

CHUYÊN ĐỀ JAVA



SERVLET CƠ BẢN

Nguyễn Hoàng Anh – nhanh@fit.hcmuns.edu.vn

Nội dung trình bày

- Servlet trong kiến trúc J2EE
- Mô hình Servlet Request & Servlet Response.
- Chu kỳ sống của Servlet
- Các đối tượng phạm vi trong Servlet
- Servlet Request
- Servlet Response
- Xử lý lỗi

The background features a central white area with a bokeh effect of out-of-focus light circles. This central area is framed by solid blue horizontal bars at the top and bottom. A thin vertical blue line is positioned to the left of the main text.

SEVLET TRONG KIẾN TRÚC J2EE

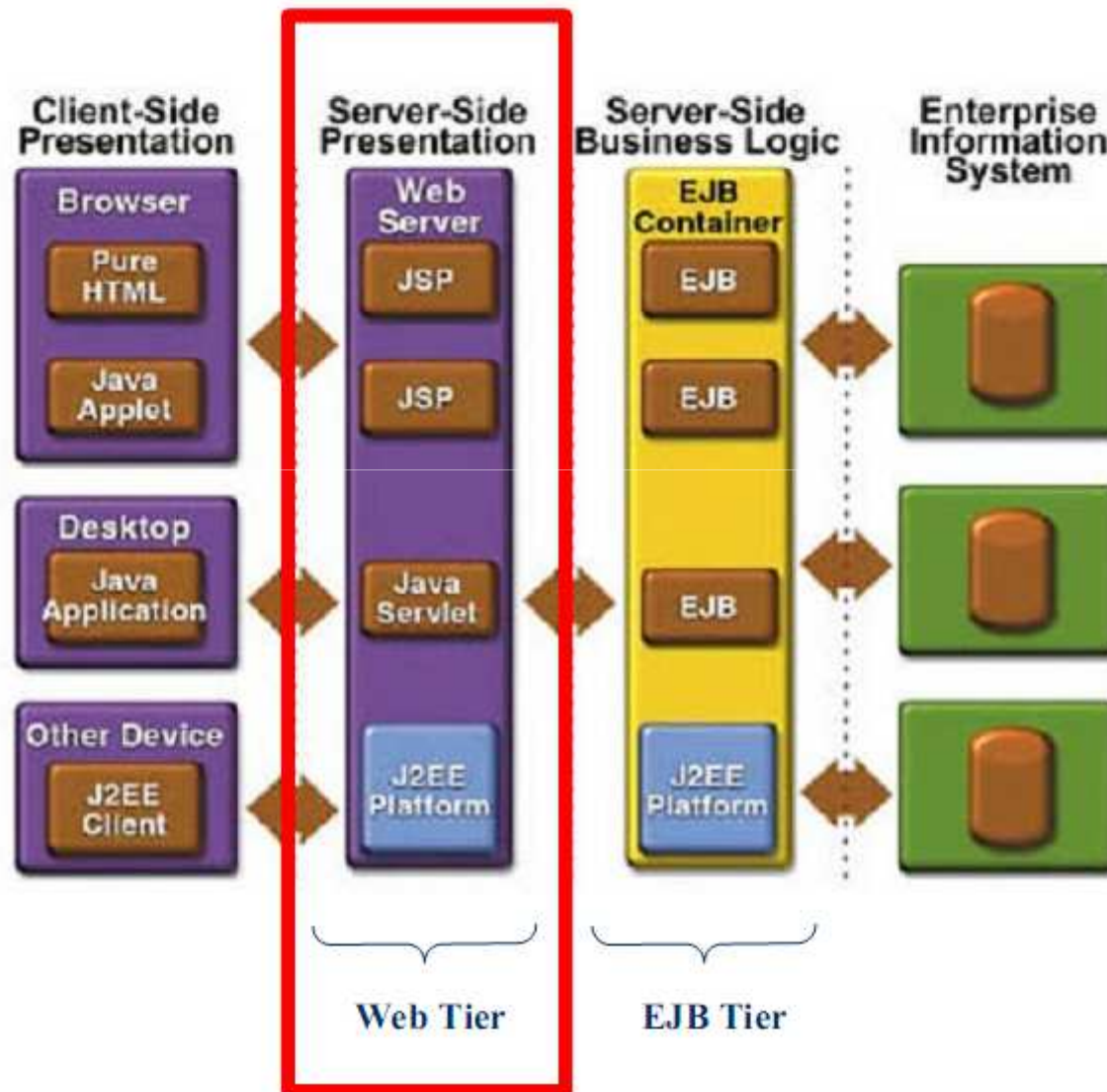
Kiến trúc J2EE

- JSP: Các Template Data, ngôn ngữ Script, Custom Element và các đối tượng Java-Server Side → nội dung động → Client.
- Template Data: XML, HTML
- Client : Web Browser

Servlets: Chương trình java cho phép phát sinh nội dung động và tương tác với các Web Client thông qua cơ chế Request - Response



Servlet và JSP bên trong Server Side Presentation



Servlet là gì?

- Là các đối tượng java
 - Dựa trên Servlet Framework và các API
 - Mở rộng chức năng của HTTP Server
- Được ánh xạ tới các URL và được quản lý bởi Container.
- Chạy trên tất cả các Web Server và App Server
- Độc lập với Server và Platform

Ví dụ: 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>");  
    }  
    ...  
}
```

Servlet vs CGI

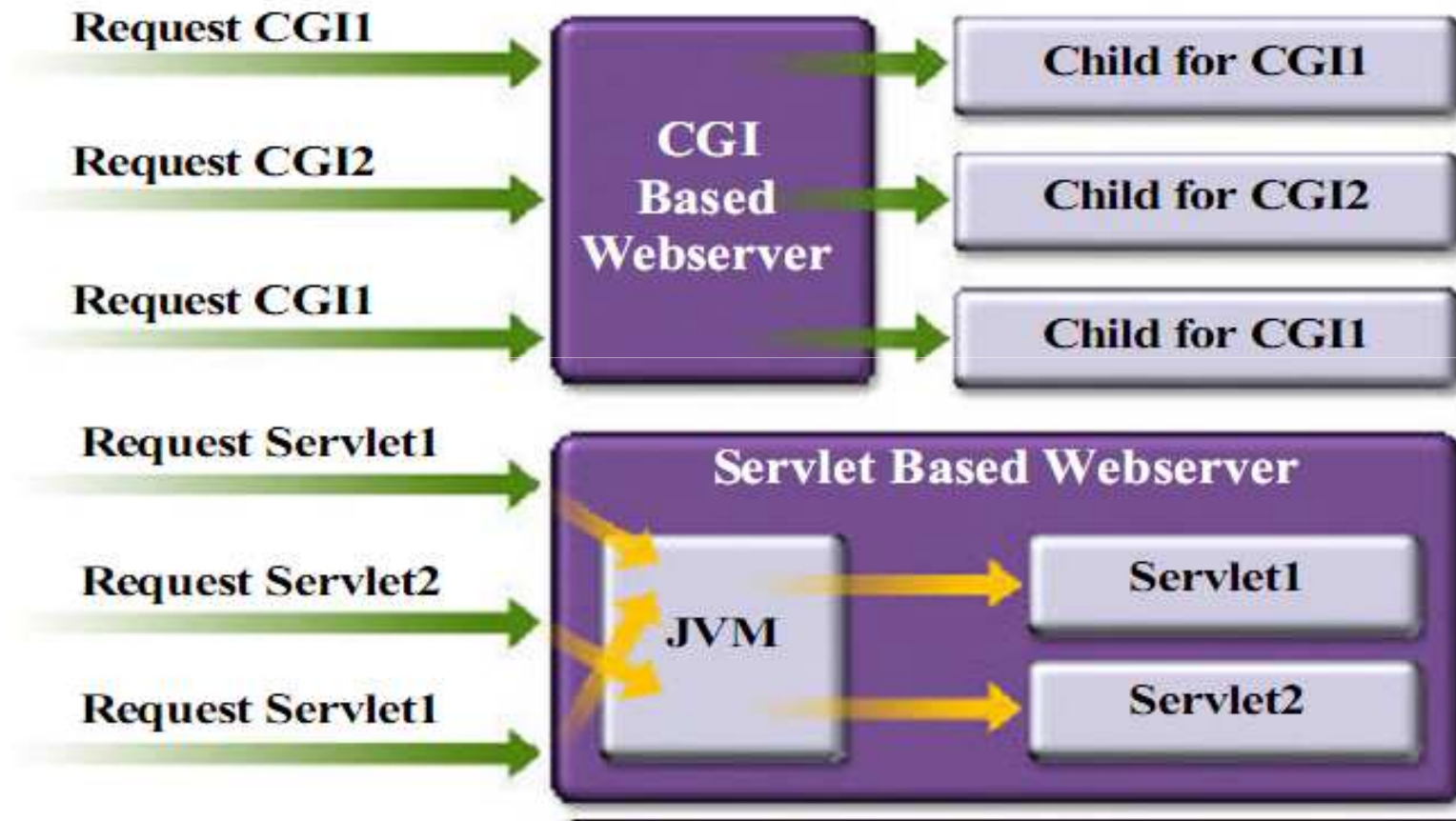
CGI

- C/C++, VB, Pert
- Khó bảo trì
- Gặp phải các vấn đề bảo mật của ngôn ngữ lập trình
- Cần nhiều tài nguyên và không hiệu quả
- Phụ thuộc vào ứng dụng và Platform

Servlet

- Java
- Mạnh, tin cậy, hiệu quả
- Cải tiến khả năng dùng lại và dễ phân chia (dựa trên Component)
- Bảo mật cao do dựa trên Java
- Độc lập Platform và dễ mang chuyển

Servlet vs CGI



Các thuận lợi khi sử dụng Servlet

- Không có các giới hạn của CGI
- Có nhiều Tool và Web Server hỗ trợ Servlet
- Sử dụng được toàn bộ Java API
- Đáng tin cậy, khả năng thực thi tốt, dễ phân chia
- Bảo mật
- Hầu hết các server cho phép nạp lại các Servlet khi chúng được thay đổi.

Công nghệ JSP là gì?

- Cho phép tách Business Logic ra từ Presentation
 - Presentation có thể là HTML hoặc XML/XSLT
 - Business Logic được cài đặt như là các JavaBean hoặc các Custom Tag
 - Bảo trì và tái sử dụng tốt hơn.
- Mở rộng thông qua các Custom Tag
- Xây dựng dựa trên công nghệ Servlet

Trang JSP là gì?

- Là tài liệu dựa trên văn bản có khả năng trả về nội dung động cho client browser.
- Chứa cả nội dung tĩnh và nội dung động
 - Nội dung tĩnh: HTML, XML
 - Nội dung động: Java Code, Java Bean, Custom Tag.

Ví dụ: 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>
```

So sánh Servlet và JSP

SERVLET

- Trang HTML được tạo từ Java Code
- Bất kỳ kiểu dữ liệu
- Khó thiết kế một trang web

JSP

- Java Code được chèn vào trang HTML
- Kiểu dữ liệu chính là văn bản
- Dễ thiết kế một trang web
- JSP được biên dịch sang Servlet

Các thuận lợi của JSP

- Nội dung và hiển thị được phân cách
- Đơn gian hóa phát triển ứng dụng với JSP với việc sử dụng JavaBean, Custom Tag
- Tái sử dụng thông qua các Component
- Khi các trang JSP thay đổi sẽ được tự động biên dịch lại.
- Dễ dàng thiết kế các trang web
- Độc lập platform

Khi nào sử dụng Servlet thay JSP

- Mở rộng chức năng của Web Server như là hỗ trợ thêm định dạng tập tin mới
- Phát sinh các đối tượng không chứa nội dung HTML như là đồ thị hoặc biểu đồ
- Tránh việc trả kết quả HTML về client trực tiếp từ Servlet thấp nhất có thể có

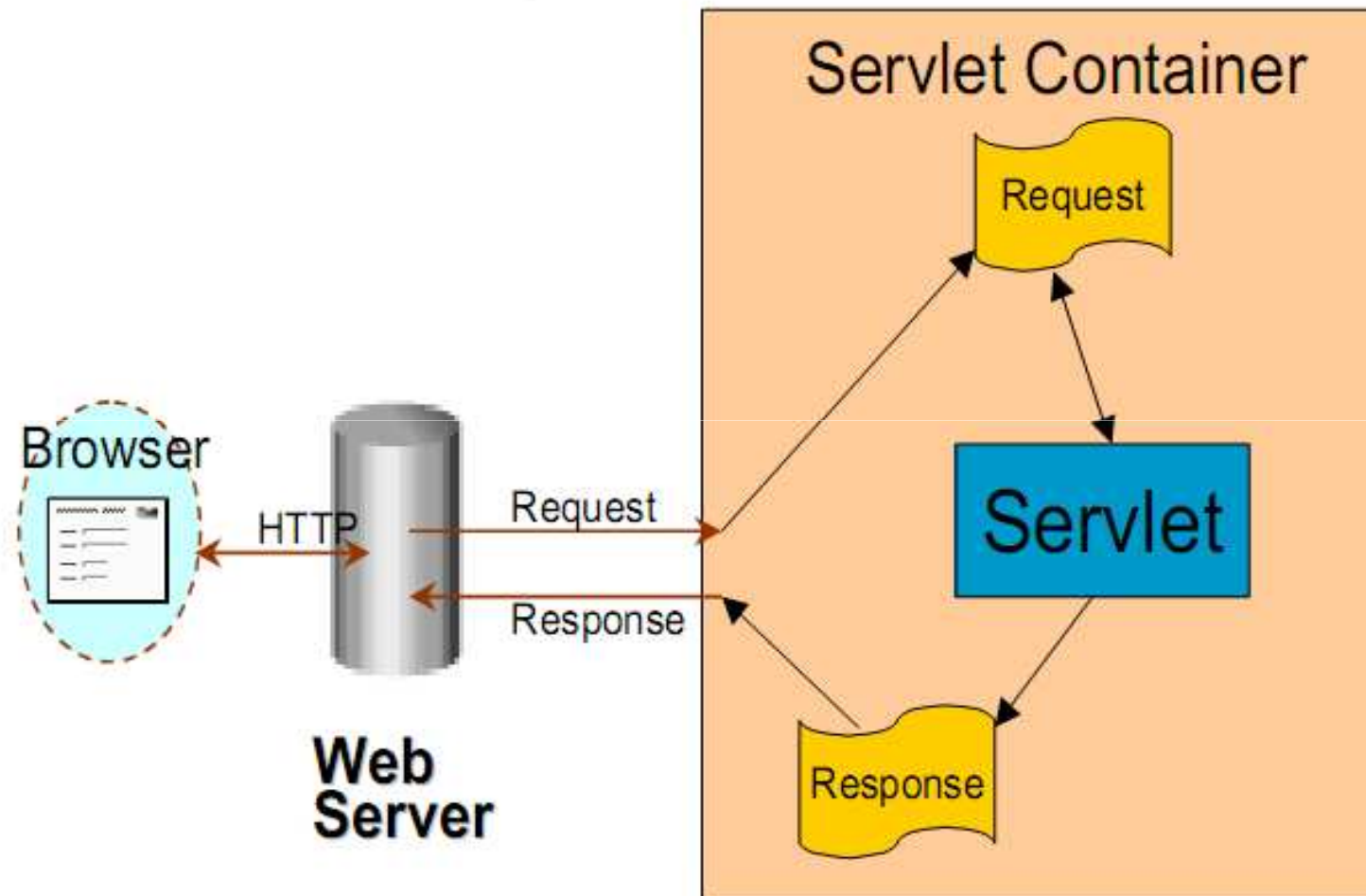
Sử dụng Servlet hay JSP?

- Trong thực tế JSP và Servlet thường được sử dụng kết hợp với nhau:
 - Kiến trúc MVC (Model – View – Controller)
 - Servlet xử lý Controller
 - JSP xử lý View

The background features a solid blue horizontal bar at the top and another at the bottom. The central area is white with a faint, artistic bokeh effect consisting of various sized grey and white circles and vertical lines, giving it a digital or network-like appearance.

MÔ HÌNH SERVLET REQUEST & SERVLET RESPONSE

Mô hình Servlet Request & Servlet Response



Servlet dùng để làm gì?

- Nhận các yêu cầu từ client (thông thường là HTTP Request)
- Lấy một số thông tin từ HTTP Request
- Thực thi các yêu cầu từ Client như là truy xuất dữ liệu sử dụng JDBC thuần túy hoặc gọi các thành phần EJB,...
- Tạo và gửi phản hồi cho client (thường HTTP Response) hoặc chuyển các yêu cầu của client đến trang JSP hoặc Servlet khác xử lý

Request & Response

- Request là gì?
 - Thông tin gửi từ Client đến Server
 - Người tạo ra Request
 - Dữ liệu gửi
 - Các HTTP Header
- Response là gì?
 - Thông tin gửi từ Server đến Client
 - Dữ liệu văn bản (HTML, XML,...), dữ liệu nhị phân (ảnh)
 - Các HTTP Header, Cookies,...

HTTP

- HTTP Request chứa:
 - Header
 - Một phương thức:
 - GET
 - POST
 - Dữ liệu yêu cầu

HTTP GET & HTTP POST

- Các yêu cầu thông dụng của client thường là : HTTP GET & HTTP POST
- GET Request:
 - Người dùng điền các thông tin được gắn vào cuối URL dưới dạng chuỗi truy vấn
 - Do chiều dài URL bị giới hạn nên dữ liệu gửi đi theo dạng GET cũng bị giới hạn
 - .../servlet/ViewCourse?FirstName=Sang&LastName=Shin

HTTP GET & HTTP POST

- POST Request:
 - Dữ liệu được gửi đi không gắn vào cuối URL
 - Không giới hạn dung lượng gửi đi

Ví dụ Servlet

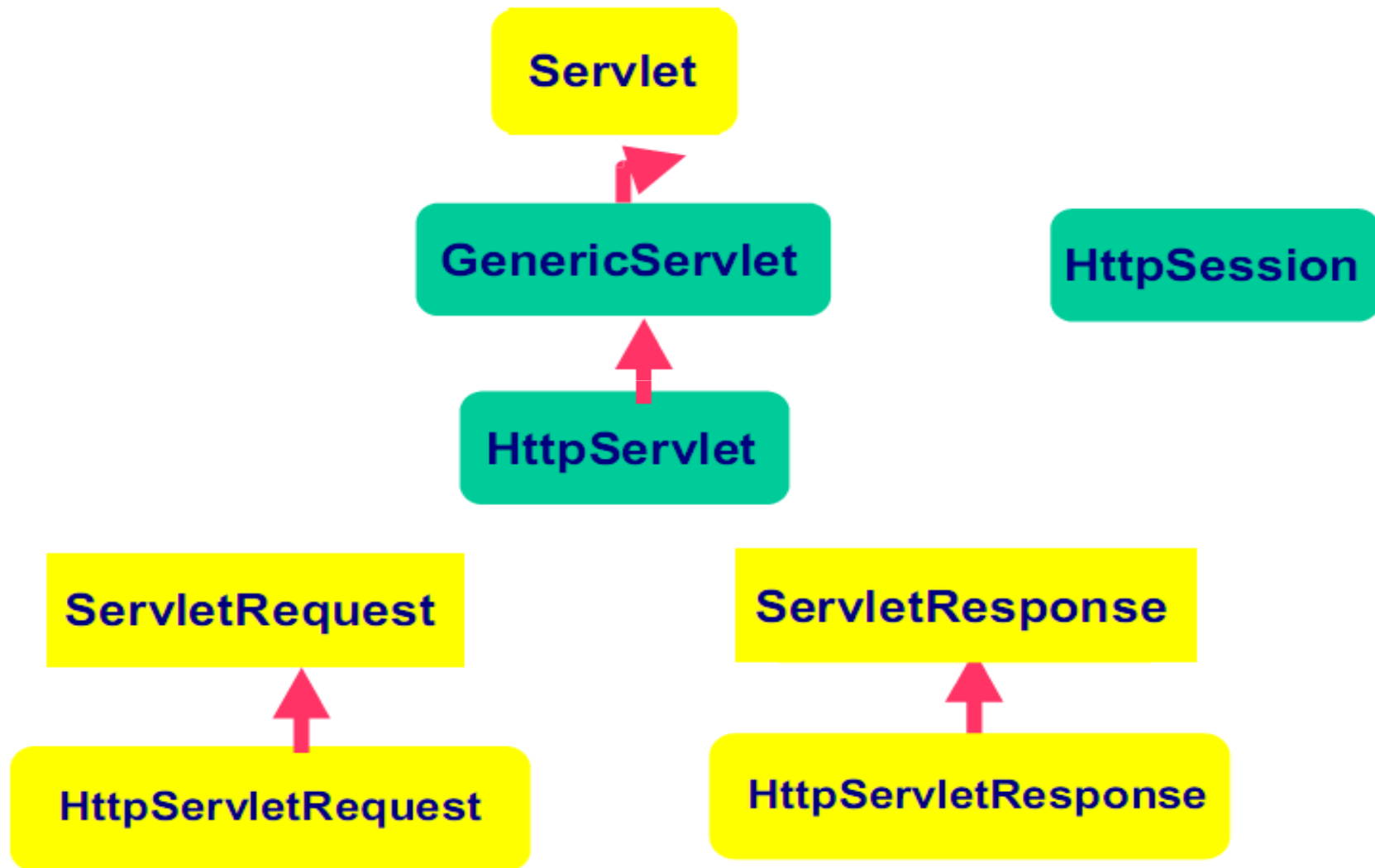
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

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>");
    }
}
```

The background features a solid blue horizontal bar at the top and another at the bottom. The central area is white with a faint, artistic bokeh effect consisting of various sized grey and white circles and vertical lines, resembling light reflections or digital data points.

CÁC INTERFACE VÀ CLASS TRONG SEVLET

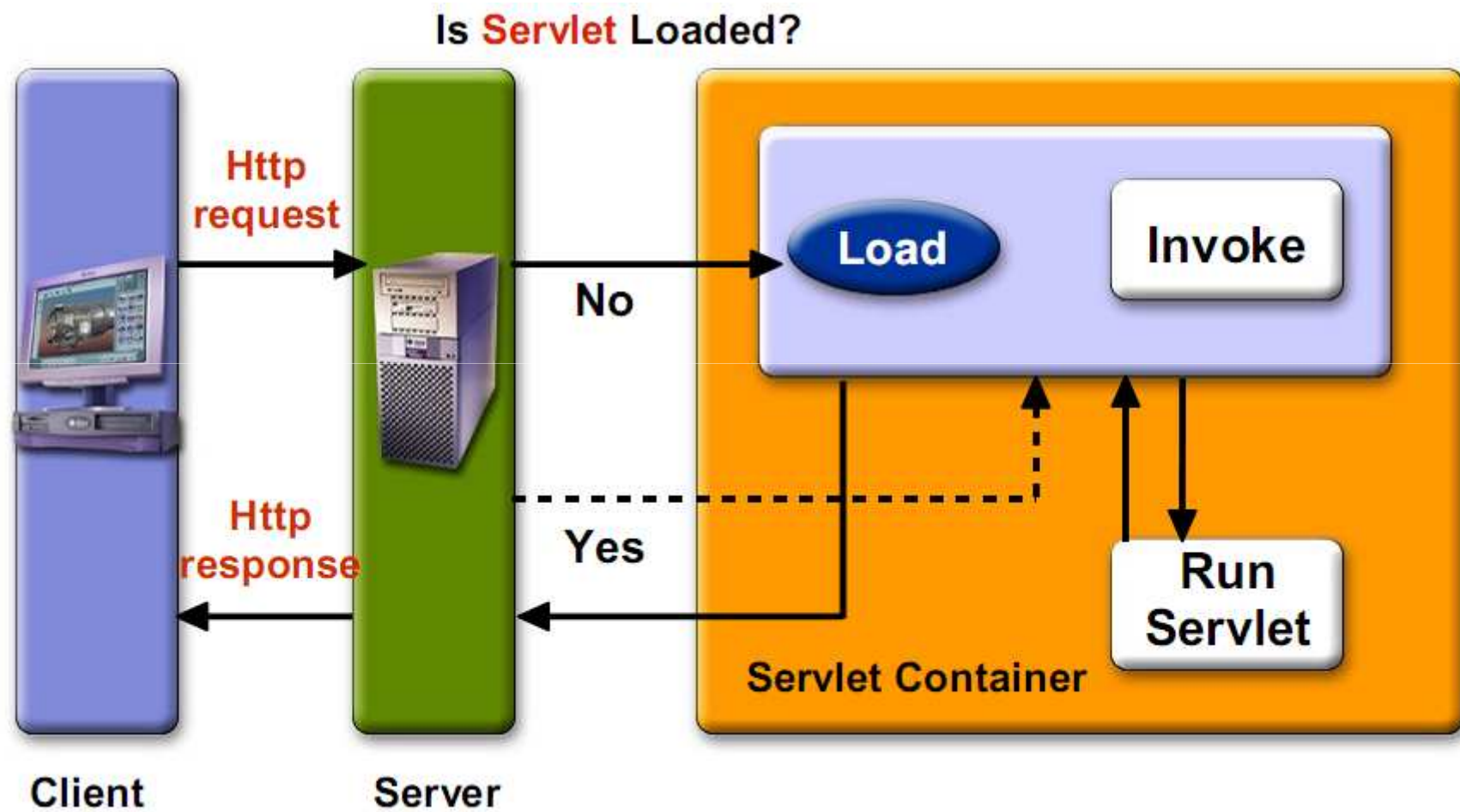
Các Interface và Class trong Servlet



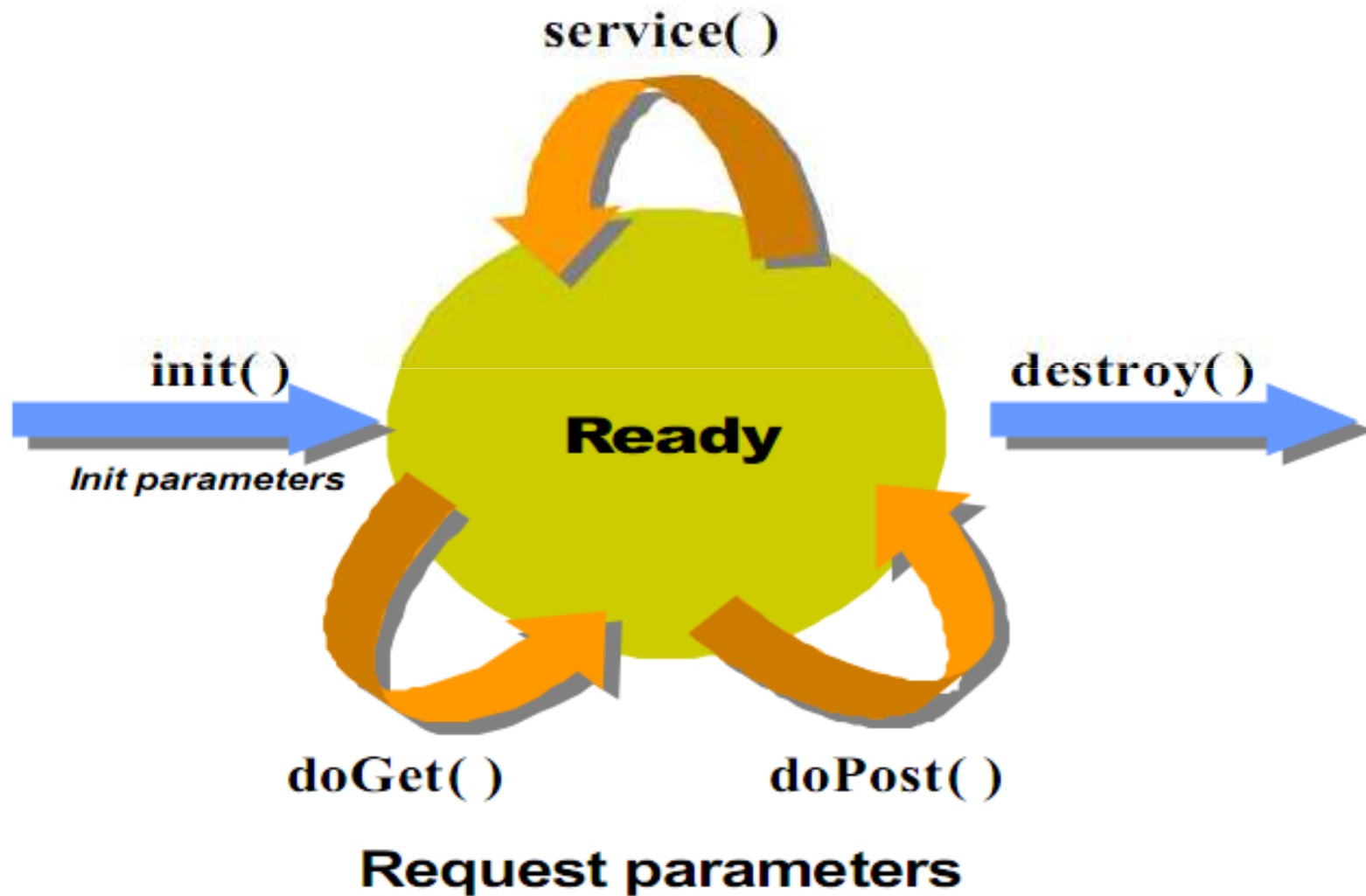
The background features a solid blue horizontal bar at the top and another at the bottom. The central area is white with a faint, artistic bokeh effect consisting of various sized grey and white circles and vertical lines, giving it a digital or network-like appearance.

CHU KỲ SỐNG CỦA SERVLET

Chu kỳ sống Servlet



Các phương thức trong chu kỳ sống Servlet



Các phương thức trong chu kỳ sống Servlet

- Được gọi thực hiện bởi Container
 - Container quản lý chu kỳ sống 1 Servlet
 - Được định nghĩa trong lớp đối tượng:
 - javax.servlet.GenericServlet
 - init()
 - destroy()
 - service () – Phương thức trừu tượng (abstract)
 - javax.servlet.http.HttpServlet
 - doGet(), doPost(), doXxx()
 - service () – Phương thức cài đặt lại (implementation)

Các phương thức trong chu kỳ sống Servlet

- `init()`
 - Được gọi duy nhất 1 lần khi thể hiện servlet được tạo ra
 - Thực hiện các cấu hình kết nối
 - Cấu hình kết nối cơ sở dữ liệu
- `destroy()`
 - Được gọi thực hiện trước khi thể hiện servlet kết thúc
 - Thực hiện các cấu hình đóng kết nối
 - Đóng kết nối cơ sở dữ liệu

Ví dụ init() trong 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.");  
    }  
    ...  
}
```

Ví dụ: init() đọc các Init Parameter

```
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();
    }
}
```

Cấu hình các Init Parameter trong 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>
```

Ví dụ: destroy()

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

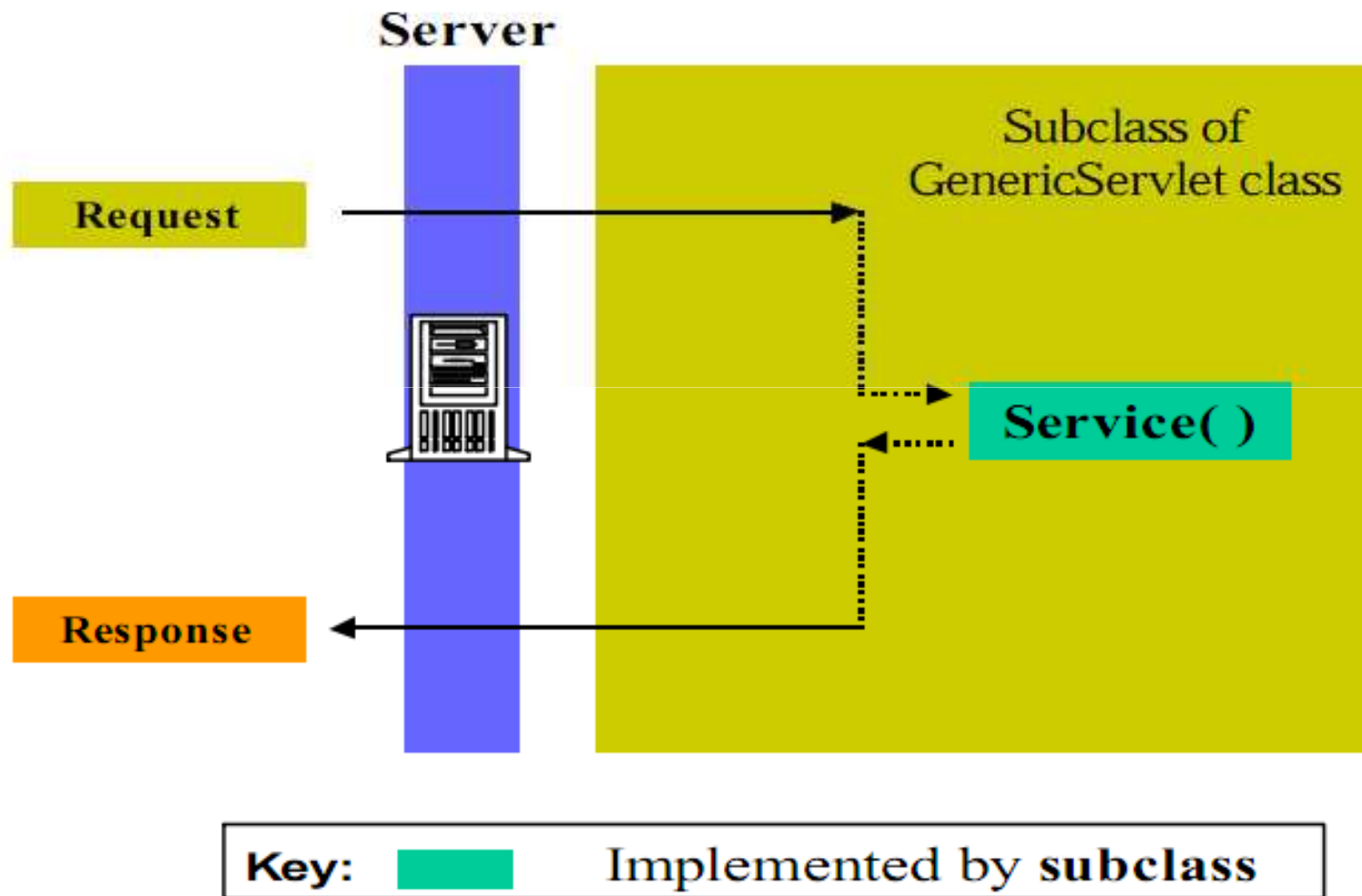

Các phương thức trong chu kỳ sống servlet

- `service()` - `javax.servlet.GenericServlet`
 - Phương thức trừu tượng (abstract)
- `service()` - `javax.servlet.http.HttpServlet`
 - Phương thức cài đặt lại (implementation)
 - Gọi phương thức `doGet()`, `doPost()`, ...
 - Không cài đặt lại phương thức này
- `doGet()`, `doPost()`, `doXxx()` -
 `javax.servlet.http.HttpServlet`
 - Xử lý HTTP GET, POST, ...
 - Cài đặt lại các phương thức này trong servlet

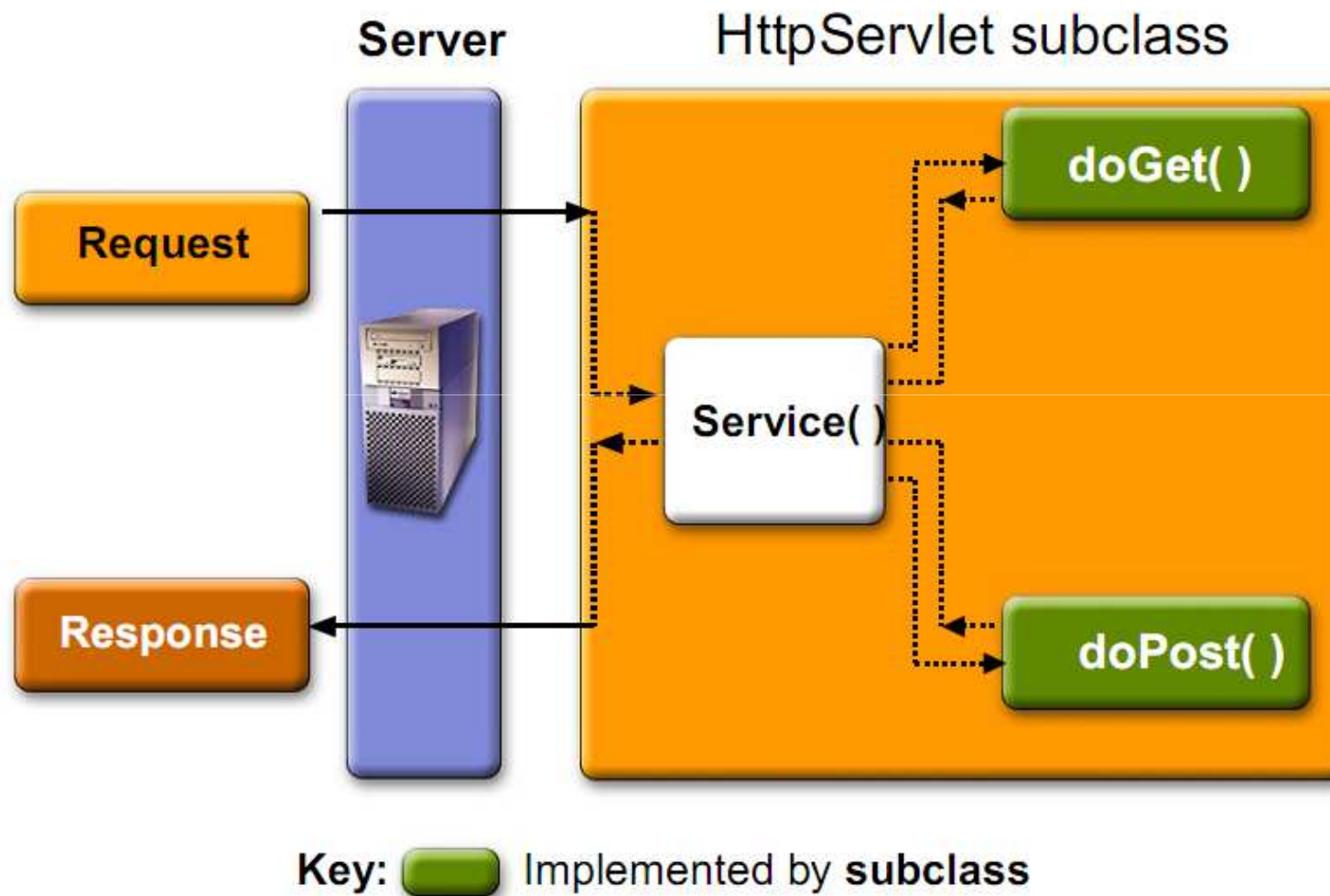
service() & doGet()/doPost()

- Các phương thức service() có tham số là request và response:
 - service(ServletRequest request, ServletResponse réponse)
- doGet() hoặc doPost() có tham số là HTTP Request và HTTP Response
 - doGet(HttpServletRequest request, HttpServletResponse response)
 - doPost(HttpServletRequest request, HttpServletResponse response)

Phương thức service()



Phương thức doGet() và doPost()



Những việc cần thực thi trong doGet() & doPost()

- Lấy thông tin từ client gửi đến thông qua `HttpRequest`
- Set(Save) và Get(read) các thuộc tính từ các đối tượng phạm vi (session,...)
- Thực thi các nghiệp vụ, ví dụ: thêm, xóa, sửa, rút trích dữ liệu từ csdl,...
- Chuyển các yêu cầu đến các Web Component khác (Servlet hoặc JSP)
- Tạo ra thông điệp HTTP response và gửi về client

Ví dụ 1: 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>");
    }
}
```

Ví dụ 2: 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);
}
```


Ví dụ 2: 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();
}
```


Các bước để tạo ra HTTP Response

- Tạo các Response Header
- Thiết lập các thuộc tính cho Response
 - Kích thước buffer
- Tạo 1 đối tượng Output Stream từ Response
- Ghi nội dung cho đối tượng Output Stream

Ví dụ: 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>");  
    }  
}
```

The background features a central white area with a faint grid of white dots of varying sizes. This central area is flanked by solid blue horizontal bars at the top and bottom. A thin vertical line is positioned on the left side, intersecting the blue bars and the central white area.

CÁC ĐỐI TƯỢNG PHẠM VI

Các đối tượng phạm vi

- Chia sẻ thông tin giữa các Web Component thông qua các thuộc tính được lưu giữ trong các đối tượng phạm vi
- Các thuộc tính được lưu giữ trong các đối tượng phạm vi được truy xuất với:
 - `setAttribute` và `getAttribute()`
- Bốn đối tượng phạm vi được định nghĩa:
 - `Web Context`, `Session`, `Request`, `Page`

Bốn đối tượng phạm vi: Phạm vi truy xuất

- Web Context (Servlet Context)
 - Truy xuất từ các web component trong một web context.
- Session
 - Truy xuất từ các web component cùng một session.
- Request
 - Truy xuất từ các web component xử lý cùng một request
- Page
 - Truy xuất từ trang JSP tạo ra nó

Bốn đối tượng phạm vi: Lớp

- Web context
 - `javax.servlet.ServletContext`
- Session
 - `javax.servlet.http.HttpSession`
- Request
 - `javax.servlet.http.HttpServletRequest`
- Page
 - `javax.servlet.jsp.PageContext`

The background of the slide features a central rectangular area with a bokeh effect, consisting of numerous out-of-focus white and grey circles of varying sizes. This central area is flanked by solid blue horizontal bars at the top and bottom. A thin vertical line is visible on the left side, passing through the blue bars and the central area.

WEB CONTEXT (SERVLET CONTEXT)

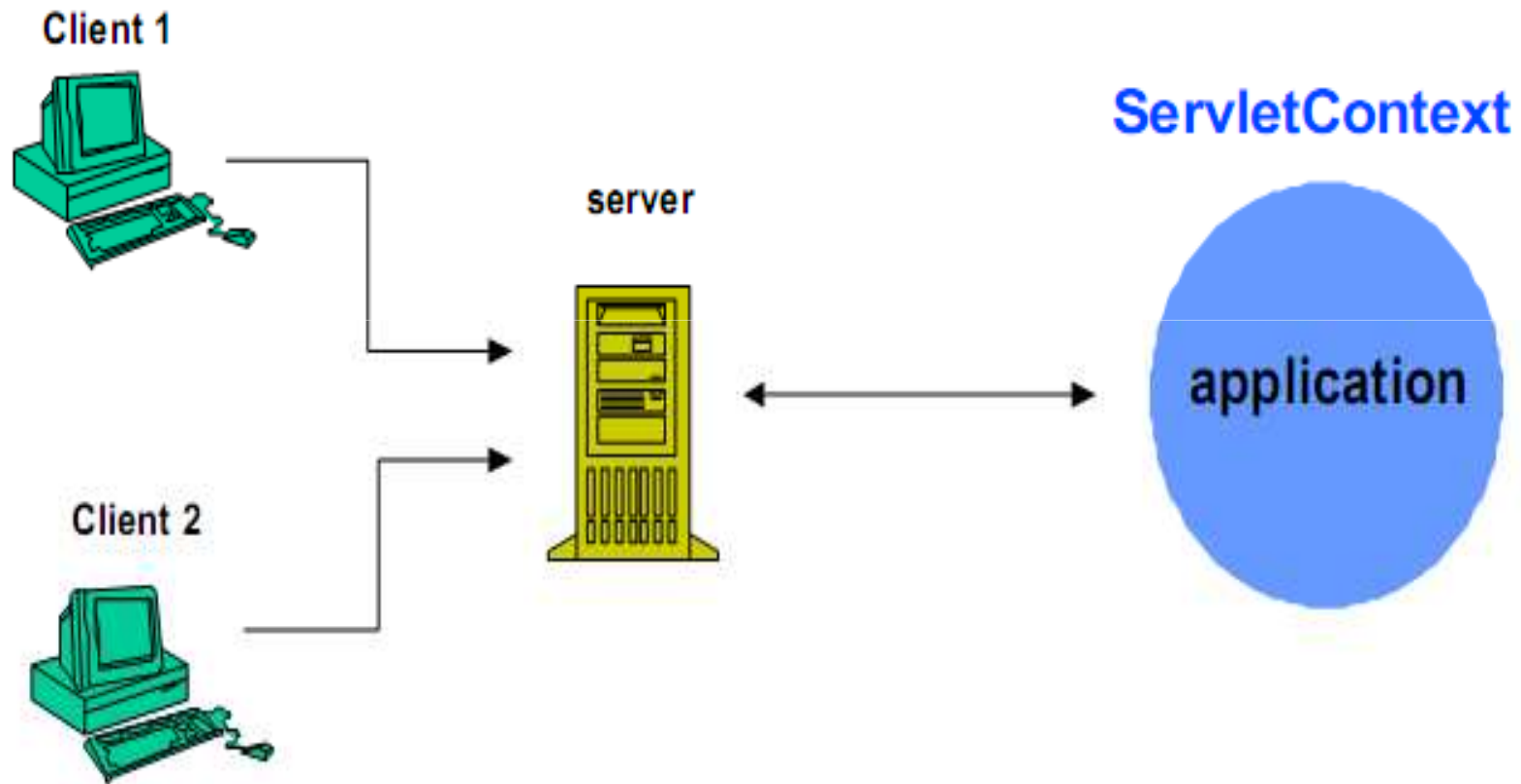
ServletContext dùng làm gì?

- Được sử dụng bởi các servlet
 - Lấy và gán các đối tượng dạng Context-wide (Application-wide)
 - Lấy đối tượng Request Dispatcher
 - Forward hoặc include web component
 - Truy xuất các tham số khởi tạo web context-wide được khai báo trong file web.xml
 - Truy xuất các web resource kết hợp với web context
 - Truy xuất đối tượng Log hoặc các thông tin hỗ trợ khác.

Phạm vi của ServletContext

- Phạm vi Context-wide (application-wide)
 - Được chia sẻ bởi tất cả các trang JSP và Servlet trong 1 ứng dụng web
 - Được gọi là phạm vi “web application”
 - Một ứng dụng web là tập các Servlet, JSP được triển khai trên 1 URL và có thể được cài đặt thông qua 1 file war.
 - Tất cả các Servlet trong ứng dụng web BookStore chia sẻ cùng đối tượng ServletContext
 - Mỗi ứng dụng web có duy nhất một đối tượng ServletContext trên một JVM

Phạm vi Web Application



Cách truy xuất đối tượng ServletContext

- Trong Servlet gọi phương thức:
 - `getServletContext()`
- Trong Servlet Filter gọi phương thức
 - `getServletContext()`
- ServletContext chứa trong đối tượng ServletConfig (web server cung cấp cho servlet khi nó được khởi tạo)
 - `Init(ServletConfig servletConfig)` trong Servlet

Ví dụ: Lấy thuộc tính từ 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.");  
    }  
}
```

Ví dụ: sử dụng đối tượng RequestDispatcher

```
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 =
        getServletContext().getRequestDispatcher("/banner");

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

    ...
}
```

Ví dụ: Tạo file Log

```
public void doGet (HttpServletRequest request,  
                  HttpServletResponse response)  
    throws ServletException, IOException {  
  
    ...  
    getServletContext().log("Life is good!");  
    ...  
    getServletContext().log("Life is bad!", someException);  
}
```

The slide features a central white area with a faint, abstract pattern of white dots and lines. This central area is framed by solid blue horizontal bars at the top and bottom. A thin vertical blue line is positioned on the left side of the white area.

Session (HttpSession)

Tại sao cần HttpSession?

- Cần một cơ chế lưu giữ trạng thái của client thông qua các yêu cầu từ cùng một browser.
 - Ví dụ: Online shopping card
- HTTP là phi trạng thái
- HttpSession lưu giữ trạng thái của client
 - Các servlet sử dụng HttpSession để gán và lấy các giá trị thuộc tính

Lấy đối tượng HttpSession

- Thông qua phương thức:
 - getSession() từ đối tượng request (HttpServletRequest)

Ví dụ: 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();  
    }  
}
```

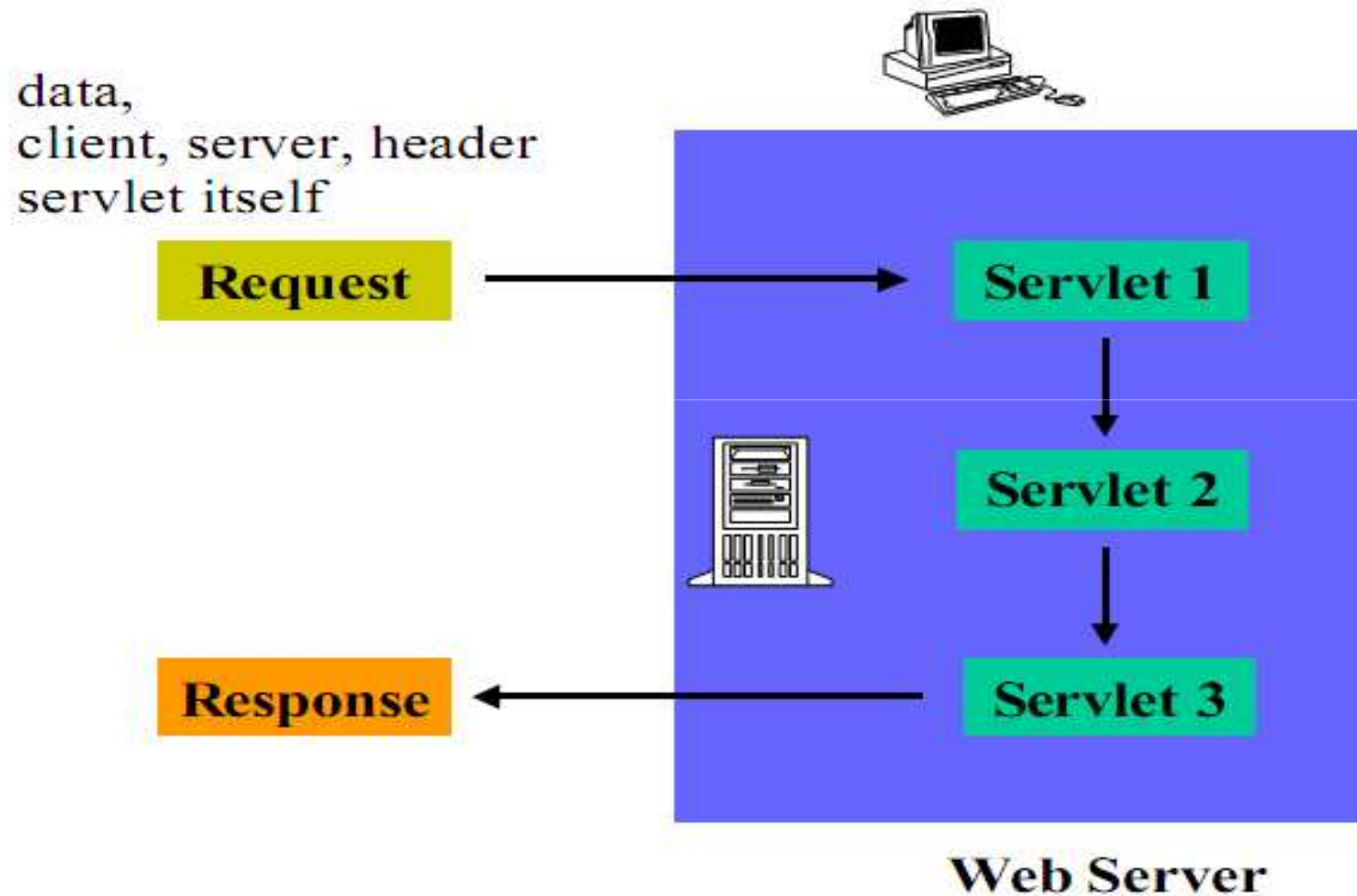
The background features a central white area with a bokeh effect of out-of-focus light circles. This central area is framed by solid blue horizontal bars at the top and bottom. A thin vertical line is positioned to the left of the text.

SERVLET REQUEST (HttpServletRequest)

Servlet Request là gì?

- Chứa dữ liệu truyền từ Client đến Server
- Tất cả các Servlet Request cài đặt interface ServletRequest, định nghĩa các phương thức cho việc truy cập:
 - Các tham số client gửi đến
 - Các thuộc tính
 - Locale
 - Client, Server
 - Input Stream
 - Thông tin về giao thức (Protocol)
 - Content Type
 - HTTPs

Request



Lấy các tham số client gửi đến

- Một Request có thể chứa nhiều tham số
- Các tham số gửi đến từ HTML Form
 - GET: Query String, thêm vào cuối URL
 - POST: dữ liệu được mã hóa, không thêm vào cuối URL
- `getParameter("paramName")`
 - Trả về giá trị của `paramName`
 - Trả về null nếu `paramName` không tồn tại
 - Sử dụng cho cả GET và POST Request

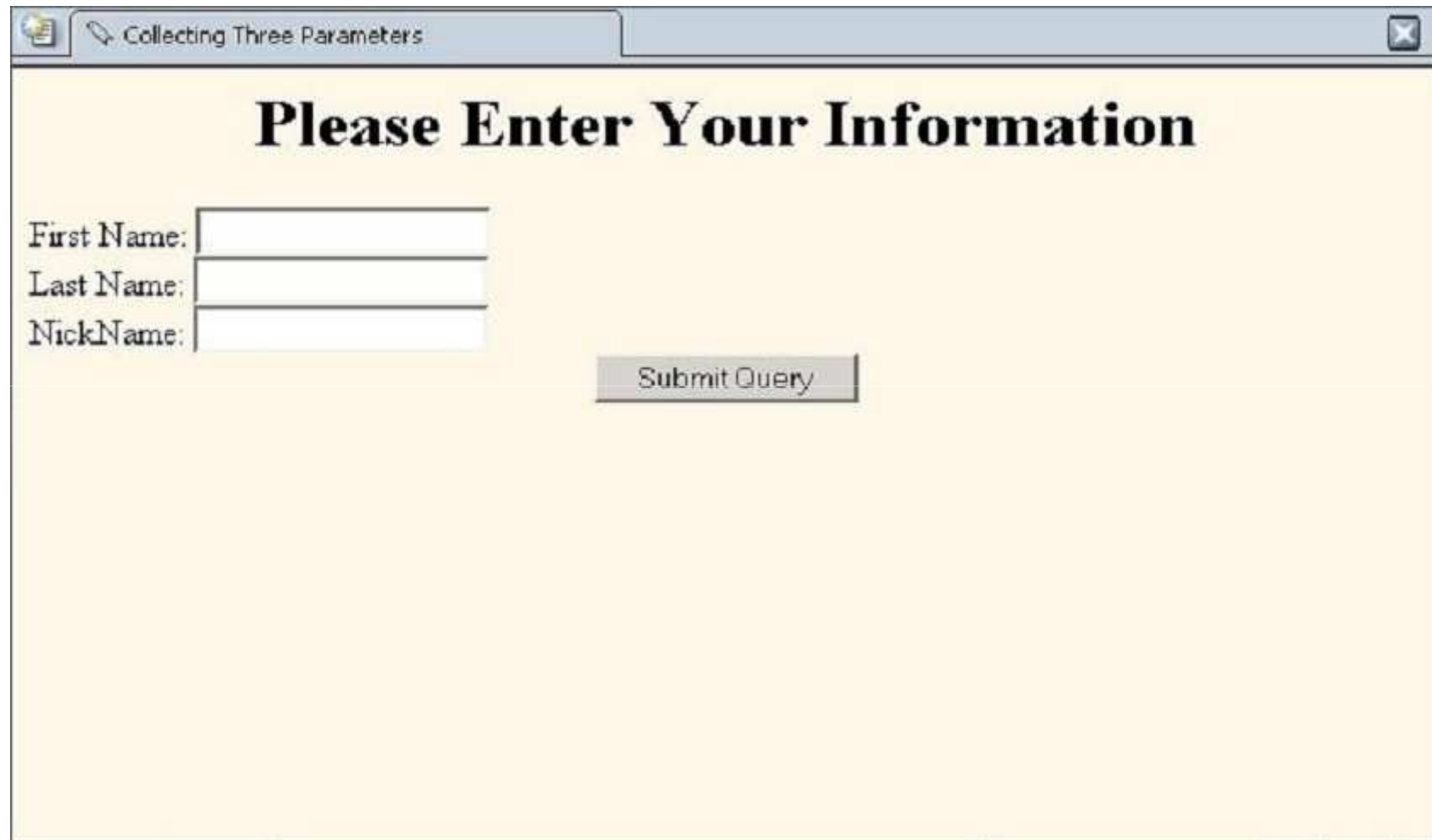
Ví dụ: HTML FORM sử dụng HTTP GET

```
<!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>
```

Ví dụ: HTML FORM sử dụng HTTP GET



The screenshot shows a web browser window with a single tab titled "Collecting Three Parameters". The page content is a form with a yellow background and the heading "Please Enter Your Information". The form contains three input fields labeled "First Name:", "Last Name:", and "NickName:". To the right of these fields is a "Submit Query" button.

Collecting Three Parameters

Please Enter Your Information

First Name:

Last Name:

NickName:

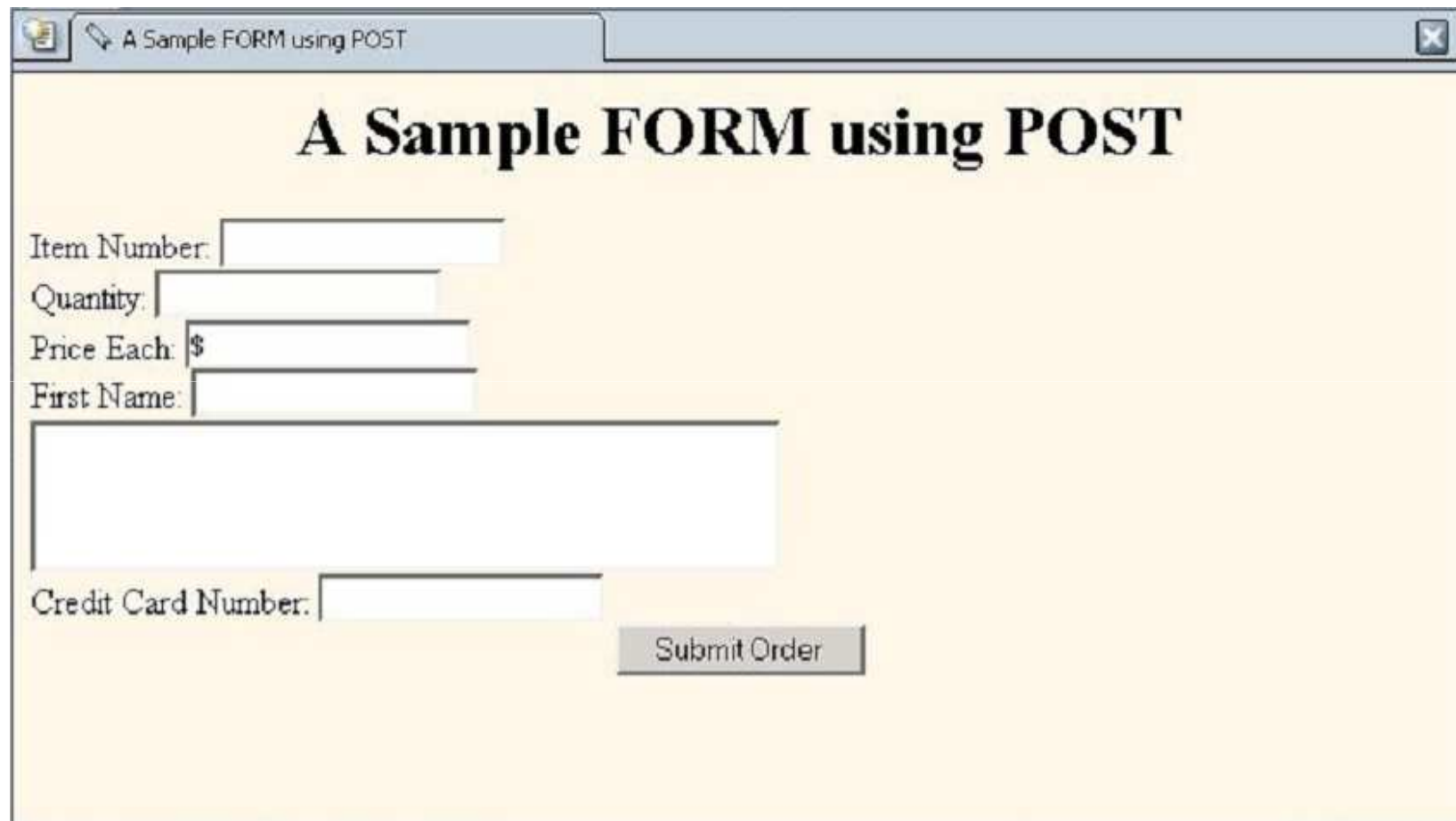
Ví dụ: Servlet – HTTP GET

```
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>Nick Name in Response</B>: "
            + request.getParameter("param3") + "\n" +
            "</UL>\n" +
            "</BODY></HTML>");
    }
}
```

Ví dụ: Form sử dụng HTTP 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>
```

Ví dụ: Form sử dụng HTTP POST



A screenshot of a web browser window with the title bar "A Sample FORM using POST". The page content is titled "A Sample FORM using POST" in a large, bold, black serif font. Below the title, there are five input fields arranged vertically on the left side of the page. The first four fields are labeled "Item Number:", "Quantity:", "Price Each:", and "First Name:" respectively. The "Price Each:" field has a dollar sign (\$) to its left. The fifth field is a larger text area. Below the text area, there is a label "Credit Card Number:" followed by a single-line input field. To the right of the "Credit Card Number:" field is a grey button with the text "Submit Order".

Item Number:

Quantity:

Price Each: \$

First Name:

Credit Card Number:

Ví dụ: Servlet – HTTP 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);
    }
}
```

Lấy thông tin của Client

- Servlet có thể lấy thông tin của client từ đối tượng request
 - String request.getRemoteAddr()
 - Lấy địa chỉ IP của client
 - String request.getRemoteHost()
 - Lấy tên host của client

Lấy thông tin Server

- Servlet có thể lấy thông tin của Server
 - String request.getServerName()
 - VD: www.sun.com
 - Int request.getServerPort()
 - VD: 8080

Lấy các thông tin hỗ trợ

- Input Stream
 - `ServletInputStream getInputStream()`
 - `java.io.BufferedReader getReader()`
- Protocol
 - `java.lang.String getProtocol()`
- Content Type
 - `java.lang.String getContentType()`
- HTTPs
 - `boolean isSecure()`

The background of the slide features a central white area with a faint, abstract pattern of white dots and vertical lines. This central area is framed by solid blue horizontal bars at the top and bottom. A thin vertical blue line is positioned on the left side, intersecting the top and bottom blue bars.

HttpServletRequest

HTTP Servlet Request?

- Chứa dữ liệu truyền từ HTTP Client đến HTTP Servlet
- Được Servlet Container tạo ra và truyền tham số đến phương thức doGet() và doPost() của Servlet
- HttpServletRequest kế thừa từ ServletRequest
 - Context, Servlet, Path, Query
 - Request Header
 - Security
 - Cookies
 - Session

HTTP Request URL

- Chứa chuỗi như sau:
 - `http://[Host]:[Port]/[Path]?[QueryString]`

HTTP Request URL: [Path]

- `http://[Host]:[Port]/[Path]?[QueryString]`
- Path bao gồm
 - Context: /<context of web app>
 - Servlet name: /<component alias>
 - Path: phần còn lại
- Ví dụ:
 - `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]/[Path]?[QueryString]`
- [Query String]: Tập các tham số và giá trị
- Có 2 cách phát sinh:
 - Xuất hiện tường minh trong trang web
 - `Add To Cart`
 - `String bookId = request.getParameter(Add);`
 - Thêm vào cuối URL khi một FORM với phương thức HTTP GET được Submit
 - `http://localhost/hello1/greeting?username=Monica+Clinton`
 - `String userName = request.getParameter(“username”)`

Context, Path, Query, Parameter

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

HTTP Request Header

- HTTP Request bao gồm các Header cung cấp thêm các thông tin về Request
- Ví dụ: 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 Header

- Accept
 - Các kiểu MIME browser có thể xử lý
- Accept-Encoding
 - Encoding(gzip,...) mà browser có thể xử lý
- Authorization
 - Định danh người dùng cho các trang bảo vệ mật khẩu
 - Thay thế cho HTTP Authorization, sử dụng HTML Form để gửi Username/password trong đối tượng session

HTTP Request Header

- Connection
 - Loại kết nối
 - Servlet nên setContentLength() để hỗ trợ Connection
- Cookie
 - Cookie gửi từ Server đến client, sử dụng getCookies()
- Host
 - Host được cung cấp trong URL
 - Yêu cầu trong HTTP 1.1

HTTP Request Header

- If-Modified-Since

- Client muốn trang nếu và chỉ nếu nó đã được thay đổi sau ngày chỉ định
- Không xử lý tình huống này một cách trực tiếp, gọi `getLastModified()` thay thế

- Referer

- URL của trang web đang tham chiếu
- Được sử dụng cho việc truy vết: Log bởi nhiều server

- User-Agent

- Chuỗi định danh Browser

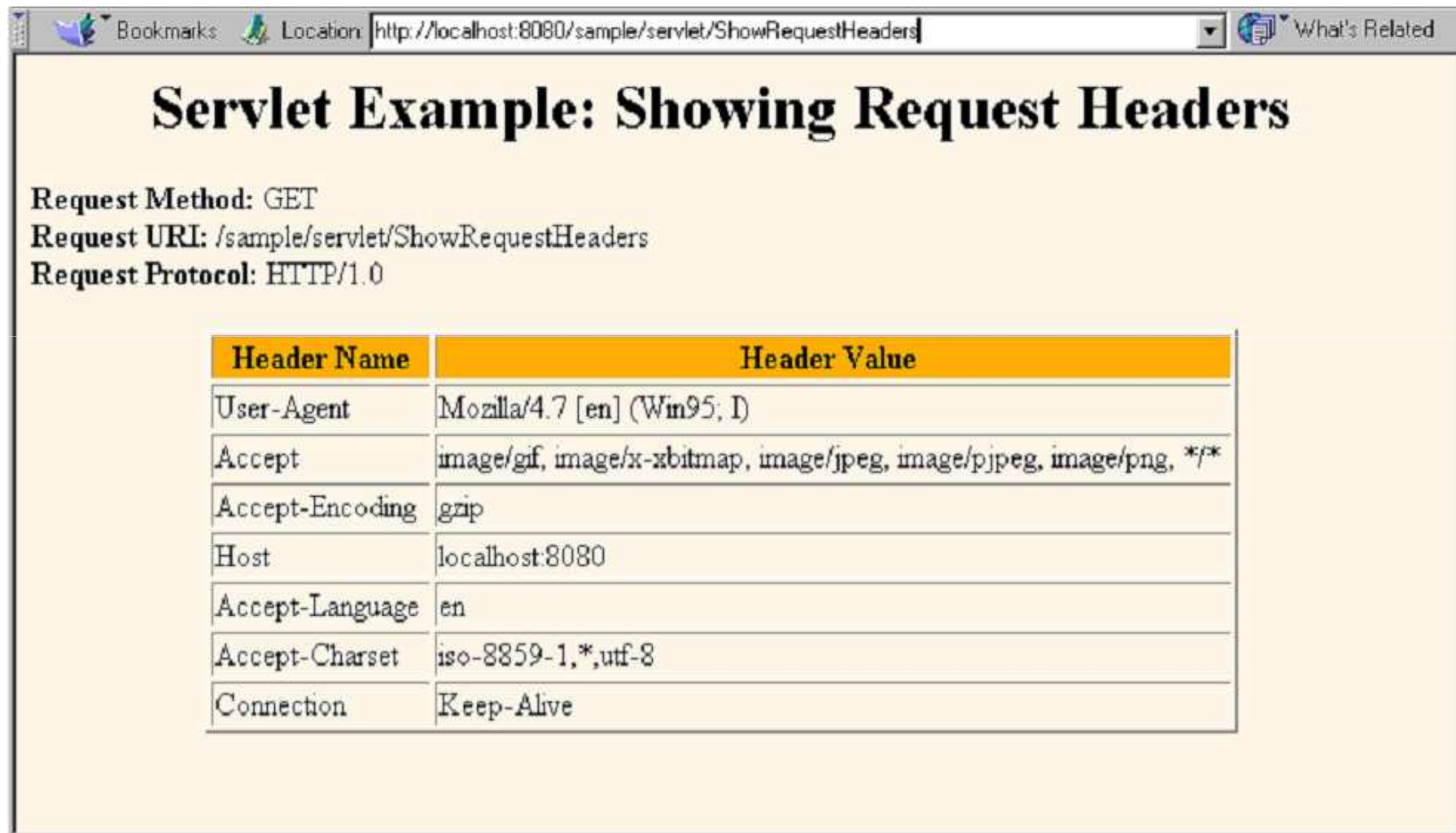
Các phương thức HTTP Header

- `String getHeader(java.lang.String name)`
- `java.util.Enumeration getHeaders(java.lang.String name)`
- `java.util.Enumeration getHeaderNames()`
- `int getIntHeader(java.lang.String name)`

Ví dụ: HTTP Request Header

```
//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));
        }
        ...
    }
}
```

Ví dụ: HTTP Request Header



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/sample/servlet/ShowRequestHeaders`. The page title is "Servlet Example: Showing Request Headers". Below the title, the following information is displayed:

- Request Method:** GET
- Request URL:** /sample/servlet/ShowRequestHeaders
- Request Protocol:** HTTP/1.0

A table with two columns, "Header Name" and "Header Value", lists the request headers:

Header Name	Header Value
User-Agent	Mozilla/4.7 [en] (Win95; D)
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

Authentication & User Security

- `String getRemoteUser()`
- `String getAuthType()`
- `boolean isUserInRole(String role)`

Cookie trong HttpServletRequest

- Cookie [] getCookies()
 - Mảng chứa tất cả các đối tượng Cookie mà Client gửi cùng với Request

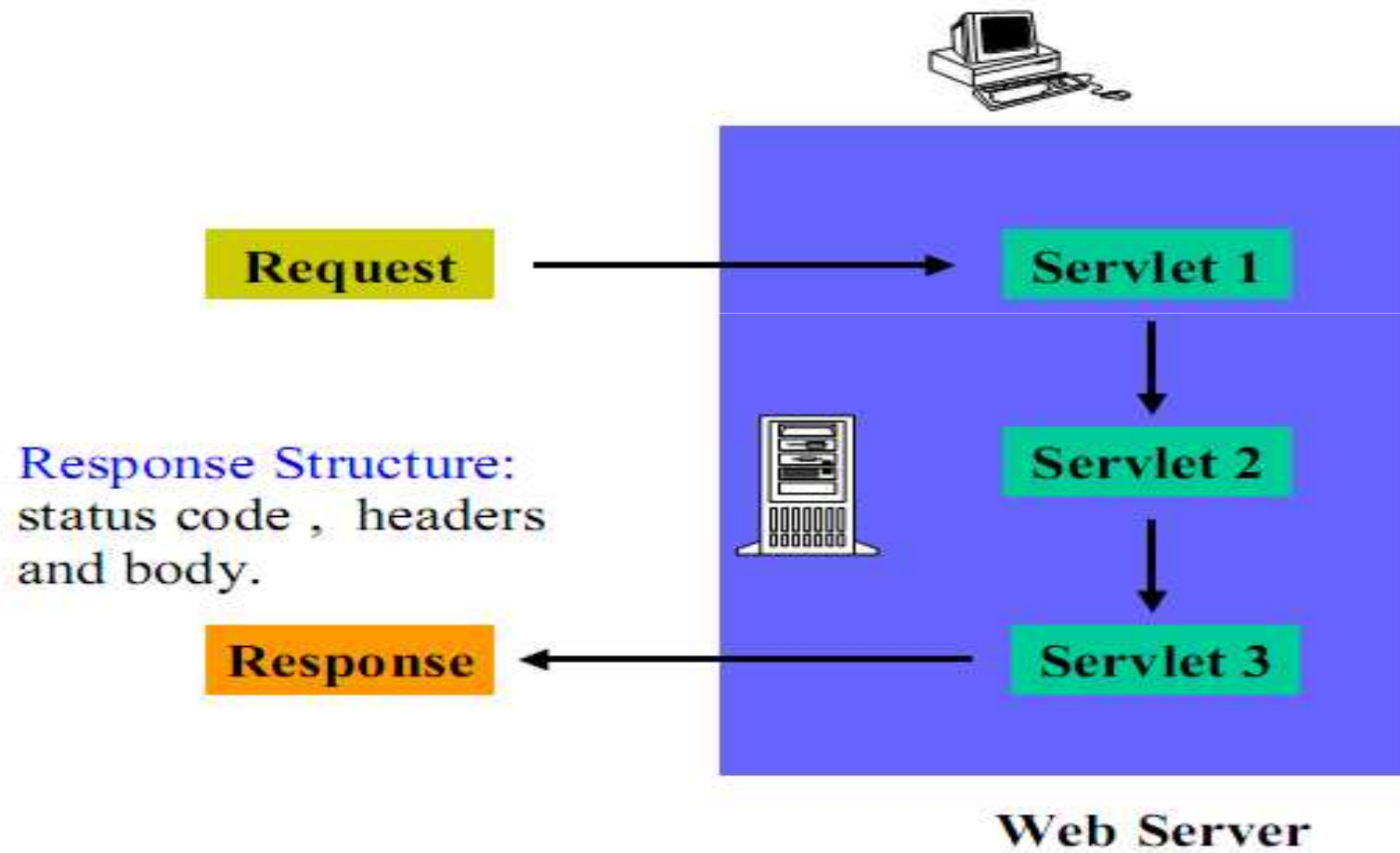
The background features a central white area with a bokeh effect of out-of-focus light circles. This is framed by solid blue horizontal bars at the top and bottom. A thin vertical line is positioned to the left of the text.

Servlet Response (HttpServletResponse)

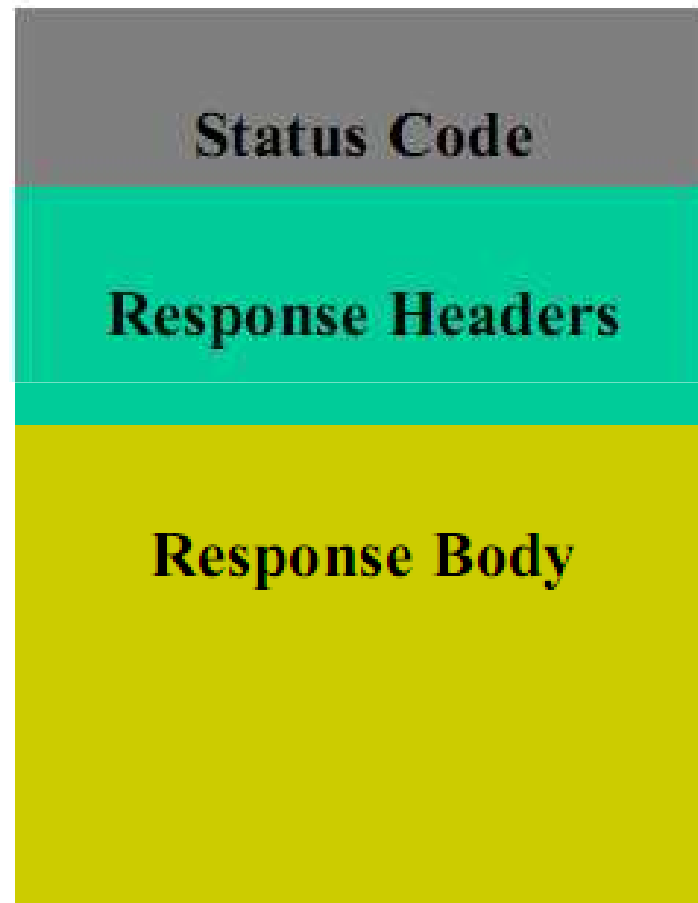
Servlet Response

- Chứa dữ liệu truyền từ Servlet đến client
- Các Servlet cài đặt interface ServletResponse
 - Output Stream
 - Content Type
 - Buffer Output
 - Locale
- HttpServletResponse kế thừa ServletResponse
 - Http Response
 - Cookies

Response



Response Structure



The background features a solid blue horizontal bar at the top and another at the bottom. The central area is white with a faint, abstract bokeh effect consisting of various sized grey and white circles. A thin vertical blue line is positioned on the left side, intersecting the top and bottom blue bars.

STATUS CODE (HTTP RESPONSE)

Http Response Status Code

- Tại sao lại cần HTTP Response Status Code?
 - Chuyển Request của Client đến trang web khác
 - Chỉ định tài nguyên bị bỏ lỡ
 - Hướng dẫn browser sử dụng cached copy

Thiết lập HTTP Response Status Code

- `public void setStatus(int statusCode)`
 - Status code được định nghĩa trong `HttpServletResponse`
- Status code thuộc 1 trong các khoảng:
 - 100-199 Informational
 - 200-299 Successful
 - 300-399 Redirection
 - 400-499 Incomplete
 - 500-599 Server Error
- Status Code mặc định là 200 (OK)

Ví dụ: Http Response Status

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

Các Status Code thông dụng

- 200 (SC_OK)
 - Thành công & Có nội dung theo kèm
 - Được sử dụng mặc định bởi servlet
- 204 (SC_No_CONTENT)
 - Thành công & Không có nội dung theo kèm
 - Browser nên giữ lại nội dung hiển thị trước.
- 301 (SC_MOVED_PERMANENTLY)
 - Nội dung được chuyển lâu
 - Browse chuyển sang Location mới tự động

Các Status Code thông dụng

- 302 (SC_MOVED_TEMPORARILY)
 - Thông điệp được tìm thấy
 - Servlet nên sử dụng sendRedirect() thay cho setStatus() khi thiết lập Header
- 401 (SC_UNAUTHORIZED)
 - Browser truy xuất password mà ko có Authorization Header
- 404(SC_NOT_FOUND)
 - Trang không tìm thấy

Phương thức gửi lỗi

- Các status code từ 400-599 có thể được sử dụng trong phương thức `sendError()`
- `Public void sendError(int sc)`
 - Server gửi 1 lỗi chỉ định bởi status code
- `Public void sendError(int code, String message)`
 - Gửi status code kèm 1 message

The background features a central white area with a bokeh effect of out-of-focus light circles. This central area is framed by solid blue horizontal bars at the top and bottom. A thin vertical blue line is positioned to the left of the text.

HEADER (HTTP RESPONSE)

Tại sao cần Http Response Header

- Chuyển Location
- Chỉ định các cookie
- Cung cấp này sửa đổi trang web
- Chỉ định cho browser nạp lại trang web sau 1 thời gian t
- Cung cấp kích thước file để Http Connection sử dụng
- Kiểu của tài liệu được phát sinh

Thiết lập Response Header

- `public void setHeader(String headerName, String headerValue)`
- `public void setDateHeader(String name, long millisecs)`
- `public void setIntHeader(String name, int headerValue)`
- `addHeader, addDateHeader, addIntHeader`

Thiết lập Response Header

- `setContentType`
- `setContentLength`
- `addCookie`
- `sendRedirect`

HTTP 1.1 Response Header

- Location
- Refresh
- Set-Cookie
- Cache-Control (1.1) & Pragma (1.0)
- Content-Encoding
- Content-Length
- Content-Type
- Last-Modified

Ví dụ: Refresh Page

```
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());  
    }  
}
```

The image features a central white rectangular area containing text. This area is flanked by solid blue horizontal bars at the top and bottom. The background within the white area is a dark, abstract composition of out-of-focus light circles (bokeh) and faint, vertical lines of small white dots, resembling a digital or data-themed aesthetic.

BODY (HTTP RESPONSE)

Ghi nội dung Response Body

- Servlet thường trả kết quả là một Response Body
- Response Body có thể là `PrintWriter` hoặc `ServletOutputStream`
- `PrintWriter`
 - `Response.getWriter()`
 - Kết xuất là ký tự
- `ServletOutputStream`
 - `response.getOutputStream()`
 - Dành cho dữ liệu nhị phân (ảnh)

The image features a central white rectangular area containing the text 'XỬ LÝ LỖI'. This area is flanked by two solid blue horizontal bars, one at the top and one at the bottom. The background of the white area is a grayscale bokeh effect with various sized light circles and a faint grid pattern.

XỬ LÝ LỖI

Xử lý lỗi

- Web Container phát sinh trang lỗi mặc định
- Có thể chỉ định trang khác để hiển thị
- Các bước xử lý lỗi
 - Tạo các trang html với các lỗi cần hiển thị
 - Hiệu chỉnh web.xml cho phù hợp

Ví dụ: Thiết lập trang lỗi trong 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>
```

Ví dụ: Thiết lập trang lỗi trong 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>
```

The background features a central white area with a pattern of grey dots of varying sizes, some arranged in vertical columns. This central area is flanked by solid blue horizontal bars at the top and bottom. A thin vertical line is visible on the left side, separating the blue bar from the white area.

CÁC TÀI NGUYÊN

Các tài nguyên sử dụng trong bài giảng

- Java Passion (J2EE)
- Core Web Programming