# CHƯƠNG VIII
# LẬP TRÌNH WEB VỚI JAVA SERVLET - JSP

# Chương 8: Lập trình web với java

1. Tổng quan nền tảng Java EE 7
2. Web server - Servlet
3. Ngôn ngữ JSP

# Tổng quan Java EE 7

- Cung cấp nền tảng chung cho các loại thành phần khác
- Cải tiến năng suất:
  - Với nhiều chú thích
  - Ít cấu hình XML
  - Đóng gói đơn giản

# Tổng quan Java EE 7(tt)

- Các tính năng mới:
  - Công nghệ mới:

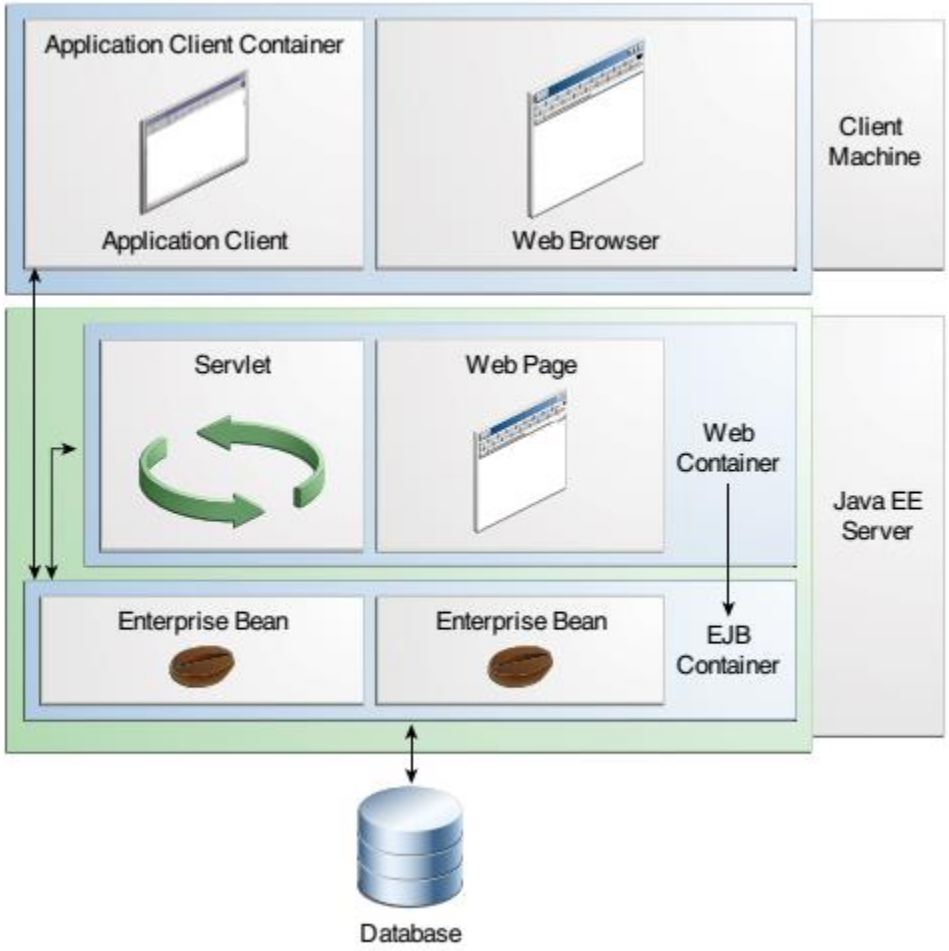
    + Ứng dụng hàng loạt cho nền tảng java

    + API java để xử lý JSON

    + API java cho WebSocket
  - Thành phần EJB (Enterprise JavaBeans Technology)
  - Các tính năng mới cho các servlet
  - Các tính năng mới cho JavaServlet Faces

# Tổng quan Java EE 7(tt)

- Các thành phần Java EE 7

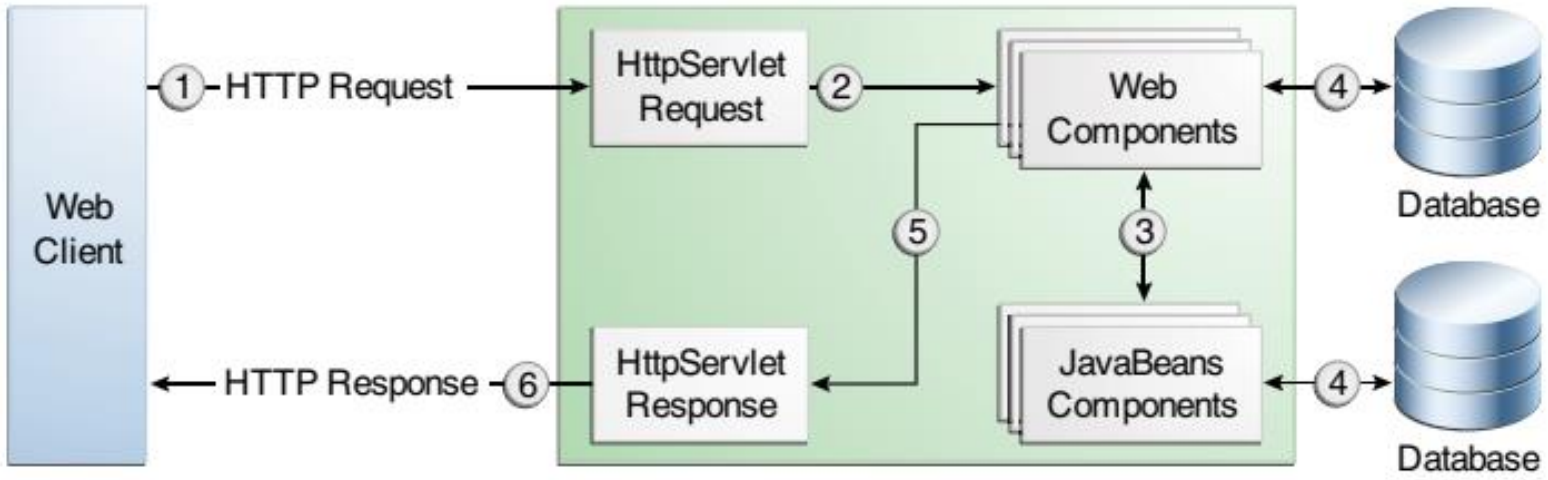


# Tổng quan Java EE 7(tt)

- Java EE server: Phần thời gian chạy của sản phẩm Java EE. Máy chủ Java EE cung cấp EJB và vùng chứa web.
- EJB container: Quản lý việc thực thi các bean doanh nghiệp cho các ứng dụng Java EE.
- Web contianer: Quản lý việc thực thi các trang web, các servlet và một số thành phần EJB cho các ứng dụng Java EE.
- Application client container: Quản lý việc thực thi các thành phần ứng dụng khách. Ứng dụng khách và vùng chứa của chúng chạy trên máy khách.
- Applet container: Quản lý việc thực thi các applet. Bao gồm một trình duyệt web và một Java plugin chạy trên máy khách cùng nhau.

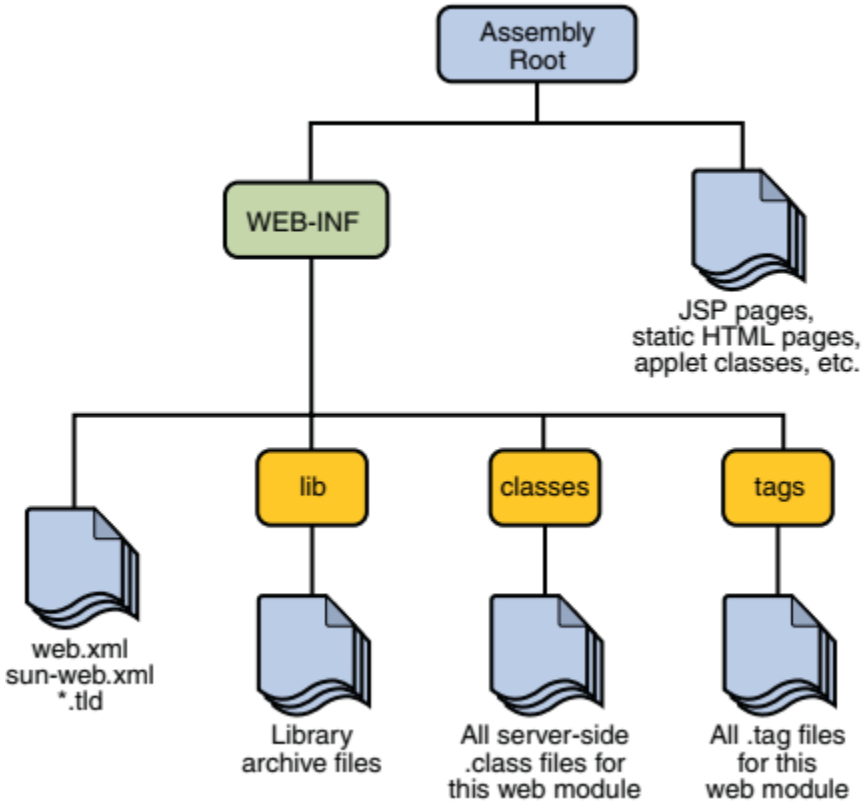
# Tổng quan Java EE 7(tt)

## Mô hình ứng dụng Web với Servlet



# Tổng quan Java EE 7(tt)

## Cấu trúc của Web Servlet



# Cơ bản về Servlet

1. Mô hình lập trình MVC với Servlet/JSP
2. Mô hình Servlet request và response
3. Vòng đời của Servlet
4. Servlet scope object
5. Servlet request
6. Servlet response: status, header, body

## Servlet là gì ?

- Sử dụng kỹ thuật thread
- 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 các web servers, mọi môi trường chỉ cần có máy ảo java.

## Ưu điểm của Servlet

- Sử dụng kỹ thuật thread
- 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 các web servers, mọi môi trường chỉ cần có máy ảo java.

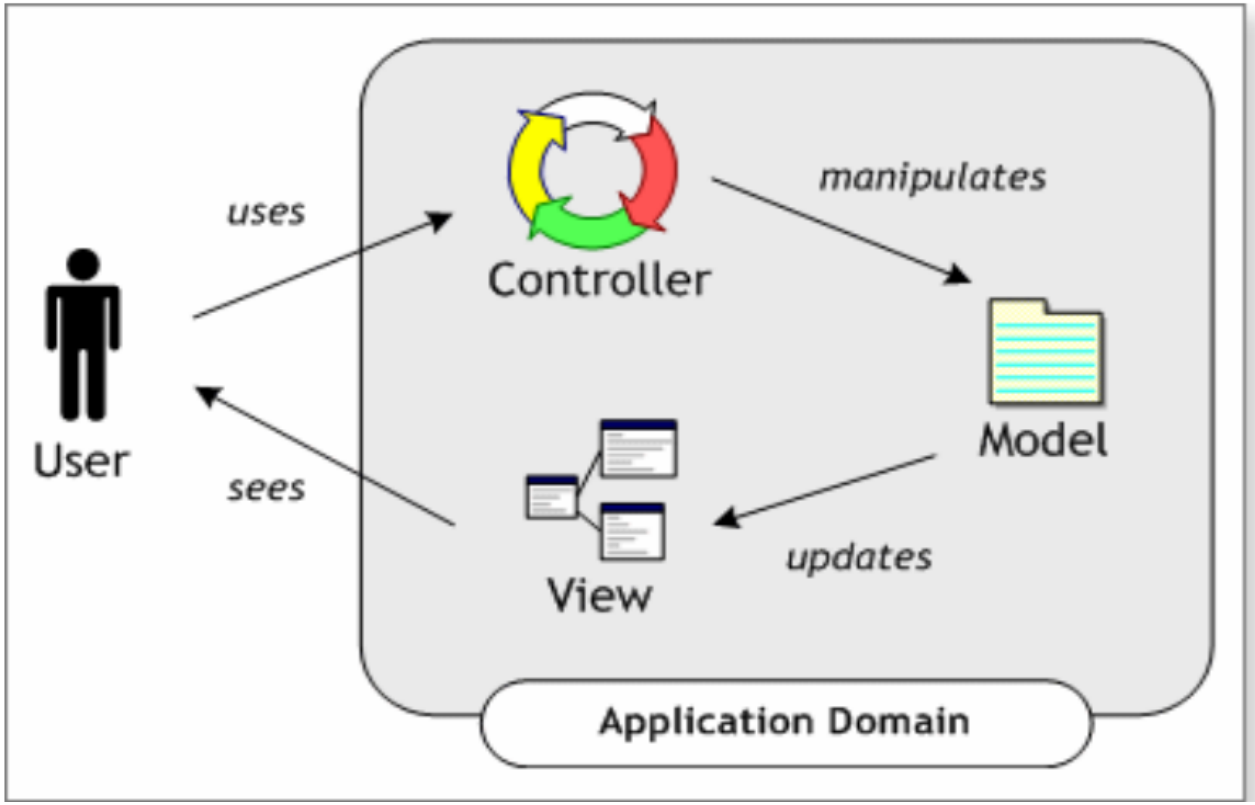
## Ưu điểm của JSP

- Tách biệt nội dung và cách thức trình bày
- Đơn giản hóa việc phát triển với JavaBeans
- Hỗ trợ tái sử dụng nhờ các components
- Độc lập Platform.

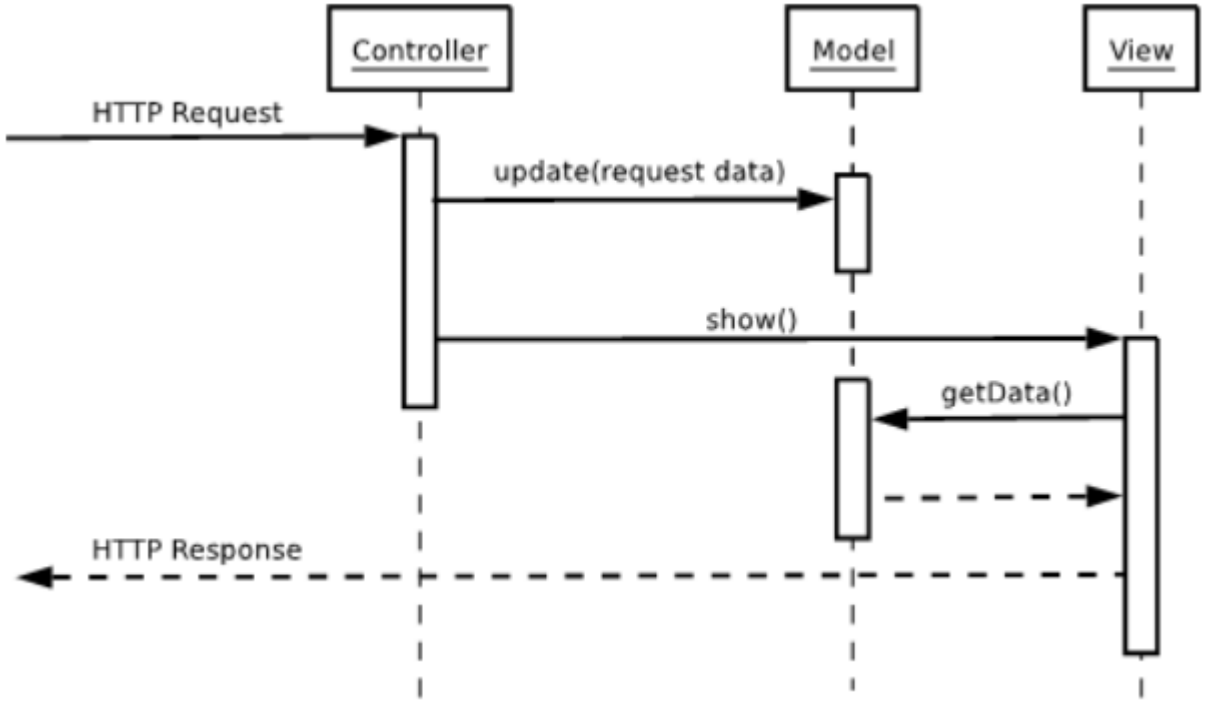
## Mô hình lập trình MVC với Servlet/JSP

- Kết hợp lập trình Servlet và JSP là thế mạnh của lập trình web với java.
- Servlet xử lý Controller, tiếp nhận và xử lí yêu cầu
- JSP xử lý View, sinh giao diện để hiển thị cho người dung.
- JavaBean, EL (Expression Language), và JSTL (Java Standard Tag Library) làm tăng thêm sức mạnh cho JSP.

# Mô hình lập trình MVC với Servlet/JSP(tt)



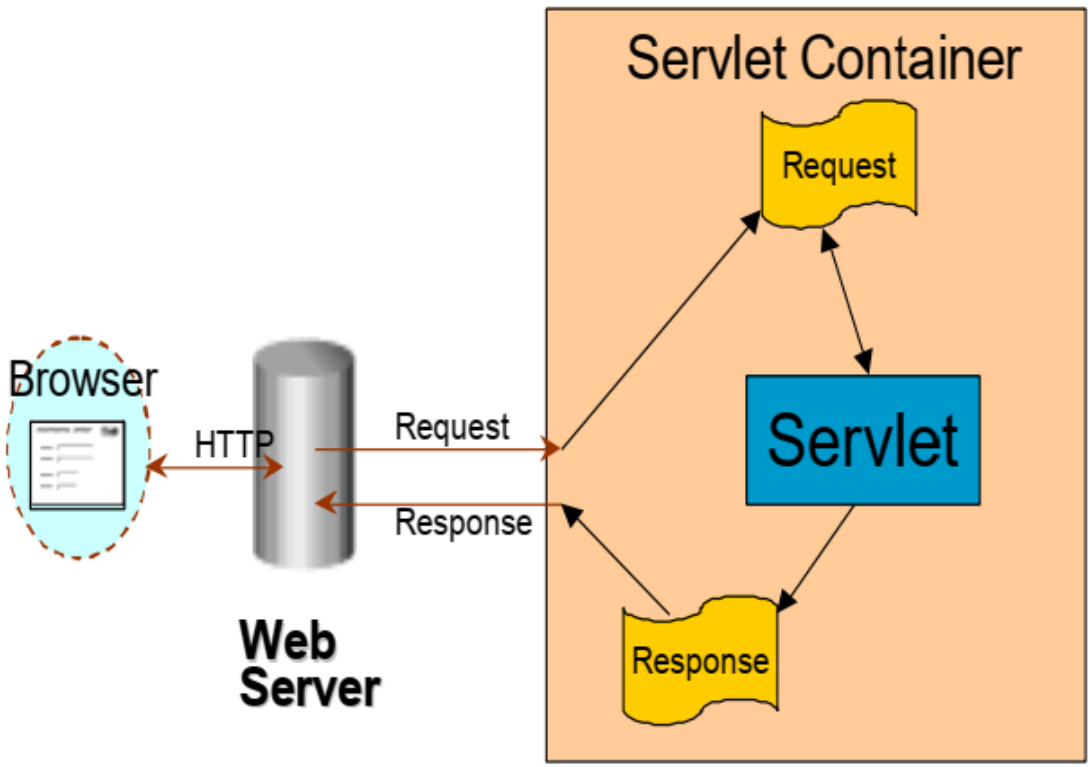
# Mô hình lập trình MVC với Servlet/JSP(tt)



## Mô hình lập trình MVC với Servlet/JSP(tt)

- Yêu cầu được gửi đến Controller (Servlet). Controller tiếp nhận, xử lý và làm việc với dữ liệu thông qua Model.
- View lấy dữ liệu từ Model (đã tạo từ Controller) để sản sinh giao diện.
- Model đóng vai trò cung cấp, cập nhật và chia sẻ giữa Controller và View.

## Mô hình Servlet Resquest và Response



## Mô hình Servlet Resquest và Response(tt)

- Request:
  - Client gửi thông tin đến server
    - Dữ liệu được user gửi đi
    - HTTP headers
- Respone:
  - Server gửi thông tin đến client
    - Dữ liệu text(html,..), image,…
    - HTTP headers, cookies,…

## Mô hình Servlet Resquest và Response(tt)

- Lớp cơ sở Servlet

```
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

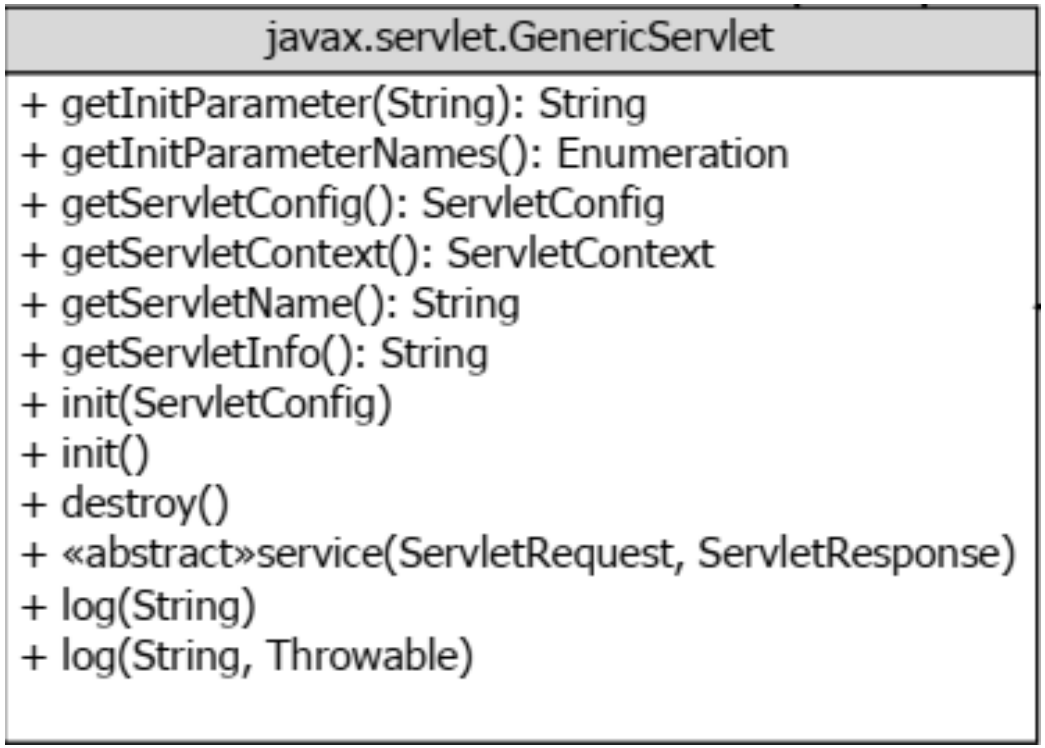
## Mô hình Servlet Resquest và Response(tt)

- Lớp cơ sở Servlet - HttpServlet

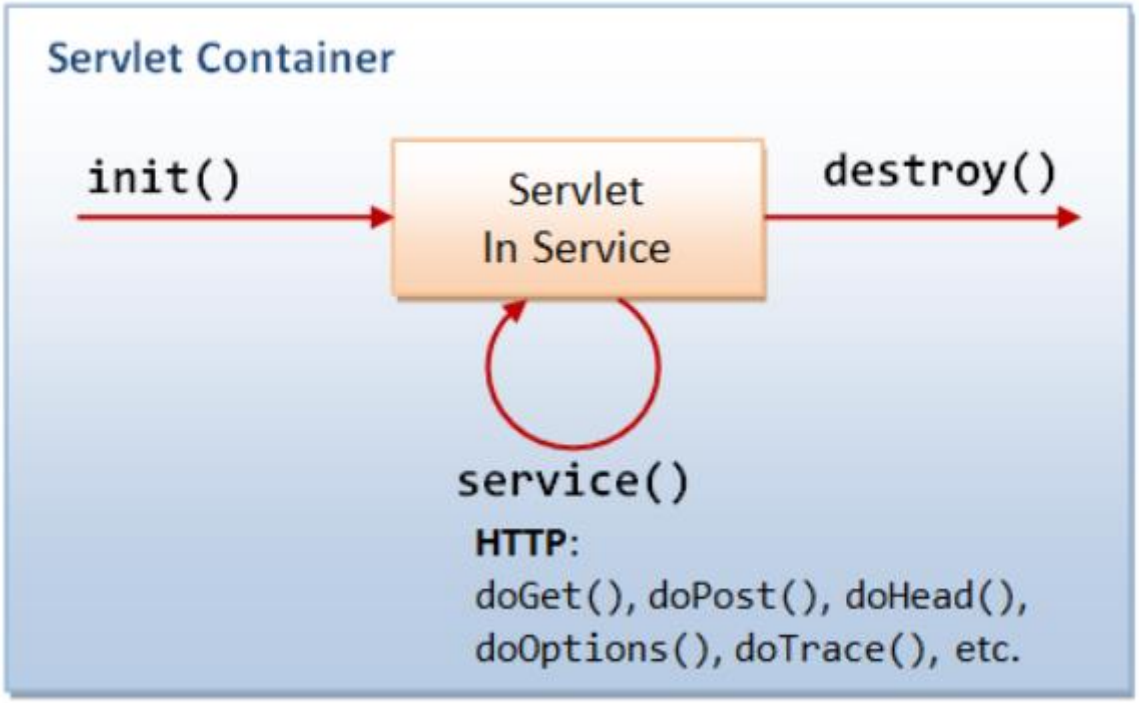
| javax.servlet.http.HttpServlet |
| --- |
| # doGet(HttpServletRequest, HttpServletResponse)<br># doPost(HttpServletRequest, HttpServletResponse)<br># doHead(HttpServletRequest, HttpServletResponse)<br># doOptions(HttpServletRequest, HttpServletResponse)<br># doTrace(HttpServletRequest, HttpServletResponse)<br># doPut(HttpServletRequest, HttpServletResponse)<br># doDelete(HttpServletRequest, HttpServletResponse)<br># getLastModified(HttpServletRequest): long<br>+ service(ServletRequest, ServletResponse)<br># service(HttpServletRequest, HttpServletResponse) |

## Mô hình Servlet Resquest và Response(tt)

- Lớp cơ sở Servlet - GenericServlet



## Vòng đời của Servlet



# Vòng đời của Servlet(tt)

## Các phương thức trong vòng đời Servlet

- Được gọi bởi container, điều khiển vòng đời của 1 servlet
- Định nghĩa trong:
  - Lớp javax.servlet.GenericServlet
    - init()
    - destroy()
    - service()

# Vòng đời của Servlet(tt)

## Các phương thức trong vòng đời Servlet (tt)

- 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

# Vòng đời của Servlet(tt)

## Các phương thức trong vòng đời Servlet (tt)

- 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

# Vòng đời của Servlet(tt)

## Các phương thức trong vòng đời Servlet (tt)

- 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)`

# Vòng đời của Servlet(tt)

## Phương thức doGet() và 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

# Scope Object

- 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

# Scope Object

## 4 loại scope: 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

# Scope Object

4 loại scope,các lớp 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

# Scope Object – Web context (ServletContext)

- 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

# Scope Object – Web context (ServletContext)

## Truy cập đối tượng servletcontext

- 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)
```

# Scope Object – Session(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
- Lấy đối tượng HttpSession:
  Sử dụng phương thức getSession() của 1 đối tượng Resquest (HttpServletRequest)

# Servlet request (HttpServletRequest)

- 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

# Servlet request (HttpServletRequest)

## Lấy 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

# Servlet request (HttpServletRequest)

## Lấy thông tin từ 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

# Servlet request (HttpServletRequest)

## Lấy thông tin từ 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"

# Servlet request (HttpServletRequest)

## Lấy thông tin khác

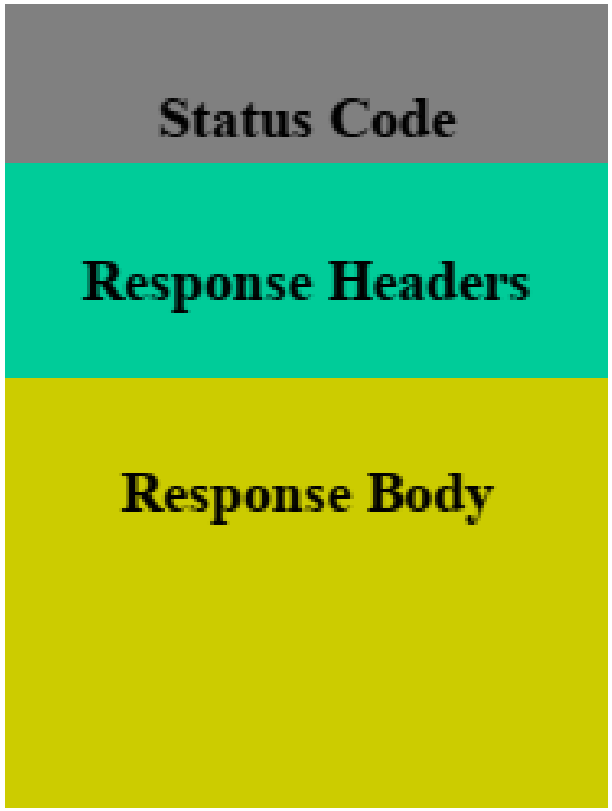
- Input stream
  - ServletInputStream getInputStream()
  - java.io.BufferedReader getReader()
- Protocol
  - java.lang.String getProtocol()
- Content type
  - java.lang.String getContentType()
- Là secure hay không (là HTTPS hay không)
  - boolean isSecure()

# Servlet Respone (HttpServletRespone)

- Chứa dữ liệu truyền từ servlet về client
- Tất cả các 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

# Servlet Respone (HttpServletRespone)

## Cấu trúc Respone



# Servlet Respone (HttpServletRespone)

## Mã trạng thái

- 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

# Servlet Respone (HttpServletRespone)

## Phương thức thiết lập status code

- 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)

# Servlet Respone (HttpServletRespone)

## Respone 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
- ...

# Servlet Respone (HttpServletRespone)

## Các phương thức thiết lập, phổ biến

- 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

# Servlet Respone (HttpServletRespone)

## Các phương thức thiết lập, phổ biến(tt)

- 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

# Servlet Respone (HttpServletRespone)

## Respone 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)