1. **Tổng quan về Servlet**

Java Servlets là các chương trình chạy trên một Web server hoặc một Application server và thực hiện như là một tầng trung gian giữa một Yêu cầu từ một trình duyệt web hoặc HTTP client với các Database hoặc các ứng dụng trên HTTP server.

Sử dụng Servlets, chúng ta có thể thu thập Input từ người sử dụng thông qua các form trên trang web, hiển thị các bản ghi (record) từ một Database hoặc từ nguồn khác, và tạo các trang web động.

Java Servlets thường có chung mục đích: là các chương trình độc lập bởi việc sử dụng Common Gateway Interface (CGI). Nhưng so với CGI, thì Servlets có các lợi thế sau:

* Hiệu năng tốt hơn đáng kể.

Servlets thực thi bên trong không gian địa chỉ của một Web server, không cần thiết phải tạo một tiến trình riêng biệt để xử lý mỗi yêu cầu từ Client.

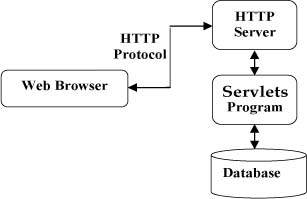
* Servlets là độc lập trên nền tảng bởi vì chúng được viết bằng Java.

Bởi vì viết bằng Java (mà Java cung cấp tình bảo mật cao trên Server) nên Servlets là đáng tin cậy.

Tính năng đầy đủ của thư viện của các lớp trong Java là luôn luôn có sẵn cho Servlets. Nó có thể giao tiếp với Applet, Database hoặc phần mềm khác thông qua các Socket và kỹ thuật RMI

1. **Cấu trúc của Servlets**

Sơ đồ dưới đây minh họa cấu trúc của Servlets trong một ứng dụng web:



* Nhiệm vụ của Servlets

Servlets thực hiện các tác vụ chủ yếu sau:

* Đọc dữ liệu hiển thị (explicit) được gửi bởi Client (hoặc trình duyệt) bao gồm một HTML Form trên một trang web hoặc nó cũng có thể từ một Applet hoặc một chương trình Custom từ HTTP Client.
* Đọc dữ liệu yêu cầu HTTP ẩn (implicit) được gửi bởi Client (hoặc trình duyệt) bao gồm cookie, các loại media.
* Xử lý dữ liệu và cho ra kết quả. Tiến trình này có thể yêu cầu Database, đang thực thi một triệu hồi tới RMI hoặc CORBA, triệu hồi một Web Service, hoặc tính toán phản hồi một cách trực tiếp.
* Gửi dữ liệu hiển thị (ví dụ: tài liệu) tới các Client (hoặc trình duyệt). Tài liệu này có thể được gửi theo nhiều định dạng khác nhau, gồm text (HTML hoặc XML), nhị phân (hình ảnh GIF), Excel, .v.v.
* Gửi phản hồi HTTP ẩn tới các Client (hoặc trình duyệt), thông báo cho trình duyệt hoặc Client về kiểu của tài liệu được trả về (ví dụ: HTML), thiết lập các Cookie và Caching các tham số, cùng các tác vụ khác.
* Package trong Servlets
* Java Servlets là các lớp trong Java chạy bởi một Web Server mà có một trình thông dịch hỗ trợ Java Servlets.
* Servlets có thể được tạo bởi sử dụng các gói javax.servlet và javax.servlet.http là một phiên bản mở rộng của thư viện lớp Java để hỗ trợ các dự án phát triển có phạm vi lớn. Các lớp này triển khai Java Servlet và JSP.
* Java Servlet đã được tạo và được biên dịch giống như các lớp khác trong Java. Sau khi cài đặt các gói servlet và thêm chúng vào Classpath trong máy, chúng ta có thể biên dịch Servlet với bộ biên dịch JDK hoặc bất kỳ bộ biên dịch nào khác.

1. **Vòng đời của Servlet**

Vòng đời của Servlet có thể được định nghĩa như là một tiến trình đầy đủ từ khi được tạo ra đến khi bị hủy. Một Servlet theo các giai đoạn sau:

* Servlet được khởi tạo bởi gọi phương thức init().
* Servlet gọi phương thức service() để xử lý một yêu cầu từ Client.
* Servlet bị hủy bởi triệu hồi phương thức destroy().

Cuối cùng, servlet trở thành rác và được thu thập bởi Garbage Collector của JVM.

Tiếp theo chúng ta đi vào chi tiết từng phương thức trong vòng đời của Servlet:

* Phương thức init()

Phương thức được thiết kế để chỉ được gọi một lần. Nó được gọi khi Servlet lần đầu được tạo, và không được gọi lại cho mỗi yêu cầu của người dùng. Vì thế, nó được sử dụng cho các khởi tạo one-time, giống như phương thức init của Applet.

Thường thì, Servlet được tạo khi một người dùng lần đầu triệu hồi một URL tương ứng với Servlet đó, chúng ta cũng có thể xác định rằng Servlet này được tải khi Server được khởi động lần đầu.Khi một người sử dụng triệu hồi một Servlet, một đối tượng đơn của Servlet được tạo, với mỗi yêu cầu từ người dùng, là kết quả trong một thread mới, mà được thao tác tới doGet hoặc doPost một cách thích hợp. Phương thức init() tạo hoặc tải một vài dữ liệu mà sẽ được sử dụng qua vòng đời của Servlet đó.

* Định nghĩa phương thức init() như sau:

public void init() throws ServletException {

// Initialization code...

}

* Phương thức service()

Phương thức service() là phương thức chính để thực hiện tác vụ thực sự. Nơi chứa servlet (ví dụ như Web Server) gọi phương thức service() để xử lý các yêu cầu từ Client (hoặc trình duyệt) và viết phản hồi đã được định dạng trở lại Client đó.

Mỗi khi Server nhận một yêu cầu cho một Servlet, thì Server tạo một Thread mới và triệu hồi service(). Phương thức service() kiểm tra kiểu yêu cầu HTTP (Kiểu GET, POST, PUT, DELET, .v.v.) và gọi các phương thức doGet, doPost, doPut, doDelete .v.v. tương ứng một cách thích hợp.

Định nghĩa phương thức service():

public void service(ServletRequest request,

ServletResponse response)

throws ServletException, IOException{

}

Phương thức service() được gọi bởi Container và nó triệu hồi các phương thức doGet, doPost, doPut, doDelet, .v.v. tương ứng. Vì thế, chúng ta không phải làm gì cả với service() nhưng việc ghi đè phương thức hoặc doGet() hoặc doPost tùy thuộc vào kiểu yêu cầu mà chúng ta nhận từ Client.

Hai phương thức doGet() và doPost() được sử dụng thường xuyên nhất với mỗi service. Sau đây là chi tiết về hai phương thức này:

* Phương thức doGet()

Một yêu cầu GET, là kết quả từ một yêu cầu chuẩn cho URL hoặc từ một HTML form, mà không có PHƯƠNG THỨC nào được xác định và nó nên được xử lý bởi phương thức doGet().

public void doGet(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

// Servlet code

}

* Phương thức doPost()

Một yêu cầu POST, là kết quả từ một HTML form mà liệt kê POST như là PHƯƠNG THỨC, và nên được xử lý bởi phương thức doPost():

public void doPost(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

// Servlet code

}

* Phương thức destroy()

Phương thức destroy() chỉ được gọi một lần ở giai đoạn cuối trong vòng đời Servlet. Phương thức này giúp servlet của chúng ta một cơ hội để đóng các kết nối tới Database, dừng thread, viết các danh sách cookie hoặc viết tính toán trên đĩa, và thực hiện các hoạt động cleanup khác.

Sau khi phương thức destroy() được gọi, đối tượng servlet này được đánh dấu cho Garbage Collector. Phương thức này trông giống như sau:

public void destroy() {

// Finalization code...

}

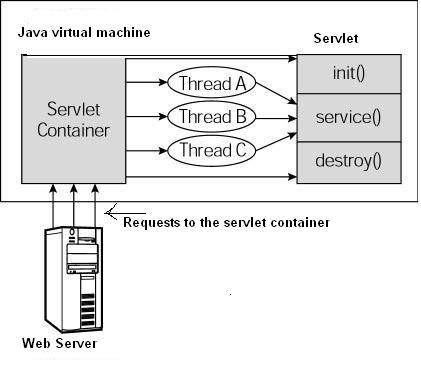
* Sơ đồ cấu trúc vòng đời của Servlet

Sơ đồ sau miêu tả các giai đoạn trong vòng đời của một Servlet đặc trưng:

Đầu tiên, các HTTP Request tới Server và được đưa tới Container của Servlet.

Container Servlet tải các servlet trước khi gọi phương thức service().

Sau đó Container Servlet xử lý nhiều yêu cầu bởi việc tạo nhiều thread, mỗi thread thực thi phương thức service() cho một đối tượng servlet đơn.



Composite Start Composite End