServletConfig` object, which the Web server provides to a servlet when the servlet is initialized
ServletContext موجودة في كائن `ServletConfig` التي يوفرها خادم الشبكة لل `servlet` حين يتم تمهيد `servlet`
 - `init (ServletConfig servletConfig)` in Servlet interface

Example: Getting Attribute Value from ServletContext

الحصول على قيمة الخاصية من 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

الحصول على كائن RequestDispatcher و كيفية إستعماله

```
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 الدخول الدفترى

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

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



Session (HttpSession) We will talk more on HTTPSession later in “Session Tracking”

سنتحدث أكثر عن HttpSession لاحقاً في “Session Tracking”

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
نحتاج إلى آلية تمكننا من المحافظة على حالة الحريف من خلال جملة من الطلبات من نفس المستخدم (أو المنبثقة من نفس المتصفح) في فترة زمنية وجيزة
 - Example: Online shopping cart
- Yet, HTTP is stateless لا يحافظ على الحالة بعد HTTP
- HttpSession maintains client state يحافظ على حالة الحريف HttpSession
 - Used by Servlets to set and get the values of session scope attributes Servlet تستعمل من قبل أو إعطاء قيم لخصائص نطاق الجلسة

كيف يمكننا معرفة HttpSession

How to Get HttpSession?

- via getSession() method of a Request object (HttpServletRequest)
من خلال طريقة getSession() لكائن الطلب (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();
    }
}
```



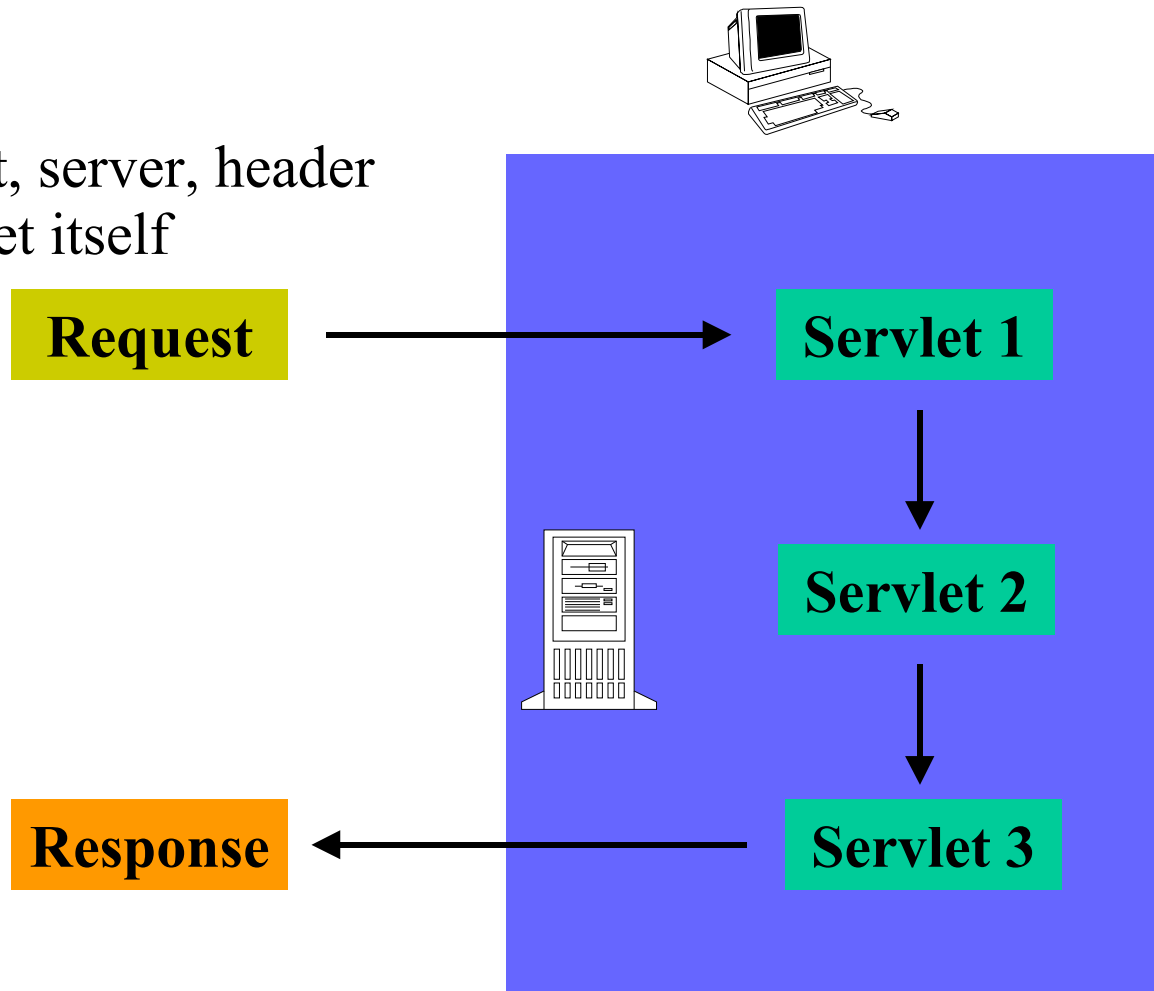
Servlet Request طلب Servlet **(HttpServletRequest)**

What is Servlet Request? ما هو طلب ال Servlet

- Contains data passed from client to servlet
يحتوي على بيانات تمر من الحريف إلى ال Servlet
- All servlet requests implement **ServletRequest** interface which defines methods for accessing
جميع طلبات servlet تقوم ببناء واجهة **ServletRequest** برمجيا حتى تعرف بطرق تمكنها من النفاذ برمجيا
 - Client sent parameters
 - Object-valued attributes
 - Locales
 - Client and server
 - Input stream
 - Protocol information
 - Content type
 - If request is made over secure channel (HTTPS)

Requests طلبات

data,
client, server, header
servlet itself



Web Server

Getting Client Sent Parameters

- A request can come with any number of parameters
يمكن أن يأتي الطلب مع عدد لا محدود من المتوسطات
- Parameters are sent from HTML forms:
ترسل المتوسطات بواسطة 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

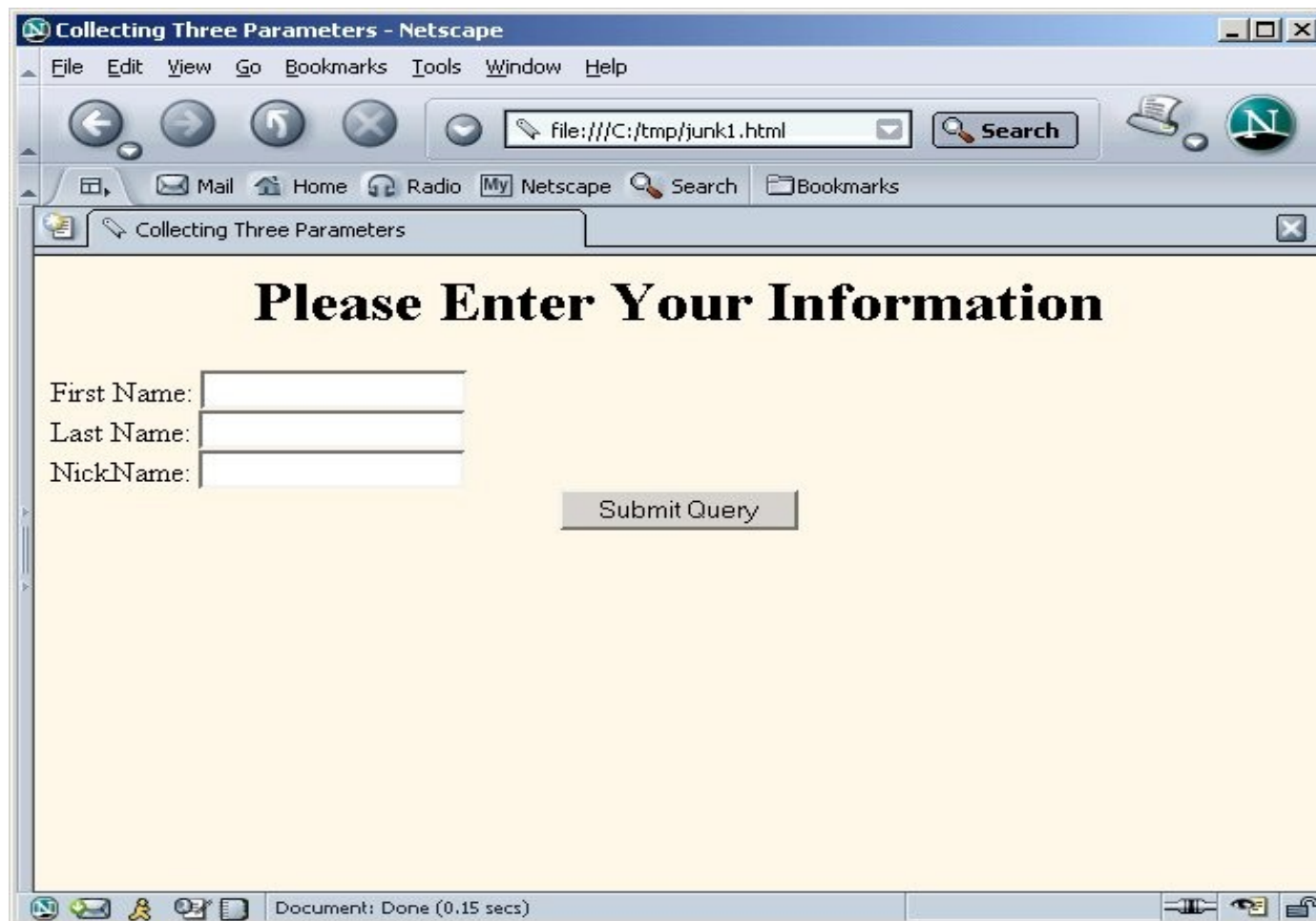
A Sample FORM using GET

```
<!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



The screenshot shows a Netscape browser window with the title 'Collecting Three Parameters - Netscape'. The address bar displays 'file:///C:/tmp/junk1.html'. The browser's menu bar includes File, Edit, View, Go, Bookmarks, Tools, Window, and Help. The toolbar contains navigation buttons (back, forward, home, stop), a search button, and a Netscape logo. The browser's status bar at the bottom shows 'Document: Done (0.15 secs)'. The main content area has a yellow background and contains the following form:

Please Enter Your Information

First Name:

Last Name:

NickName:

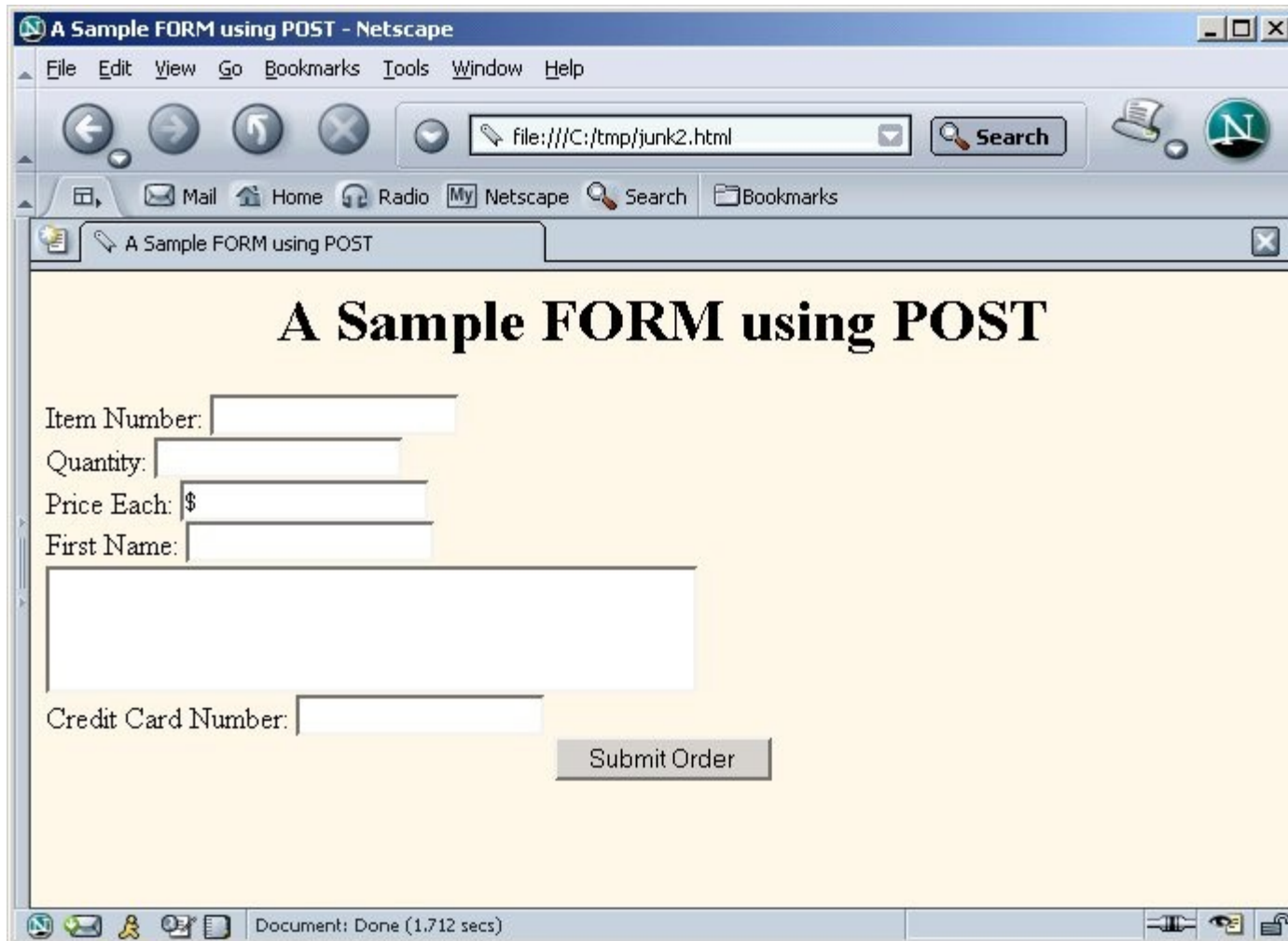
A FORM Based Servlet: Get

```
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



The screenshot shows a Netscape browser window with the title "A Sample FORM using POST - Netscape". The address bar displays "file:///C:/tmp/junk2.html". The browser's menu bar includes File, Edit, View, Go, Bookmarks, Tools, Window, and Help. The toolbar contains navigation buttons (back, forward, home, stop), a search button, and a Netscape logo. The main content area displays the form titled "A Sample FORM using POST". The form includes the following fields:

- Item Number:
- Quantity:
- Price Each: \$
- First Name:
-
- Credit Card Number:
-

The status bar at the bottom indicates "Document: Done (1.712 secs)".

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);
    }
}
```

Who Set Object/value Attributes

- Request attributes can be set in two ways

يمكن تعيين خصائص الطلب بطريقتين

- Servlet container itself might set attributes to make available custom information about a request

حاوية Servlet نفسها يمكن أن تعين خصائص حتى تتمكن من توفير معلومات خاصة بالطلب

- example: javax.servlet.request.X509Certificate attribute for HTTPS

- Servlet set application-specific attribute

Servlet تقوم بتحديد خصائص مواصفات التطبيق

- void setAttribute(java.lang.String name, java.lang.Object o)
- Embedded into a request before a **RequestDispatcher** call

مضمن في طلب قبل نداء RequestDispatcher

Getting Locale Information

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

    HttpSession session = request.getSession();
    ResourceBundle messages =

    (ResourceBundle) session.getAttribute("messages");

    if (messages == null) {
        Locale locale=request.getLocale();
        messages = ResourceBundle.getBundle(
            "messages.BookstoreMessages", locale);
        session.setAttribute("messages", messages);
    }
```

Getting Client Information

- Servlet can get client information from the request
يمكن للServlet أن تأخذ معلومات خاصة بالحريف من خلال الطلب
 - `String request.getRemoteAddr()`
 - Get client's IP address
 - `String request.getRemoteHost()`
 - Get client's host name

Getting Server Information

- Servlet can get server's information:

يمكن للServlet أن تأخذ معلومات عن الخادم

- `String request.getServerName()`
 - e.g. "www.sun.com"
- `int request.getServerPort()`
 - e.g. Port number "8080"

Getting Misc. Information الحصول على معلومات متنوعة

- Input stream
 - `ServletInputStream getInputStream()`
 - `java.io.BufferedReader getReader()`
- Protocol
 - `java.lang.String getProtocol()`
- Content type
 - `java.lang.String getContentType()`
- Is secure or not (if it is HTTPS or not)
 - `boolean isSecure()`



HttpServletRequest

What is HTTP Servlet Request?

- Contains data passed from HTTP client to HTTP servlet
يحتوي على بيانات تمر من حريف HTTP إلى HTTPServlet
- Created by servlet container and passed to servlet as a parameter of `doGet()` or `doPost()` methods
أنشأت عن طريق حاوية servlet و تمر إلى servlet كموسط في طريقة `doGet()` أو طريقة `doPost()`
- `HttpServletRequest` is an extension of `ServletRequest` and provides additional methods for accessing
`HttpServletRequest` هو إمتداد لـ `ServletRequest` و يوفر طرق إضافية للنفاز
 - 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 يحتوي على الأجزاء التالية
 - `http://[host]:[port]/[request path]?[query string]`

HTTP Request URL: [request path]

- `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**
السلسلة الإستعلامية متكونة من مجموعة من
الموسطات و القيم التي أدخلها المستخدم
- Two ways query strings are generated
طريقتان لتوليد السلسلة الإستعلامية
 - A query string can explicitly appear in a web page
السلسلة الإستعلامية يمكن أن تظهر ضمناً في صفحة الشبكة
 - `<a href="/bookstore1/catalog?Add=101">Add To Cart</a>`
 - `String bookId = request.getParameter("Add");`
 - A query string is appended to a URL when a form with a **GET HTTP method** is submitted
يتم إلحاق سلسلة إستعلامية إلى
المسار حين يتم الإرسال على طريقة GET HTTP
 - `http://localhost/hello1/greeting?username=Monica+Clinton`
 - `String userName=request.getParameter("username")`

Context , Path , Query , Parameter Methods

السياق , المسار , إستعلام , طرق المتوسطات

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

HTTP Request Headers

ترويسات طلب HTTP

- HTTP requests include headers which provide extra information about the request طلبات HTTP تتضمن ترويسات توفر بدورها معلومات دقيقة حول الطلب

- Example of HTTP 1.1 Request:

GET /search? keywords= servlets+ jsp HTTP/ 1.1

Accept: image/ gif, image/ jpg, */*

Accept-Encoding: gzip

Connection: Keep- Alive

Cookie: userID= id456578

Host: www.sun.com

Referer: http://www.sun.com/codecamp.html

User-Agent: Mozilla/ 4.7 [en] (Win98; U)

HTTP Request Headers

ترويسات طلب HTTP

- Accept
 - Indicates MIME types browser can handle.
- Accept-Encoding
 - Indicates encoding (e. g., gzip or compress) browser can handle
- Authorization
 - User identification for password- protected pages
 - Instead of HTTP authorization, use HTML forms to send username/password and store info in session object

HTTP Request Headers

ترويسات طلب HTTP

- Connection

- In HTTP 1.1, persistent connection is default
- Servlets should set Content-Length with setContentLength (use ByteArrayOutputStream to determine length of output) to support persistent connections.

يجب على Servlets تعيين طول المحتوى عن طريق دالة setContentLength لدعم الإتصال المستمر

- Cookie

- Gives cookies sent to client by server sometime earlier. Use get_cookies, not getHeader

- Host

- Indicates host given in original URL.
- This is required in HTTP 1.1.

HTTP Request Headers

ترويسات طلب HTTP

- **If-Modified-Since**
 - Indicates client wants page only if it has been changed after specified date.
 - Don't handle this situation directly; implement `getLastModified` instead.
- **Referer**
 - URL of referring Web page.
 - Useful for tracking traffic; logged by many servers.
- **User-Agent**
 - String identifying the browser making the request.
 - Use with extreme caution!

HTTP Header Methods

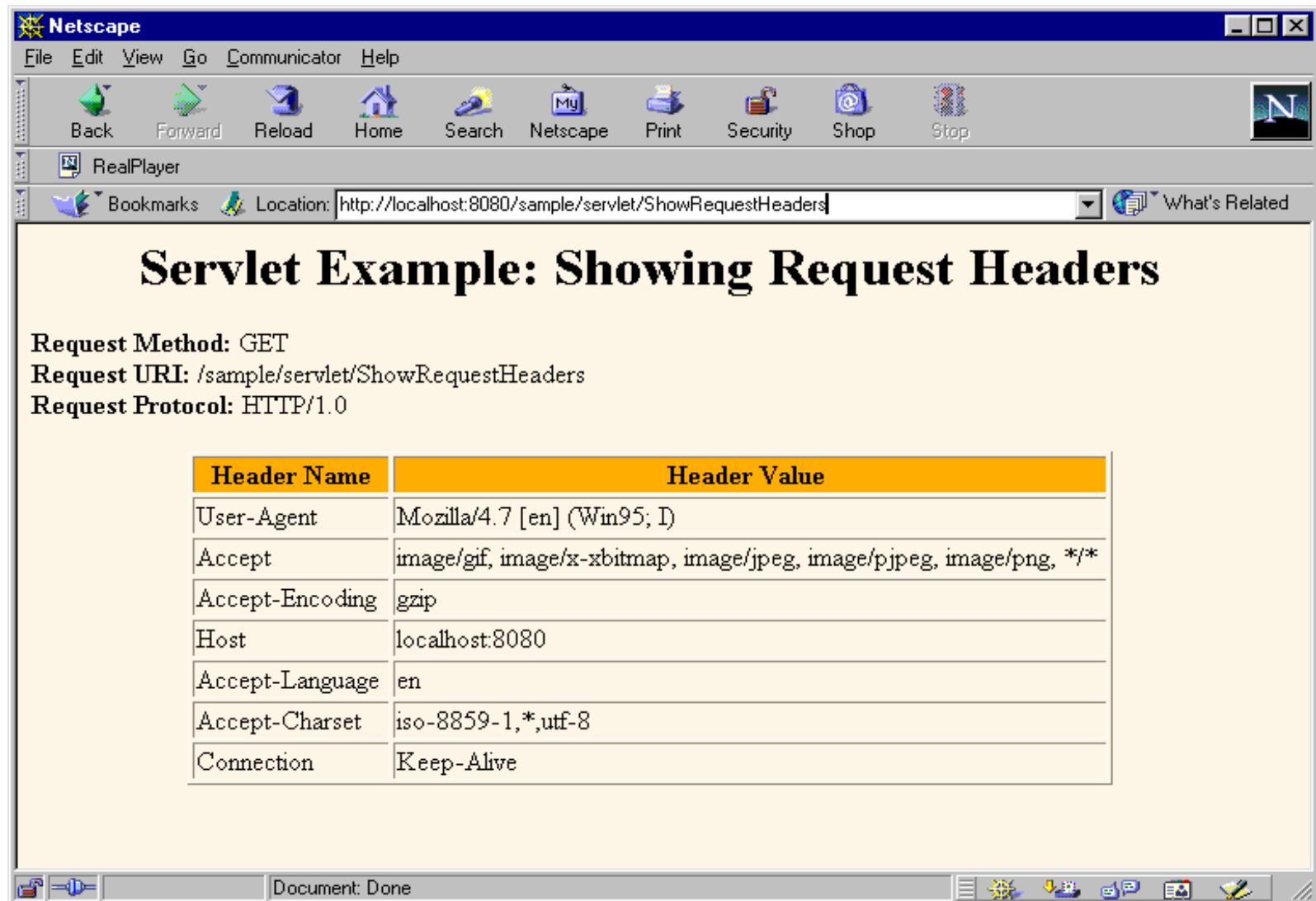
- `String getHeader(java.lang.String name)`
 - value of the specified request header as `String`
- `java.util.Enumeration getHeaders(java.lang.String name)`
 - values of the specified request header
- `java.util.Enumeration getHeaderNames()`
 - names of request headers
- `int getIntHeader(java.lang.String name)`
 - value of the specified request header as an `int`

Showing Request Headers

//Shows all the request headers sent on this particular request.

```
public class ShowRequestHeaders extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title = "Servlet Example: Showing Request Headers";
        out.println("<HTML>" + ...
                    "<B>Request Method: </B>" +
                    request.getMethod() + "<BR>\n" +
                    "<B>Request URI: </B>" +
                    request.getRequestURI() + "<BR>\n" +
                    "<B>Request Protocol: </B>" +
                    request.getProtocol() + "<BR><BR>\n" +
                    ...
                    "<TH>Header Name<TH>Header Value");
        Enumeration headerNames = request.getHeaderNames();
        while(headerNames.hasMoreElements()) {
            String headerName = (String)headerNames.nextElement();
            out.println("<TR><TD>" + headerName);
            out.println("    <TD>" + request.getHeader(headerName));
        }
        ...
    }
}
```

Request Headers Sample عينة من ترويسة طلب



The screenshot shows the Netscape Communicator window. The title bar says "Netscape". The menu bar includes File, Edit, View, Go, Communicator, and Help. The toolbar contains icons for Back, Forward, Reload, Home, Search, Netscape, Print, Security, Shop, and Stop. The address bar shows the location: `http://localhost:8080/sample/servlet/ShowRequestHeaders`. The main content area displays the title "Servlet Example: Showing Request Headers" and the following information:

Request Method: GET
Request URI: /sample/servlet/ShowRequestHeaders
Request Protocol: HTTP/1.0

Header Name	Header Value
User-Agent	Mozilla/4.7 [en] (Win95; I)
Accept	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding	gzip
Host	localhost:8080
Accept-Language	en
Accept-Charset	iso-8859-1,*,utf-8
Connection	Keep-Alive

The status bar at the bottom shows "Document: Done".

Authentication & User Security Information Methods

- `String getRemoteUser()`
 - name for the client user if the servlet has been password protected, null otherwise
- `String getAuthType()`
 - name of the authentication scheme used to protect the servlet
- `boolean isUserInRole(java.lang.String role)`
 - Is user is included in the specified logical "role"?
- `String getRemoteUser()`
 - login of the user making this request, if the user has been authenticated, null otherwise

Cookie Method (in HttpServletRequest)

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



Servlet Response **(HttpServletResponse)**

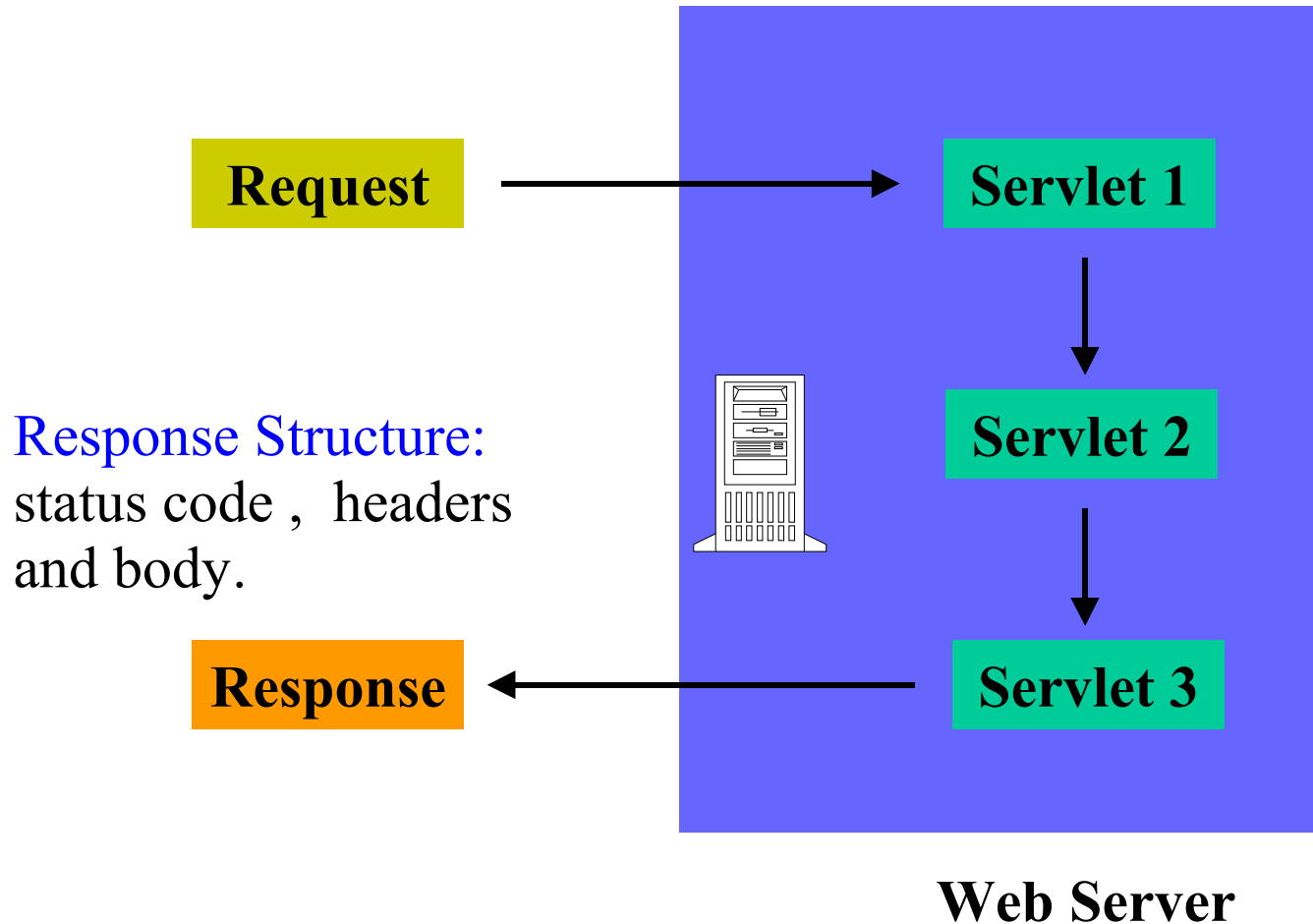
إجابة Servlet

What is Servlet Response? ما هي إجابة Servlet؟

- Contains data passed from servlet to client
تحتوي على بيانات تم إرسالها من servlet إلى الحريف
- All servlet responses implement ServletResponse interface
جميع إجابات servlet تقوم بإكمال بناء الواجهة ServletResponse برمجيا
 - Retrieve an output stream إسترجاع الإخراج المتدفق
 - Indicate content type تحديد نوع المحتوى
 - Indicate whether to buffer output
 - Set localization information تعيين مكان المعلومات
- **HttpServletResponse** extends ServletResponse
HttpServletResponse تقوم بتمديد ServletResponse
 - HTTP response status code
 - Cookies

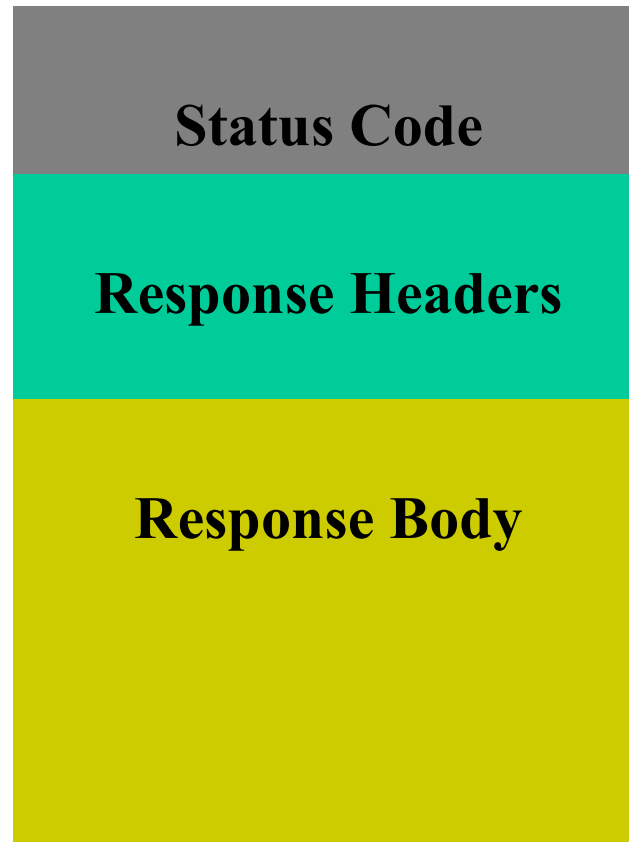
Responses

الإجابات



Response Structure

بنية الردود



Status Code in Http Response

حالة الشيفرة في إجابة Http

HTTP Response Status Codes

حالة الشيفرات في إجابة Http

- Why do we need HTTP response status code?

لماذا نحتاج إلى إجابة حالة المصدر؟

- Forward client to another page لإرسال الحريف إلى صفحة أخرى
- Indicates resource is missing للإشارة إلى فقدان المورد
- Instruct browser to use cached copy أمر المتصفح بإستعمال النسخة المخفية

Methods for Setting HTTP Response Status Codes

طرق إعداد حالة شيفرات إجابة HTTP

- `public void setStatus(int statusCode)`
 - Status codes are defined in `HttpServletResponse`
 - Status codes are numeric fall into five general categories:
 - 100-199 Informational
 - 200-299 Successful
 - 300-399 Redirection
 - 400-499 Incomplete
 - 500-599 Server Error
 - Default status code is 200 (OK)

Example of HTTP Response Status

مثال لحالة إجابة HTTP

```
HTTP/ 1.1 200 OK
Content-Type: text/ html
<! DOCTYPE ...>
<HTML
...
</ HTML>
```

Common Status Codes

- 200 (SC_OK)
 - Success and document follows
 - Default for servlets
- 204 (SC_No_CONTENT)
 - Success but no response body
 - Browser should keep displaying previous document
- 301 (SC_MOVED_PERMANENTLY)
 - The document moved permanently (indicated in Location header)
 - Browsers go to new location automatically

Common Status Codes

- 302 (SC_MOVED_TEMPORARILY)
 - Note the message is "Found"
 - Requested document temporarily moved elsewhere (indicated in Location header)
 - Browsers go to new location automatically
 - Servlets should use `sendRedirect`, not `setStatus`, when setting this header
- 401 (SC_UNAUTHORIZED)
 - Browser tried to access password-protected page without proper Authorization header
- 404 (SC_NOT_FOUND)
 - No such page

Methods for Sending Error طرق إرسال الأخطاء

- Error status codes (400-599) can be used in `sendError` methods.
- `public void sendError(int sc)`
 - The server may give the error special treatment
- `public void sendError(int code, String message)`
 - Wraps `message` inside small HTML document

setStatus() & sendError()

```
try {
    returnAFile(fileName, out)
}
catch (FileNotFoundException e)
{   response.setStatus(response.SC_NOT_FOUND) ;
    out.println("Response body") ;
}
```

has same effect as

```
try {
    returnAFile(fileName, out)
}
catch (FileNotFoundException e)
{   response.sendError(response.SC_NOT_FOUND) ;
}
```

Header in Http Response

الترويسة في إجابة Http

Why HTTP Response Headers?

- Give forwarding location *إعطاء الموقع المرسل
- Specify cookies تحديد cookies
- Supply the page modification date تزويد الصفحة بتاريخ التعديلات
- Instruct the browser to reload the page after a designated interval أمر المتصفح بإعادة تحميل الصفحة بعد فترة زمنية معينة
- Give the file size so that persistent HTTP connections can be used إعطاء حجم الملف حتى يمكن إستعمال إتصالات HTTP مستمرة
- Designate the type of document being generated تعيين نوع الوثيقة التي تم توليدها *
- Etc.

Methods for Setting Arbitrary Response Headers

- `public void setHeader( String headerName, String headerValue)`
 - Sets an arbitrary header.
- `public void setDateHeader( String name, long millisecs)`
 - Converts milliseconds since 1970 to a date string in GMT format
- `public void setIntHeader( String name, int headerValue)`
 - Prevents need to convert int to String before calling `setHeader`
- `addHeader, addDateHeader, addIntHeader`
 - Adds new occurrence of header instead of replacing.

Methods for setting Common Response Headers

- **setContentType**
 - Sets the Content- Type header. Servlets almost always use this.
- **setContentLength**
 - Sets the Content- Length header. Used for persistent HTTP connections.
- **addCookie**
 - Adds a value to the Set- Cookie header.
- **sendRedirect**
 - Sets the Location header and changes status code.

Common HTTP 1.1 Response Headers

- Location
 - Specifies a document's new location.
 - Use `sendRedirect` instead of setting this directly.
- Refresh
 - Specifies a delay before the browser automatically reloads a page.
- Set-Cookie
 - The cookies that browser should remember. Don't set this header directly.
 - use `addCookie` instead.

Common HTTP 1.1 Response Headers (cont.)

- Cache-Control (1.1) and Pragma (1.0)
 - A no-cache value prevents browsers from caching page. Send both headers or check HTTP version.
- Content- Encoding
 - The way document is encoded. Browser reverses this encoding before handling document.
- Content- Length
 - The number of bytes in the response. Used for persistent HTTP connections.

Common HTTP 1.1 Response Headers (cont.)

- Content- Type
 - The MIME type of the document being returned.
 - Use `setContentType` to set this header.
- Last- Modified
 - The time document was last changed
 - Don't set this header explicitly.
 - provide a `getLastModified` method instead.

Refresh Sample Code

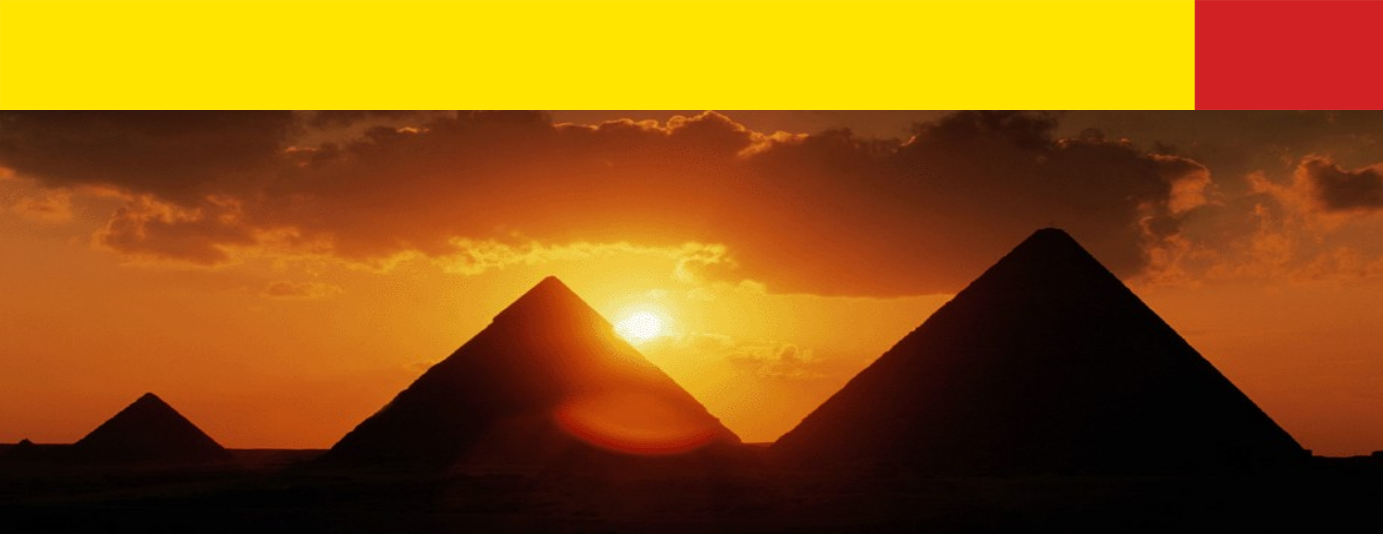
```
public class DateRefresh extends HttpServlet {  
    public void doGet(HttpServletRequest req,  
                      HttpServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/plain");  
        PrintWriter out = res.getWriter();  
        res.setHeader("Refresh", "5");  
        out.println(new Date().toString());  
    }  
}
```

Body in Http Response

الجسم في إجابة Http

Writing a Response Body كتابة جسم الإجابة

- A servlet almost always returns a response body
تقوم servlet دائما تقريبا بإرجاع جسم للإجابة
- Response body could either be a **PrintWriter** or a **ServletOutputStream**
جسم الإجابة إما أن يكون **PrintWriter** أو **ServletOutputStream**
- **PrintWriter**
 - Using `response.getWriter()`
 - For character-based output
- **ServletOutputStream**
 - Using `response.getOutputStream()`
 - For binary (image) data



Handling Errors

معالجة الأخطاء

Handling Errors معالجة الأخطاء

- Web container generates default error page
تقوم حاوية الشبكة بتوليد صفحة خطأ مبدئية
- You can specify custom default page to be displayed instead
يمكن تحديد صفحة مبدئية خاصة للظهور بدلاً من صفحة الخطأ
- Steps to handle errors خطوات لمعالجة الأخطاء
 - Create appropriate error html pages for error conditions
إنشاء صفحات html مناسبة لشروط الخطأ
 - Modify the web.xml accordingly تعديل web.xml وفقاً لذلك

Example: Setting Error Pages in web.xml

```
<error-page>
  <exception-type>
    exception.BookNotFoundException
  </exception-type>
  <location>/errorpage1.html</location>
</error-page>
<error-page>
  <exception-type>
    exception.BooksNotFoundException
  </exception-type>
  <location>/errorpage2.html</location>
</error-page>
<error-page>
  <exception-type>exception.OrderException</exception-type>
  <location>/errorpage3.html</location>
</error-page>
```



Passion!