




Bộ môn Công nghệ Phần mềm
Viện CNTT & TT
Trường Đại học Bách Khoa Hà Nội

LẬP TRÌNH WEB HƯỚNG JAVA

Bài 08: Cơ bản về Servlets

Giảng viên: ThS. Trịnh Tuấn Đạt
Bộ môn CNPM
Email: trinhthuandat.bk@gmail.com/dattt@soict.hut.edu.vn

Nội dung

- 1. Servlet trong kiến trúc J2EE
- 2. Mô hình Servlet Request & response
- 3. Vòng đời của Servlet
- 4. Servlet scope objects
- 5. Servlet request
- 6. Servlet response: Status, Header, Body
- 7. Xử lý lỗi (Error)

DatTT-DSE-SOICT-HUST

2

Chủ đề nâng cao (Học trong bài Advanced Servlet ở sau)

- Theo vết Session (Session Tracking)
- Servlet Filters
- Xử lý sự kiện trong vòng đời Servlet
- Kỹ thuật Include, forward, và redirect
- Điều khiển tương tranh
- Invoker Servlet

DatTT-DSE-SOICT-HUST

3

1. Servlet trong kiến trúc J2EE

DatTT-DSE-SOICT-HUST

4

Kiến trúc J2EE 1.2

Một công nghệ Web mở rộng, kết hợp với các đối tượng Java, trả về nội dung động cho client dưới dạng HTML hoặc XML. Client thường là Web Browser

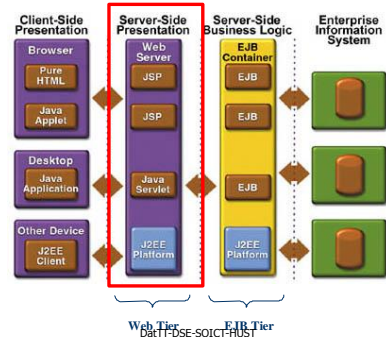
Java Servlet: 1 chương trình Java, mở rộng chức năng 1 web server, sinh nội dung động và tương tác với web clients sử dụng mô hình request-response



DatTT-DSE-SOICT-HUST

5

Vị trí của Servlet và JSP?

Web Tier EJB Tier
DatTT-DSE-SOICT-HUST

6

Servlet là gì?

- Các đối tượng Java™, mở rộng chức năng của 1 HTTP server.
- Được ánh xạ (mapped) với 1 URL và được quản lý bởi **container** tương ứng
- Chạy được trên tất cả các **web servers** và các **app servers** chuẩn
- Không phụ thuộc vào Platform hoặc server

DatTT-DSE-SOICT-HUST

7

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

DatTT-DSE-SOICT-HUST

8

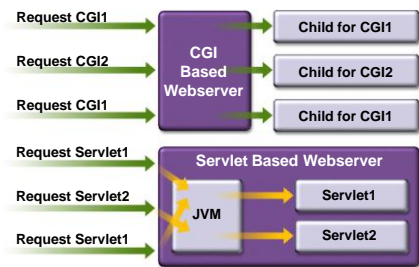
CGI vs. Servlet

CGI	Servlet
<ul style="list-style-type: none"> • 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 	<ul style="list-style-type: none"> • Written in Java • Powerful, reliable, and efficient • Improves scalability, reusability (component based) • Leverages built-in security of Java programming language • Platform independent and portable

DatTT-DSE-SOICT-HUST

9

Servlet vs. CGI



DatTT-DSE-SOICT-HUST

10

Ưu điểm của Servlet

- Không mắc phải các hạn chế của CGI
- Có rất nhiều các tools phía third-party và nhiều Web servers hỗ trợ Servlet
- Truy cập được mọi Java APIs
- Độ tin cậy cao (**Reliable**), hiệu năng tốt, dễ mở rộng (**scalability**)
- Độc lập Platform & server
- An toàn (**Secure**)
- Hầu hết server cho phép tự động load lại Servlet khi có thay đổi

DatTT-DSE-SOICT-HUST

11

Công nghệ JSP là gì?

- Cho phép **phân tách** tầng business logic với tầng presentation
 - Presentation trình bày dưới dạng HTML hoặc XML/XSLT
 - Business logic được cài đặt trong các **Java Beans** hoặc các thẻ mở rộng (**custom tags**)
 - Dễ bảo trì, tái sử dụng
- Dễ mở rộng nhờ các custom tags
- Xây dựng trên nền tảng công nghệ Servlet

DatTT-DSE-SOICT-HUST

12

Trang JSP?

- Một tài liệu text, cho phép trả về nội dung động cho trình duyệt phía client
- Bao gồm cả nội dung tĩnh và nội dung động
 - Nội dung tĩnh: HTML, XML
 - Nội dung động: mã lập trình, JavaBeans, custom tags

DatTT-DSE-SOICT-HUST

13

Ví dụ 1 trang 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>
```

DatTT-DSE-SOICT-HUST

14

So sánh Servlets và JSP

Servlets	JSP
<ul style="list-style-type: none"> • HTML code in Java • Any form of Data • Not easy to author a web page 	<ul style="list-style-type: none"> • Java-like code in HTML • Structured Text • Very easy to author a web page • Code is compiled into a servlet

DatTT-DSE-SOICT-HUST

15

Ưu điểm của JSP

- Tách biệt nội dung và cách thức trình bày
- Đơn giản hóa việc phát triển với JavaBeans và custom tags
- Hỗ trợ tái sử dụng nhờ sử dụng các components
- Tự động biên dịch lại khi có thay đổi mã nguồn
- Độc lập Platform
- Dễ viết

DatTT-DSE-SOICT-HUST

16

Khi nào sử dụng Servlet thay cho JSP

- Mở rộng chức năng của 1 Web server, như hỗ trợ định dạng file mới
- Sinh các đối tượng không chứa HTML như graphs hoặc pie charts
- Lưu ý: **TRÁNH** trả về mã HTML từ servlets, nếu có thể

DatTT-DSE-SOICT-HUST

17

Sử dụng Servlet hay JSP?

- Trong thực tế, servlet và JSP đều được sử dụng
 - Trong kiến trúc MVC (Model, View, Controller)
 - Servlet xử lý Controller
 - JSP xử lý View

DatTT-DSE-SOICT-HUST

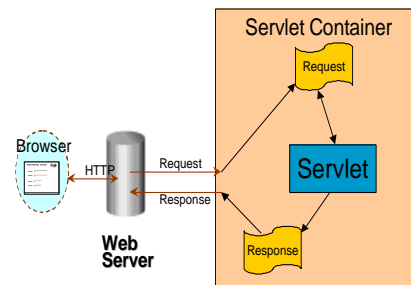
18

2. Mô hình Servlet Request & response

DatTT-DSE-SOICT-HUST

19

Mô hình Servlet Request & response



DatTT-DSE-SOICT-HUST

20

Nhiệm vụ của Servlet?

- Nhận client request (hầu hết ở dạng HTTP request)
- Trích xuất 1 số thông tin từ request
- Xử lý nghiệp vụ (truy cập DB, gọi EJBs, ...) hoặc sinh động nội dung
- Tạo và gửi trả response cho client (hầu hết ở dạng HTTP response) hoặc forward request cho servlet khác/cho trang JSP

DatTT-DSE-SOICT-HUST

21

Requests và Responses

- Request là gì?
 - Thông tin được gửi từ client tới 1 server
 - Ai tạo ra request
 - Dữ liệu gì được user nhập vào và gửi đi
 - HTTP headers
- Response là gì?
 - Thông tin được gửi đến client từ 1 server
 - Dữ liệu Text (html, thuần text) hoặc dữ liệu binary (image)
 - HTTP headers, cookies, ...

DatTT-DSE-SOICT-HUST

22

HTTP

- HTTP request bao gồm
 - header
 - Phương thức
 - Get: Thông tin nhập vào trong form được truyền như 1 phần của URL
 - Post: Thông tin nhập vào trong form được truyền trong nội dung thông điệp (**message body**)
 - Put
 - Header
 - Dữ liệu trong request (**request data**)

DatTT-DSE-SOICT-HUST

23

HTTP GET và POST

- Các client requests thông dụng nhất
 - HTTP GET & HTTP POST
- GET requests:
 - Thông tin người dùng nhập vào **đính kèm** trong URL dưới dạng 1 query string
 - Chỉ gửi được lượng dữ liệu giới hạn
 - `.../servlet/ViewCourse?FirstName=Sang&LastName=Shin`
- POST requests:
 - Thông tin người dùng nhập vào được gửi dưới dạng dữ liệu (không đính kèm vào URL)
 - Gửi được lượng dữ liệu bất kỳ

DatTT-DSE-SOICT-HUST

24

Ví dụ Servlet

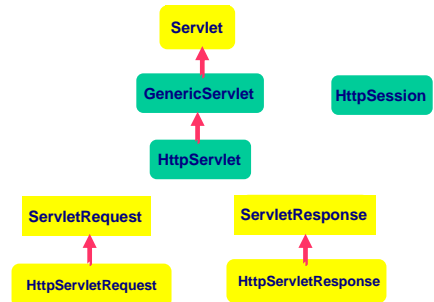
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

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>");
    }
}
```

DatTT-DSE-SOICT-HUST

25

Các lớp và giao diện Servlet



DatTT-DSE-SOICT-HUST

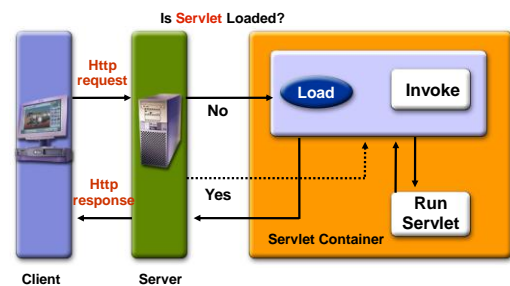
26

3. Vòng đời của Servlet

DatTT-DSE-SOICT-HUST

27

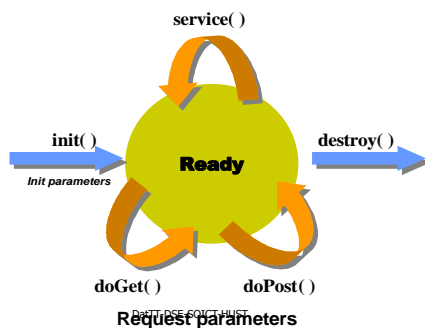
Vòng đời Servlet



DatTT-DSE-SOICT-HUST

28

Các phương thức trong vòng đời Servlet



DatTT-DSE-SOICT-HUST

29

Các phương thức trong vòng đời Servlet

- Được gọi bởi container
 - Container điều khiển vòng đời của 1 servlet
- Định nghĩa trong:
 - Lớp `javax.servlet.GenericServlet`
 - `init()`
 - `destroy()`
 - `service()` - là phương thức **abstract**
 - Lớp `javax.servlet.http.HttpServlet`
 - `doGet()`, `doPost()`, `doXXX()`
 - `service()` - implementation

DatTT-DSE-SOICT-HUST

30

Các phương thức trong vòng đời Servlet

- **init()**
 - Được gọi **MỘT** lần khi servlet được tạo thể hiện hóa lần đầu tiên
 - Thực hiện các **khởi tạo** trong phương thức này
 - Ví dụ: tạo 1 kết nối CSDL
- **destroy()**
 - Được gọi trước khi hủy 1 servlet instance
 - Thực hiện thao tác dọn dẹp
 - Ví dụ: đóng kết nối CSDL đã mở

DatTT-DSE-SOICT-HUST

31

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

DatTT-DSE-SOICT-HUST

32

Ví dụ: init() đọc tham số cấu hình

```
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();
    }
}
```

DatTT-DSE-SOICT-HUST

33

Thiết lập các tham số 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:rfmi:CloudscapeDB
      </param-value>
    </init-param>
  </servlet>
</web-app>
```

DatTT-DSE-SOICT-HUST

34

Ví dụ: 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;
    }
    ...
}
```

DatTT-DSE-SOICT-HUST

35

Các phương thức trong vòng đời Servlet

- **service()** trong `javax.servlet.GenericServlet`
 - Phương thức **Abstract**
- **service()** trong lớp `javax.servlet.http.HttpServlet`
 - Phương thức cụ thể (đã cài đặt)
 - gọi tới (**dispatch**) `doGet()`, `doPost()`
 - **KHÔNG** override phương thức này!
- **doGet()**, **doPost()**, **doXxx()** trong `javax.servlet.http.HttpServlet`
 - Xử lý các HTTP GET, POST requests
 - Lập trình viên override những phương thức này trong servlet của mình để có xử lý phù hợp

DatTT-DSE-SOICT-HUST

36

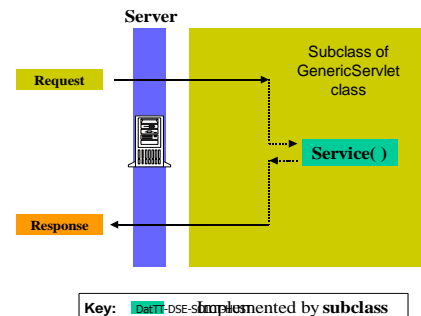
service() & doGet()/doPost()

- Phương thức **service()** nhận các requests và responses tổng quát:
 - `service(ServletRequest request, ServletResponse response)`
- doGet()** và **doPost()** nhận các HTTP requests và responses:
 - `doGet(HttpServletRequest request, HttpServletResponse response)`
 - `doPost(HttpServletRequest request, HttpServletResponse response)`

DatTT-DSE-SOICT-HUST

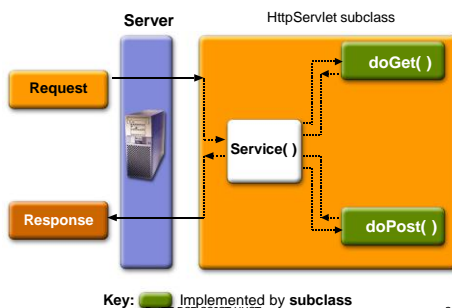
37

Phương thức Service()



38

Phương thức doGet() và doPost()



DatTT-DSE-SOICT-HUST

39

Những việc cần làm trong doGet() & doPost()

- Trích xuất các thông tin gửi từ client (HTTP parameter) từ HTTP request
- Thiết lập/truy cập các thuộc tính của các **Scope objects**
- Thực hiện các xử lý nghiệp vụ (**business logic**) hoặc truy cập CSDL
- Tùy chọn forward request tới các Web components khác (Servlet hoặc JSP)
- Sinh HTTP response và trả về cho client

DatTT-DSE-SOICT-HUST

40

Ví dụ 1 phương thức doGet() đơn giản

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

DatTT-DSE-SOICT-HUST

41

Ví dụ 1 phương thức doGet() phức tạp (1/2)

```
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);
}
```

DatTT-DSE-SOICT-HUST

42

Ví dụ 1 phương thức doGet() phức tạp (2/2)

```
// 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();
```

DatTT-DSE-SOICT-HUST

43

Các bước tạo một HTTP Response

- Điền vào 1 số Response headers (ví dụ: **content type**)
- Thiết lập 1 số đặc tính của response
 - Kích thước Buffer
- Lấy 1 đối tượng **output stream** từ response đang xét
- Viết nội dung cần trả về cho client vào output stream

DatTT-DSE-SOICT-HUST

44

Ví dụ một Response đơn giản

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

DatTT-DSE-SOICT-HUST

45

4. Scope Objects

Scope Objects

- Thông tin được **chia sẻ** giữa các web components thông qua các thuộc tính (**Attributes**) của các **Scope objects**
 - Các thuộc tính là các cặp **name/object**
- Các thuộc tính được tham chiếu trong các Scope objects thông qua phương thức
 - `getAttribute()` & `setAttribute()`
- 4 loại Scope objects được định nghĩa
 - **Web context, session, request, page**

DatTT-DSE-SOICT-HUST

47

4 loại Scope Objects: giới hạn truy cập

- Web context (ServletConext)
 - Truy cập từ các Web components trong 1 Web context
- Session
 - Truy cập từ các Web components xử lý request trong 1 session
- Request
 - Truy cập từ các Web components xử lý request đó
- Page
 - Truy cập từ trang JSP tạo ra object đó

DatTT-DSE-SOICT-HUST

48

4 loại Scope Objects: các Class tương ứng

- 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`

DatTT-DSE-SOICT-HUST

49

4.1. Web Context (ServletContext)

DatTT-DSE-SOICT-HUST

50

ServletContext dùng để làm gì?

- Được sử dụng bởi Servlet để:
 - Thiết lập các thuộc tính có tầm vực context (trong toàn ứng dụng)
 - Lấy ra đối tượng request dispatcher
 - Forward hoặc include các web component khác
 - Truy cập các tham số khởi tạo tầm vực Web context thiết lập trong file `web.xml`
 - Truy cập các tài nguyên Web kết hợp với Web context
 - Ghi Log
 - Truy cập các thông tin khác

DatTT-DSE-SOICT-HUST

51

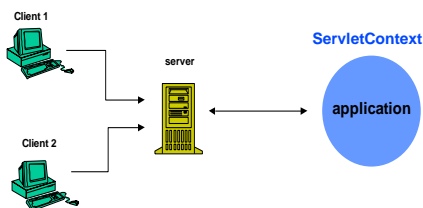
Tầm vực (Scope) của ServletContext

- Có tầm vực context (**Context-wide scope**)
 - Được chia sẻ bởi tất cả các servlets và các trang JSP trong cùng 1 "web application"
 - Ví thể còn gọi là "web application scope"
 - Một "web application" là 1 tập các servlets và các content khác, chung 1 phần URL, và có thể cài đặt qua 1 file `*.war`
 - Có duy nhất 1 đối tượng ServletContext cho mỗi "web application" trên mỗi Java Virtual Machine

DatTT-DSE-SOICT-HUST

52

ServletContext: Web Application Scope



DatTT-DSE-SOICT-HUST

53

Truy cập tới đối tượng ServletContext như thế nào?

- Trong code **servlet** hoặc code **servlet filter**, gọi hàm `getServletContext()`
- Trong đối tượng ServletConfig cũng chứa đối tượng ServletContext
 - Web server cung cấp ServletConfig cho mỗi servlet khi khởi tạo nó: trong giao diện Servlet `init (ServletConfig servletConfig)`

DatTT-DSE-SOICT-HUST

54

Ví dụ: Lấy giá trị của 1 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
            ServletException("Couldn't get database.");
    }
}
```

DatTT-DSE-SOICT-HUST

55

Ví dụ: lấy ra và 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 =
        session.getServletContext().getRequestDispatcher("/banner");

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

DatTT-DSE-SOICT-HUST

56

Ví dụ: Logging

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

DatTT-DSE-SOICT-HUST

57

4.2. Session (HttpSession)

Trình bày chi tiết hơn trong bài
"Session Tracking"

DatTT-DSE-SOICT-HUST

58

Tại sao cần HttpSession?

- Cần 1 cơ chế để lưu trữ trạng thái client theo thời gian sau 1 loạt các request từ cùng 1 người dùng (cùng 1 trình duyệt)
 - Ví dụ: Giỏ hàng (**Online shopping cart**)
- HTTP là giao thức phi trạng thái (**stateless**)
- HttpSession lưu trữ (**maintain**) trạng thái client
 - Sử dụng bởi các Servlets để set và get giá trị các thuộc tính có tầm vực session

DatTT-DSE-SOICT-HUST

59

Lấy ra đối tượng HttpSession?

- Qua phương thức getSession() của 1 đối tượng Request (HttpServletRequest)

DatTT-DSE-SOICT-HUST

60

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();
    }
}
```

DatTT-DSE-SOICT-HUST

61

5. Servlet request (HttpServletRequest)

DatTT-DSE-SOICT-HUST

62

Servlet Request là gì?

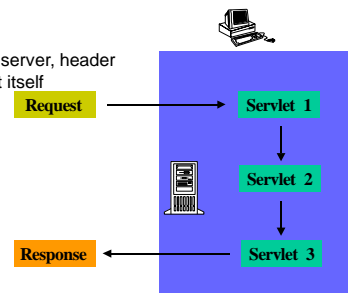
- Chứa dữ liệu gửi từ client đến servlet
- Tất cả các servlet requests đều thực thi giao diện `ServletRequest` định nghĩa các phương thức truy cập tới:
 - Các tham số (parameters) gửi từ clients
 - Object-valued attributes
 - Client và server
 - Input stream
 - Thông tin về giao thức (Protocol information)
 - Content type
 - request có được tạo trên 1 kênh truyền secure không (secure channel. Ví dụ: HTTPS)

DatTT-DSE-SOICT-HUST

63

Requests

data,
client, server, header
servlet itself



DatTT-DSE-SOICT-HUST

Web Server

64

Lấy các tham số gửi từ Client

- Một request có thể đính kèm số lượng tham số bất kỳ
- Các tham số được gửi từ các forms HTML
 - GET: dưới dạng 1 query string, đính kèm vào URL
 - POST: tham số được mã hóa, không xuất hiện trong URL
- `getParameter("paraName")`
 - Trả về giá trị của tham số `paraName`
 - Trả về null nếu không có tham số tên tương ứng được gọi
 - Làm việc như nhau với GET và POST requests

DatTT-DSE-SOICT-HUST

65

Ví dụ Form gửi theo phương thức 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>
```

DatTT-DSE-SOICT-HUST

66

Ví dụ Form gửi theo phương thức GET

DatTT-DSE-SOICT-HUST

67

Xử lý trong Servlet

```
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>" +
            "<H1 ALIGN=CENTER>" + title + "</H1>" +
            "<UL>" +
            "<LI><B>First Name in Response</B>: " +
            request.getParameter("param1") + "<br>" +
            "<LI><B>Last Name in Response</B>: " +
            request.getParameter("param2") + "<br>" +
            "<LI><B>Nick Name in Response</B>: " +
            request.getParameter("param3") + "<br>" +
            "</UL>" +
            "</BODY></HTML>");
    }
}
```

DatTT-DSE-SOICT-HUST

68

Ví dụ Form gửi theo phương thức 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>
```

DatTT-DSE-SOICT-HUST

69

Ví dụ Form gửi theo phương thức POST

DatTT-DSE-SOICT-HUST

70

Xử lý trong Servlet

```
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);
    }
}
```

DatTT-DSE-SOICT-HUST

71

Thiết lập thuộc tính tầm vực Request

- Các thuộc tính tầm vực Request có thể được thiết lập theo 2 cách
 - Servlet container có thể tự thiết lập 1 thuộc tính trong 1 request
 - Ví dụ: thuộc tính javax.servlet.request.X509Certificate attribute for HTTPS
 - Servlet cũng có thể thiết lập thuộc tính:
 - void setAttribute(java.lang.String name, java.lang.Object o)

DatTT-DSE-SOICT-HUST

72

Ví dụ:

```
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);
    }
}
```

DatTT-DSE-SOICT-HUST

73

Lấy thông tin client

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

DatTT-DSE-SOICT-HUST

74

Lấy thông tin Server

- Servlet có thể lấy các thông tin về server:
 - String request.getServerName()
 - Ví dụ: "www.sun.com"
 - int request.getServerPort()
 - Ví dụ: Port number "8080"

DatTT-DSE-SOICT-HUST

75

Lấy ra các thông tin khác

- Input stream
 - ServletInputStream getInputStream()
 - java.io.BufferedReader getReader()
- Protocol
 - java.lang.String getProtocol()
- Content type
 - java.lang.String getContentType()
- Là secure hay không (là HTTPS hay không)
 - boolean isSecure()

DatTT-DSE-SOICT-HUST

76

HttpServletRequest

DatTT-DSE-SOICT-HUST

77

HTTP Servlet Request là gì?

- Chứa dữ liệu truyền từ HTTP client tới HTTP servlet
- Được tạo bởi servlet container và được truyền cho servlet như 1 tham số của phương thức doGet() hoặc doPost()
- HttpServletRequest mở rộng interface ServletRequest và cung cấp thêm các phương thức cho phép truy cập
 - HTTP request URL
 - Context, servlet, path, query information
 - Các thông tin về HTTP Request header
 - Thông tin về Authentication type và User security
 - Cookies
 - Session

DatTT-DSE-SOICT-HUST

78

HTTP Request URL

- Chứa các phần như sau:
 - `http://[host]:[port]/[request path]?[query string]`

DatTT-DSE-SOICT-HUST

79

HTTP Request URL: [request path]

- `http://[host]:[port]/[request path]?[query string]`
- [request path] bao gồm
 - Context: /<context of web app>
 - Servlet name: /<component alias>
 - Path information: phần còn lại
- Ví dụ
 - `http://localhost:8080/hello1/greeting`
 - `http://localhost:8080/hello1/greeting.jsp`
 - `http://daydreamer/catalog/lawn/index.html`

DatTT-DSE-SOICT-HUST

80

HTTP Request URL: [query string]

- `http://[host]:[port]/[request path]?[query string]`
- [query string] bao gồm tập các tham số và giá trị người dùng nhập vào
- 2 cách sinh ra query strings
 - Một query string có thể xuất hiện ngay trong 1 trang web
 - `Add To Cart`
 - `String bookId = request.getParameter("Add");`
 - Một query string sẽ được gắn vào 1 URL khi submit 1 form qua phương thức **GET HTTP**
 - `http://localhost/hello1/greeting?username=Monica+Clinton`
 - `String userName=request.getParameter("username")`

DatTT-DSE-SOICT-HUST

81

Context, Path, Query, Parameter Methods

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

DatTT-DSE-SOICT-HUST

82

HTTP Request Headers

- HTTP requests chứa nhiều **request headers** cung cấp các thông tin phụ về request
- Ví dụ HTTP 1.1 Request:

```
GET /search? keywords= servlets+ jsp HTTP/ 1.1
Accept: image/ gif, image/ jpeg, */*
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)
```

DatTT-DSE-SOICT-HUST

83

HTTP Request Headers

- Accept
 - Chỉ ra những loại MIME trình duyệt có thể xử lý
- Accept-Encoding
 - Chỉ ra loại mã hóa (Ví dụ gzip hoặc compress) trình duyệt có thể xử lý
- Authorization
 - Nhận dạng người dùng cho các trang bảo mật
 - Thay vì HTTP authorization, sử dụng HTML forms để gửi username/password và lưu trữ thông tin trong session object

DatTT-DSE-SOICT-HUST

84

HTTP Request Headers

- Connection
 - Trong HTTP 1.1, mặc định là kết nối persistent (**persistent connection**)
 - Servlets nên thiết lập Content-Length bằng phương thức `setContentLength` (sử dụng `ByteArrayOutputStream` chỉ định độ dài của output) để hỗ trợ kết nối persistent.
- Cookie
 - cookies server gửi cho client trước đó.
- Host
 - Chỉ định host từ URL gốc
 - Được yêu cầu trong HTTP 1.1.

DatTT-DSE-SOICT-HUST

85

Các phương thức HTTP Header

- `String getHeader(java.lang.String name)`
 - Giá trị String của 1 **request header** cụ thể
- `java.util.Enumeration getHeaders(java.lang.String name)`
 - Giá trị Enum của 1 **request header**
- `java.util.Enumeration getHeaderNames()`
- `int getIntHeader(java.lang.String name)`

DatTT-DSE-SOICT-HUST

86

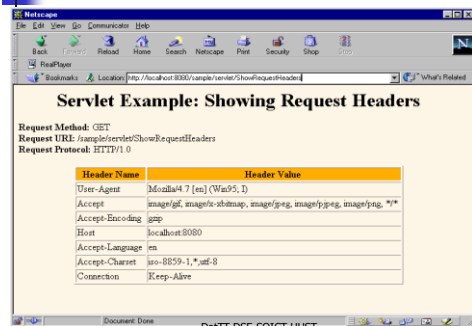
Code lấy ra các Request Headers

```
//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>\n" +
            ...
            "<TD>Header Name</TD>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));
        }
        ...
    }
}
```

DatTT-DSE-SOICT-HUST

87

Kết quả lấy ra các Request Headers



Header Name	Header Value
User-Agent	Mozilla/4.7 [en] (Win95; I)
Accept	image/gif, image/x-shmup, image/png, image/jpeg, image/svg, */*
Accept-Encoding	gzip
Host	localhost:8080
Accept-Language	en
Accept-Charset	iso-8859-1, *; q=0
Connection	Keep-Alive

DatTT-DSE-SOICT-HUST

88

Các phương thức Authentication và thông tin người dùng

- `String getRemoteUser()`
 - Định danh (name) của **client user** nếu servlet là **password protected**, null nếu ngược lại
- `String getAuthType()`
 - Loại kỹ thuật authentication sử dụng để bảo vệ servlet
- `boolean isUserInRole(java.lang.String role)`
 - User có "role" hay không?

DatTT-DSE-SOICT-HUST

89

Các phương thức Cookie (trong HttpServletRequest)

- `Cookie[] getCookies()`
 - Một mảng chứa tất cả các đối tượng Cookie client gửi trong request

DatTT-DSE-SOICT-HUST

90

6. Servlet Response (HttpServletResponse)

DatTT-DSE-SOICT-HUST

91

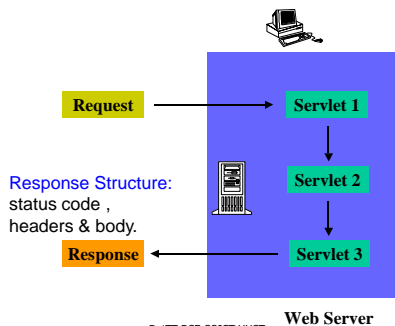
Servlet Response là gì?

- Chứa dữ liệu truyền từ servlet về client
- Tất cả các servlet responses thực thi giao diện ServletResponse
 - Lấy 1 **output stream**
 - Chỉ định **content type**
 - Có thiết lập buffer đầu ra không
 - Thiết lập **localization information**
- HttpServletResponse kế thừa giao diện ServletResponse
 - Mã trạng thái HTTP trả về (**HTTP response status code**)
 - Cookies

DatTT-DSE-SOICT-HUST

92

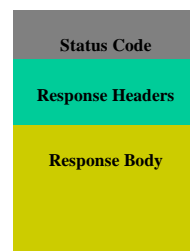
Responses



DatTT-DSE-SOICT-HUST

93

Cấu trúc Response



DatTT-DSE-SOICT-HUST

94

6.1. Mã trạng thái (Status Code) trong Http Response

DatTT-DSE-SOICT-HUST

95

HTTP Response Status Codes

- Tại sao cần **HTTP response status code**?
 - Giúp trình duyệt forward đến 1 trang khác
 - Chỉ ra được có resource bị thiếu
 - Hướng dẫn browser sử dụng bản sao được cache của dữ liệu

DatTT-DSE-SOICT-HUST

96

Phương thức thiết lập HTTP Response Status Codes

- `public void setStatus(int statusCode)`
 - Mã trạng thái được định nghĩa trong `HttpServletResponse`
 - Các mã trạng thái chia làm 5 nhóm:
 - 100-199 Informational
 - 200-299 Successful
 - 300-399 Redirection
 - 400-499 Incomplete
 - 500-599 Server Error
 - Mã trạng thái mặc định là 200 (OK)

DatTT-DSE-SOICT-HUST

97

Ví dụ HTTP Response Status

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

DatTT-DSE-SOICT-HUST

98

Các mã trạng thái phổ biến

- 200 (SC_OK)
 - Mã thành công, kèm nội dung gửi theo
 - Mặc định cho servlet
- 204 (SC_No_CONTENT)
 - Mã thành công nhưng không có nội dung gửi theo
 - Trình duyệt sẽ hiển thị nội dung nhận lần trước
- 301 (SC_MOVED_PERMANENTLY)
 - Tài liệu yêu cầu đã bị loại bỏ, Trình duyệt tự động request đến địa chỉ mới

DatTT-DSE-SOICT-HUST

99

Các mã trạng thái phổ biến

- 302 (SC_MOVED_TEMPORARILY)
 - Chú ý thông điệp ở đây là "Đã tìm thấy" (Found)
 - Tài liệu được yêu cầu tạm thời chuyển sang nơi khác (được chỉ ra trong Location header)
 - Browsers tự động chuyển request đến vị trí mới
 - Servlets nên sử dụng phương thức `sendRedirect`, thay vì `setStatus`, khi thiết lập header này
- 401 (SC_UNAUTHORIZED)
 - Trình duyệt cố gắng truy cập trang có yêu cầu password mà không có **Authorization header**
- 404 (SC_NOT_FOUND)
 - Không có trang yêu cầu

DatTT-DSE-SOICT-HUST

100

Các phương thức gửi lỗi (Error)

- Các mã trạng thái lỗi (400-599) có thể được sử dụng trong phương thức `sendError`.
- `public void sendError(int sc)`
- `public void sendError(int code, String message)`
 - Đóng gói thông điệp trong 1 HTML document nhỏ

DatTT-DSE-SOICT-HUST

101

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);
}

DatTT-DSE-SOICT-HUST
```

102

6.2. Header trong Http Response

DatTT-DSE-SOICT-HUST

103

Tại sao cần HTTP Response Headers?

- Forward đến địa chỉ mới nào
- Sửa cookies
- Cung cấp thông tin thời gian chỉnh sửa page.
- Hướng dẫn trình duyệt load lại trang sau 1 khoảng thời gian nhất định
- Đưa ra kích thước file được sử dụng trong HTTP connections loại persistent
- Chỉ định loại document sinh ra & trả về client
- ...

DatTT-DSE-SOICT-HUST

104

Các phương thức thiết lập Response Headers

- `public void setHeader(String headerName, String headerValue)`
 - Thiết lập 1 header bất kỳ
- `public void setDateHeader(String name, long millisecs)`
- `public void setIntHeader(String name, int headerValue)`
- `addHeader, addDateHeader, addIntHeader`
 - Thêm mới header

DatTT-DSE-SOICT-HUST

105

Các phương thức thiết lập các Response Headers phổ biến

- `setContentType`
 - Thiết lập **Content-Type** header. Servlets gần như luôn sử dụng phương thức này.
- `setContentLength`
 - Thiết lập **Content-Length** header. Được sử dụng cho HTTP connections loại persistent .
- `addCookie`
 - Thêm 1 giá trị trong **Set-Cookie** header.
- `sendRedirect`
 - Thiết lập **Location** header và thay đổi mã trạng thái

DatTT-DSE-SOICT-HUST

106

Một số loại MIME phổ biến

Type	Meaning
application/msword	Microsoft Word document
application/octet-stream	Unrecognized or binary data
application/pdf	Acrobat (.pdf) file
application/postscript	PostScript file
application/vnd.ms-excel	Excel spreadsheet
application/vnd.ms-powerpoint	Powerpoint presentation
application/x-gzip	Gzip archive
application/x-java-archive	JAR file
application/x-java-vm	Java bytecode (.class) file
application/zip	Zip archive
audio/basic	Sound file in .au or .snd format
audio/x-aiff	AIFF sound file
audio/x-wav	Microsoft Windows sound file
audio/midi	MIDI sound file
text/css	HTML cascading style sheet
text/html	HTML document
text/plain	Plain text
text/xml	XML document
image/gif	GIF image
image/jpeg	JPEG image
image/png	PNG image
image/tiff	TIFF image
video/mpeg	MPEG video clip
video/quicktime	QuickTime video clip

DatTT-DSE-SOICT-HUST

107

HTTP 1.1 Response Headers phổ biến

- Location**
 - Chỉ ra địa chỉ mới của 1 document
 - Nên sử dụng phương thức `sendRedirect`, thay vì thiết lập trực tiếp
- Refresh**
 - Chỉ ra khoảng thời gian định kỳ trình duyệt tự động load lại trang
- Set-Cookie**
 - Cookies mà trình duyệt phải lưu trữ.
 - Không thiết lập trực tiếp header này, mà sử dụng phương thức `addCookie`

DatTT-DSE-SOICT-HUST

108

HTTP 1.1 Response Headers phổ biến (2)

- Cache-Control (1.1) và Pragma (1.0)
 - Giá trị no-cache ngăn trình duyệt **caching page**.
Gửi cả 2 loại headers hoặc check phiên bản HTTP
- Content-Encoding
 - Cách thức mã hóa document.
 - Browser giải mã trước khi xử lý document
- Content-Length
 - Số byte trong response. Được sử dụng cho HTTP connections loại persistent .

DatTT-DSE-SOICT-HUST

109

HTTP 1.1 Response Headers phổ biến (3)

- Content-Type
 - Loại MIME của document được trả về.
 - Sử dụng phương thức **setContentTypes** để thiết lập header này.
- Last-Modified
 - Thời điểm thay đổi cuối cùng của document
 - Cung cấp phương thức **getLastModified** thay vì thiết lập trực tiếp header này,.

DatTT-DSE-SOICT-HUST

110

Ví dụ thiết lập Refresh header

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

DatTT-DSE-SOICT-HUST

111

6.3. Body trong Http Response

DatTT-DSE-SOICT-HUST

112

Tạo Response Body

- Một servlet gần như luôn trả về 1 response body
- Response body có thể là một **PrintWriter** hoặc một **ServletOutputStream**
- **PrintWriter**
 - Sử dụng phương thức **response.getWriter()**
 - Cho output loại ký tự (character-based)
- **ServletOutputStream**
 - Sử dụng phương thức **response.getOutputStream()**
 - Cho dữ liệu dạng binary (ví dụ: image)

DatTT-DSE-SOICT-HUST

113

7. Xử lý lỗi (Errors)

DatTT-DSE-SOICT-HUST

114

Xử lý lỗi

- Web container sinh ra trang hiển thị lỗi (error page) mặc định
 - LTV có thể thay bằng trang mới
- Các bước xử lý lỗi:
 - Tạo các trang html tương ứng với các loại lỗi khác nhau
 - Chỉnh sửa file web.xml

DatTT-DSE-SOICT-HUST

115

Ví dụ: Thiết lập các trang hiển thị lỗi trong 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>

```

DatTT-DSE-SOICT-HUST

116