BÀI 3:
Phát triển ứng dụng Java Web với Servlet

Mục tiêu bài học

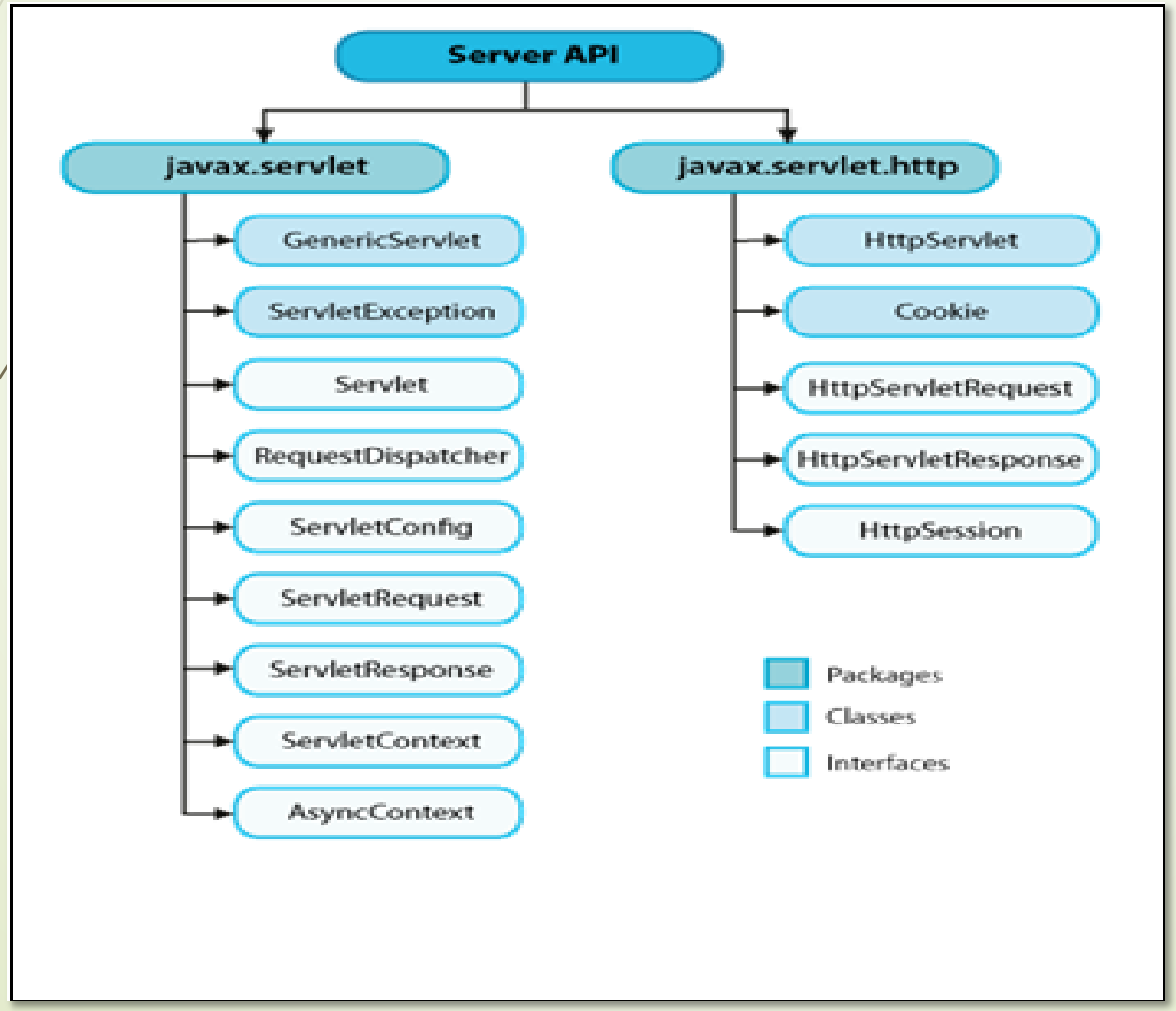
- Giới thiệu về Servlet: Kiến trúc tổng quan, vòng đời
- Cơ chế xử lý request của Servlet
- Giới thiệu về Servlet API
- Các bước phát triển ứng dụng web với Servlet

Servlet API

- Servlet API:
  - Được dùng để tạo và quản lý các servlets.
  - Bao gồm nhiều các interfaces và các classes cho phép tạo ra các servlet.
  - Các classes và interfaces thuộc Servlet API đều nằm trong các packages sau:
    - `javax.servlet`
    - `javax.servlet.http`

Servlet API

- Hình bên dưới minh họa về hệ thống phân cấp các lớp của Servlet API.



Servlet API

- Gói `javax.servlet`:
  - Bao gồm các interfaces được mô tả như bảng bên dưới.

Interface	***Mô tả***
`Servlet`	*Là interface nằm trên đầu của hệ thống phân cấp, cung cấp các methods mà tất cả các servlets đều phải cài đặt.*
`ServletConfig`	*Cung cấp các thông tin cấu hình của servlet trong quá trình khởi tạo và sử dụng servlet.*
`ServletContext`	*Cung cấp các phương thức để cho phép servlets có thể giao tiếp với môi trường bên ngoài (Web container).*
`ServletRequest`	*Cung cấp các thông tin mà người dùng đã gửi trong quá trình request lên servlet.*
`ServletResponse`	*Cung cấp các thông tin để hỗ trợ servlet trong quá trình gửi lại response cho client.*
`RequestDispatcher`	*Cho phép giao tiếp giữa các thành phần của một ứng dụng web. Cho phép chèn nội dung của một servlet khác, hoặc chuyển tiếp giữa các servlet.*
`AsyncContext`	*Cung cấp các methods để cho phép xử lý nhiều client requests một cách bất đồng bộ (asynchronously).*

Servlet API

- Servlet interface:
  - Là interface nằm trên đầu của hệ thống phân cấp, cung cấp các methods mà tất cả các servlets đều phải cài đặt.
  - Các methods sẽ được cài đặt bởi servlet class kế thừa từ lớp GenericServlet hoặc HttpServlet.
- Một số methods thông dụng của Servlet class:
  - void init(ServletConfig config) throws ServletException
  - ServletConfig getServletConfig()
  - String getServletInfo()
  - void service(ServletRequest request, ServletResponse response) throws ServletException, IOException
  - void destroy()

Servlet API

- `ServletConfig` interface:
  - Được cài đặt bởi Web container trong quá trình khởi tạo của servlet để truyền các thông tin cấu hình cho servlet.
  - Được truyền cho phương thức `init()` của servlet trong suốt quá trình khởi tạo.
  - Có thể được sử dụng để truyền các tham số khởi tạo (initialization parameters) cho servlets.
- Các tham số khởi tạo sẽ được truyền dưới dạng các cặp name-value.
  - VD: Có thể truyền connection URL như một tham số khởi tạo cho servlet.

Servlet API

- Một số methods thông dụng của `ServletConfig` interface:
  - `String getInitParameter(String param)`: Trả về một đối tượng `String` chứa giá trị của một tham số khởi tạo theo tên.
  - `Enumeration<String> getInitParameterNames()`: Trả về tên của tất cả các tham số khởi tạo dưới một đối tượng enumeration chứa các chuỗi là tên của các tham số khởi tạo.
  - `ServletContext getServletContext()`: Trả về đối tượng `ServletContext` được kết hợp với servlet đang thực thi.
  - `String getServletName()`: Trả về tên của servlet.

Servlet API

- `ServletContext` interface
  - Cung cấp thông tin về môi trường đang thực thi servlet.
- Mỗi ứng dụng Web chỉ chứa duy nhất một đối tượng `ServletContex`.
- Một đối tượng `ServletContext` cũng được biết như là một Web context.
- Đối tượng `ServletContext` cung cấp các methods có thể được dùng để giao tiếp với Web container.

Servlet API

- Một số methods phổ biến của `ServletContext` interface:
  - `public void setAttribute(String, Object)`: Thiết lập một thuộc tính và lưu dưới dạng một cặp name-value của đối tượng `ServletContext`.
  - `public Object getAttribute(String attrname)`: Trả về giá trị của attribute theo tên.
  - `public String getInitParameter(String pname)`: Trả về giá trị của tham số khởi tạo theo tên.

Servlet API

- `ServletRequest` interface cho phép một servlet có thể xử lý client requests bằng cách lấy ra những thông tin mà máy khách đã gửi lên cho servlet.
- Một số phương thức phổ biến của `ServletRequest` interface:
  - `public String getParameter(String paramName)`
  - `public String[] getParameterValues(String paramName)`
  - `public Enumeration getParameterNames()`
  - `public String getRemoteHost()`
  - `public String getRemoteAddr()`

Servlet API

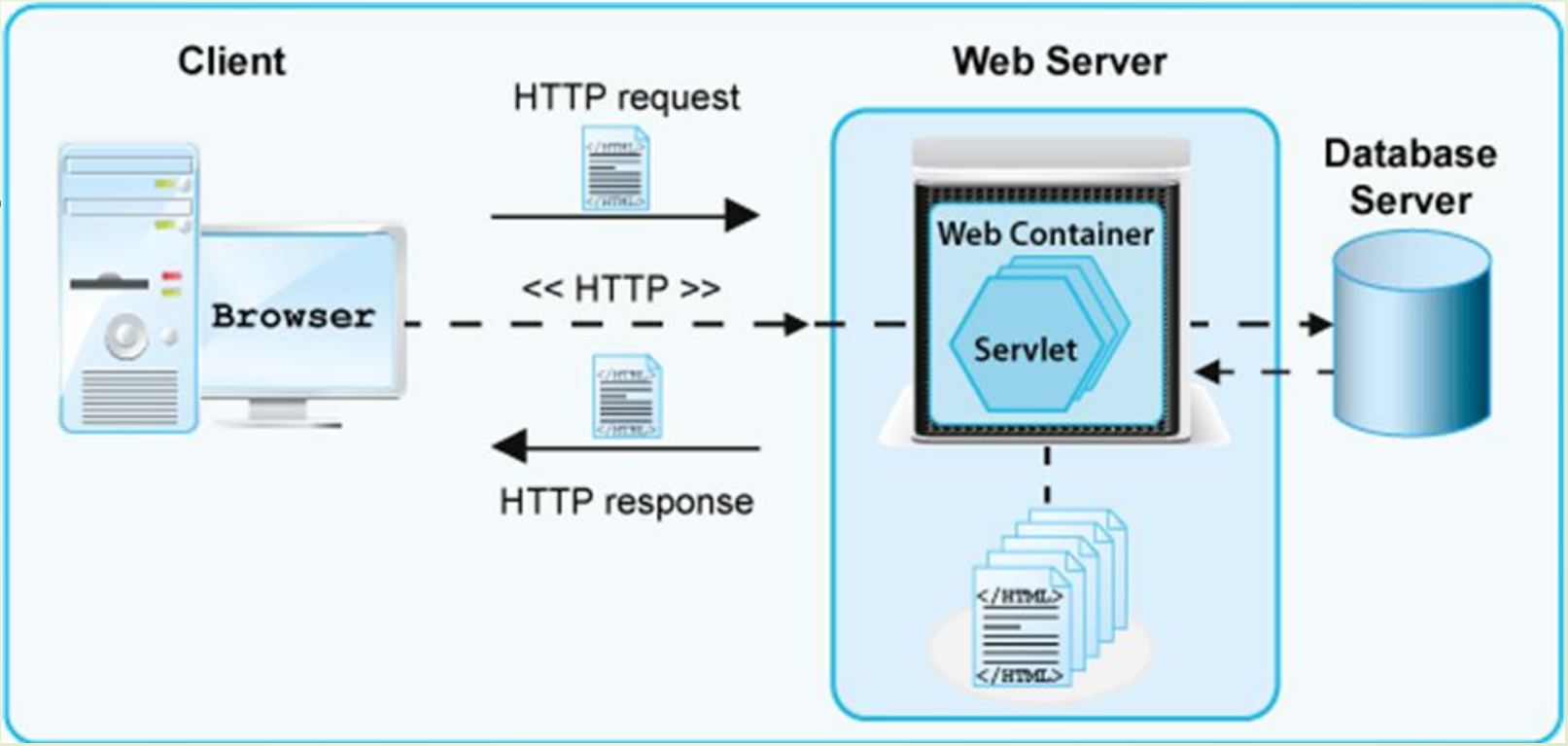
- `ServletResponse` interface cung cấp các phương thức cho phép một servlet phản hồi lại các client requests.
- Một số phương thức thông dụng của `ServletResponse` interface:
  - `public ServletOutputStream getOutputStream() throws IOException`
  - `public PrintWriter getWriter() throws IOException`
  - `public void setContentType(String type)`

Servlet API

- Gói `javax.servlet.http`:
  - Cung cấp các classes và interfaces cho phép tạo ra các servlet có thể giao tiếp bằng giao thức HTTP.
  - Gói này cung cấp lớp `HttpServlet` để cho phép tạo ra các servlets. Đây là một lớp abstract mà các servlet được sử dụng với giao thức HTTP đều cần phải kế thừa.
- Một số interfaces phổ biến của gói `javax.servlet.http`:
  - `HttpServletRequest`
  - `HttpServletResponse`
  - `HttpSession`

Giới thiệu về Web Container

- Web container:
  - Là một thành phần của Web server cung cấp môi trường run-time cho việc thực thi các servlets.
- Hình bên dưới minh họa về kiến trúc của Web container.



Giới thiệu về Web Container

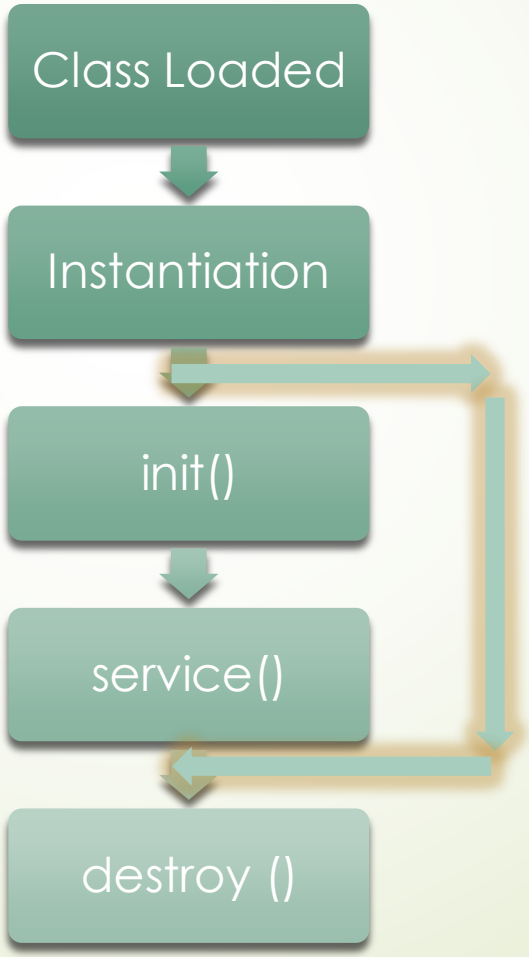
- Web container cung cấp các dịch vụ sau cho ứng dụng Web:
  - Hỗ trợ giao tiếp (Communication support)
  - Quản lý vòng đời (Lifecycle management)
  - Hỗ trợ đa luồng (Multithreading support)
  - Cung cấp cơ chế bảo mật (Declarative security)
  - Hỗ trợ JSP

Vòng đời của Servlet

- Vòng đời của servlet được quản lý bởi Web container, bằng cách sử dụng các phương thức sau của `Servlet` interface:
  - `init()`
  - `service()`
  - `destroy()`

Vòng đời của Servlet

- Hình bên dưới đây minh họa về thứ tự thực thi các phương thức trong vòng đời của Servlet.



Quá trình xử lý của Servlet Request

- Dưới đây là các bước trong quá trình servlet xử lý request của client:
  1. Người dùng gửi một request lên server. Server sau đó sẽ chuyển tiếp (forwards) request cho Web container để xử lý.
  2. Web container tạo ra các đối tượng `HttpServletRequest` và `HttpServletResponse` để phục vụ cho việc xử lý request và sinh ra response.
  3. Web container sau đó sẽ khởi tạo servlet thread bằng cách gọi phương thức `init()` và truyền các đối tượng `HttpServletRequest` và `HttpServletResponse` cho servlet.
  4. Phương thức `service()` của servlet sẽ được gọi để xử lý request của client.
  5. Tùy thuộc kiểu request, phương thức `service()` sẽ gọi phương thức `doGet()` hoặc `doPost()`.

Quá trình xử lý của Servlet Request

6. Phương thức `doGet()` hoặc `doPost()` sẽ sinh ra response cho máy khách và gắn kết quả với đối tượng response.
7. Web container gửi kết quả được sinh ra cho máy khách.
8. Web container xóa servlet instance cùng với các đối tượng request và response.

Tạo và sử dụng Servlets

- Để tạo và sử dụng các servlet, ta cần thực hiện các tác vụ sau:
  1. Tạo servlet.
  2. Cấu hình servlet vừa tạo.
  3. Biên dịch và đóng gói servlet.
  4. Triển khai servlet.
  5. Truy cập và gọi đến servlet bằng cách sử dụng một trình duyệt web.

Tạo Servlet

- Servlet có thể được tạo bằng cách định nghĩa một class kế thừa từ lớp `HttpServlet` hoặc `GenericServlet`.
- Ví dụ:

Cấu hình Servlet

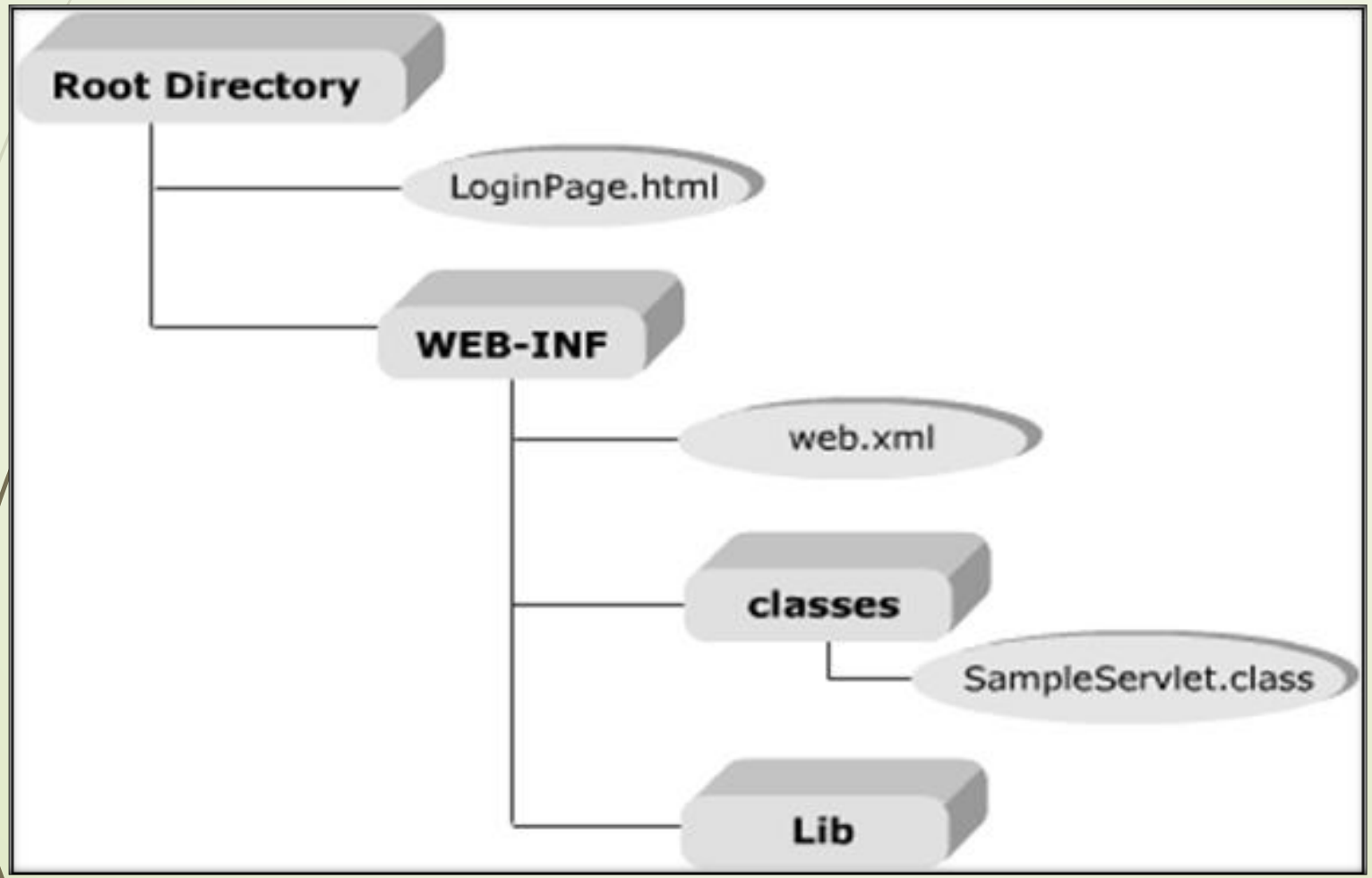
- Quá trình cấu hình có nghĩa là ta kết hợp một URL với một servlet.
- Servlet có thể được cấu hình bằng một trong 2 phương pháp sau:
  - Sử dụng file cấu hình (Deployment Descriptor): Đây là file XML có tên là web.xml, file này sẽ chứa thông tin cấu hình của tất cả các thành phần của ứng dụng web.
  - annotations: Là các metadata có thể được chèn vào mã nguồn. Annotation là các thành phần khai báo được dùng để cung cấp thông tin mô tả, để giải thích về ý nghĩa của các thành phần lập trình, VD như các classes, fields, và các methods trong chương trình.

Cấu hình Servlet

- Dưới đây là một số thành phần phổ biến của file web.xml cùng với ý nghĩa của chúng:
  - `<servlet>`: Thiết lập chi tiết các thông tin cấu hình của các servlets được sử dụng trong ứng dụng Web, và bao gồm các thẻ con sau:
    - `<servlet-name>: Cấu hình tên servlet`
    - `<servlet-class>: Khai báo tên class định nghĩa servlet`
  - `<servlet-mapping>`: Cho phép cấu hình mapping giữa một url và một tên servlet. Gồm có các thành phần con sau:
    - `<servlet-name>: Khai báo tên servlet`
    - `<url-pattern>: Cấu hình đường dẫn để gọi servlet`
  - `<session-config>: Cấu hình session cho ứng dụng`
  - `<init-param>: Cấu hình tham số khởi tạo cho servlet`

Biên dịch và đóng gói Servlet

- Quá trình biên dịch servlet sẽ tạo ra một file class.
- Quá trình packaging sẽ cho phép đóng gói ứng dụng thành một cấu trúc tuân theo chuẩn Java EE như hình bên dưới.



Tổng kết

- Giới thiệu về Servlet: Kiến trúc tổng quan, vòng đời
- Cơ chế xử lý request của Servlet
- Giới thiệu về Servlet API
- Các bước phát triển ứng dụng web với Servlet