Vòng đời (Life Cycle) 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, nhưng bạn 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.

Đây là minh họa cho 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, bạn không phải làm gì cả với service() nhưng việc bạn ghi đè phương thức hoặc doGet() hoặc doPost tùy thuộc vào kiểu yêu cầu mà bạn 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 bạn 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.