**Tổng quan ứng dụng Web**

|  |  |
| --- | --- |
|  |  |
| **Lỗi thông dụng** |  |
| **So sánh website động và website tĩnh** | |  |  | | --- | --- | | **Website tĩnh** | **Website động** | | Nội dung dựng sẵn và không thay đổi cho mỗi lần tải. | Nội dung được tạo ra nhanh chóng và thường xuyên thay đổi. | | Sử dụng mã HTML (hoặc thêm CSS, JavaScript) để phát triển một trang web. | Sử dụng ngôn ngữ phía máy chủ như PHP, SERVLET, JSP, và ASP.NET v.v... để phát triển một website. | | Trả về cùng một nội dung cho mỗi lần yêu cầu truy cập. | Nó có thể tạo ra HTML khác nhau cho mỗi yêu cầu. | | Nội dung website chỉ thay đổi khi ai đó xuất bản và cập nhật tệp tin lên trên máy chủ | Có thể thay đổi thông tin bằng cách thay đổi dữ liệu qua hệ thống quản lý nội dung (CMS) hoặc cơ sở dữ liệu. | | Lợi thế là truy cập nhanh chóng vì chỉ tải các tệp tin. | Lợi thế là có Hệ thống Quản lý Nội dung (CMS) thuận tiện thay đổi và quản lý nội dung. | |
| **Các phương thức HTTP request:** | |  |  | | --- | --- | | **HTTP Request** | **Mô tả** | | **GET** | Được dùng để lấy tài nguyên nằm ở địa chỉ được mô tả trong URL. | | **POST** | Được dùng để hỏi server chấp nhận thông tin được đính trong phần body của request ví dụ như file upload, thông tin khách hàng… Cấu tạo của request POST tương tự như GET được gắn hêm thông tin bổ sung. Phương thức POST thường được gửi đến server dưới dạng form. | | **HEAD** | Dùng để lấy thông tin về các header của request mà có thể nhận về. Cách hoạt động giống như GET nhưng phía server chỉ trả về các thông tin của header. | | **TRACE** | Trả về thông tin truy vấn mà máy chủ nhận được. Được sử dụng để kiểm tra truy vấn hoặc khắc phục sự cố. | | **PUT** | Thông báo bổ sung thông tin được đính kèm vào tài nguyên được mô tả ở URL | | **DELETE** | Thông báo xóa thông tin được đính kèm khỏi tài nguyên được mô tả ở URL. Ví dụ xóa bỏ một khách hàng với id được truyền đi. | | **OPTIONS** | Được sử dụng bởi Client để tìm ra danh sách các phương thức HTTP và các chức năng được hỗ trợ của một Server. | |
|  | |  |  | | --- | --- | | Định nghĩa cơ bản về 2 phương thức | 1. Phương thức get:  * Yêu cầu có thể được lưu trữ(cached). * Giữ lại trong lịch sử duyệt Web của trình   duyệt.   * Có thể được đánh dấu ghi nhớ lại   (bookmarked).   * Không được dùng để làm việc với những   dữ liệu nhạy cảm   * Có giới hạn về độ dài * Nên được sử dụng để nhận dữ liệu  1. Phương thức post:  * Yêu cầu sẽ không bao giờ được lưu trữ   (cached).   * Không lưu trữ lại trong lịch sử trình duyệt   Web.   * Không thể đánh dấu ghi nhớ lại(bookmarked). * Không bị giới hạn với độ dài của dữ liệu. | | Những điểm khác nhau giữa 2 phương pháp | |  |  |  | | --- | --- | --- | |  | Get | Post | | Nút quay lại/Tải lại | Không có vấn đề gì cả | Dữ liệu sẽ bị gửi lại(Trình duyệt sẽ hỏi ý kiến của bạn) | | Đánh dấu(bookmarked) | Có thể | Không thể | | Lưu trữ(cached) | Có thể | Không thể | | Kiểu mã hóa(encoding type) | application/x-www-form-urlencoded | application/x-www-form-urlencoded hoặc multipart/form-data sử dụng các multipart encoding cho các tập dữ liệu nhị phân(binary data) | | Lịch sử | Các thông số được lưu lại trong trình duyệt. | Các thông số không được lưu lại. | | Độ dài giới hạn | Có giới hạn, khi sử dụng phương thức GET thì sẽ nhập dữ liệu lên thanh URL(độ dài giới hạn là 2048 kí tự) | Không giới hạn | | Kiểu dữ liệu giới hạn | Chỉ có các kí tự ASCII | Không giới hạn, mã nhị phân cũng được. | | Bảo mật | GET ít bảo mật hơn so với Post do các dữ liệu gửi được hiển thị trên URL.  Không bao giờ sử dụng GET để gửi các thông tin như mật khẩu hay các dữ liệu nhạy cảm khác | POST thì an toàn hơn GET một ít do các dữ liệu không bị lưu lại trên trình duyệt hay trên server Web log. | | Hiển thị | Dữ liệu hiển thị trên URL | Không show dữ liệu | | |
| **Các phương thức của lớp GenericServlet** | 1. **public void init(ServletConfig config)** được dùng để khởi tạo servlet với cấu hình được cung cấp. 2. **public abstract void service(ServletRequest request, ServletResponse response)** là phương thức đón nhận các request đến. Nó được gọi ra vào mỗi khi người dùng gửi request đến một servlet. 3. **public void destroy()** được gọi một lần duy nhất trong suốt vòng đời của servlet để giải phóng một servlet và chỉ ra rằng servlet sẽ bị huỷ. 4. **public ServletConfig getServletConfig()** trả về đối tượng ServletConfig. 5. **public String getServletInfo()** trả về các thông tin về servlet như: người viết, bản quyền, phiên bản …. 6. **public void init()** là một phương thức thuận tiện trong việc lập trình, có thể khởi tạo servlet bằng phương thức này mà không cần tới super.init(config) 7. **public ServletContext getServletContext()** trả về đối tượng ServletContext. 8. **public String getInitParameter(String name)** trả về giá trị tham số ứng với tên tham số truyền vào 9. **public Enumeration getInitParameterNames()** trả về tất cả các tham số được định nghĩa trong tập tin web.xml. 10. **public String getServletName()** trả về tên của đối tượng servlet. 11. **public void log(String msg)**ghi thông điệp được cung cấp vào tập tin log của servlet 12. **public void log(String msg,Throwable t)**ghi thông điệp giải thích về lỗi vào tập tin log cũng như stack trace. |
| **Các phương thức của giao diện *Servlet*** | |  |  | | --- | --- | | **Phương thức** | **Mô tả** | | **public void init(ServletConfig config)** | Khởi chạy servlet. Nó là phương thức vòng đời và được gọi lên bởi web container chỉ 1 lần duy nhất. | | **public void service(ServletRequest request,ServletResponse response)** | cung cấp sự phản hồi tới yêu-cầu được gửi đến. Nó được gọi lên bởi web container mỗi khi có yêu-cầu tới. | | **public void destroy()** | được gọi lên chỉ 1 lần duy nhất và chỉ ra rằng servlet đang bị tiêu hủy. | | **public ServletConfig getServletConfig()** | trả về đối tượng ServletConfig. | | **public String getServletInfo()** | trả về thông tin về servlet như tác giả, bản quyền, phiên bản… | |
|  |  |
| **Các đặc tính của Servlet Container** | * **Standalone:**là kiểu máy chủ dựa trên java điển hình, trong đó servlet container và máy chủ web là được tích hợp trong 1 chương trình duy nhất. Ví dụ: Tomcat * **In-process:**servlet container được tách riêng ra khỏi máy chủ web, chạy trong không gian địa chỉ của máy chủ chính như 1 chương trình có thể cài cắm. Ví dụ: Tomcat chạy trong JBoss. * **Out-of-process:**Máy chủ web và servlet container là các chương trình khác nhau được chạy trong các tiến trình riêng biệt. Để thực hiện các giao tiếp giữa chúng, máy chủ web sử dụng 1 chương trình cài cắm đặc thù cho (và thường là cung cấp bởi) servlet container. |
| **Các phương thức của lớp *HttpServlet*** | 1. **public void service(*ServletRequest* req, *ServletResponse* res)**   chuyển tiếp yêu cầu tới phương thức service() bằng cách chuyển đổi yêu cầu và phản hồi thành những đối tượng HTTP   1. **protected void service(HttpServletRequest req, HttpServletResponse res)** nhận yêu cầu từ phương thức service, và chuyển tiếp yêu cầu đó tới các phương thức doXXX tùy thuộc vào phương thức của yêu cầu http được gửi tới. 2. **protected void doGet(HttpServletRequest req, HttpServletResponse res)**xử lý yêu cầu GET. Nó được gọi lên bởi web container. 3. **protected void doPost(HttpServletRequest req, HttpServletResponse res)** xử lý yêu cầu POST. Nó được sử dụng bởi web container. 4. **protected void doHead(HttpServletRequest req, HttpServletResponse res)** xử lý yêu cầu HEAD. Nó được sử dụng bởi web container. 5. **protected void doOptions(HttpServletRequest req, HttpServletResponse res)** xử lý yêu cầu OPTIONS. Nó được sử dụng bởi web container. 6. **protected void doPut(HttpServletRequest req, HttpServletResponse res)** xử lý yêu cầu PUT. Nó được sử dụng bởi web container. 7. **protected void doTrace(HttpServletRequest req, HttpServletResponse res)** xử lý yêu cầu TRACE. Nó được sử dụng bởi web container. 8. **protected void doDelete(HttpServletRequest req, HttpServletResponse res)** xử lý yêu cầu DELETE. Nó được sử dụng bởi web container.**protected long getLastModified(HttpServletRequest req)**trả về thời điểm sửa đổi cuối cùng của đối tượng yêu cầu được cung cấp, tính từ ngày 1/1/1970 theo múi giờ GMT. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |