**BẢNG ĐÁNH GIÁ ĐIỂM THUYẾT TRÌNH**

**MÔN LẬP TRÌNH JAVA NÂNG CAO**

**LỚP CTK44-PM**

|  |  |
| --- | --- |
| **Thông tin đề tài thuyết trình** | **Thông tin sinh viên thực hiện đánh giá** |
| Tên đề tài: JSP Template Engine | MSSV của sinh viên đánh giá: 2012353 |
| Điểm tổng kết: Ngày trình bày: 19/05/2023 | Họ tên sinh viên & ký xác nhận: Hạ  Đoàn Cao Nhật Hạ |
| ***Thông tin sinh viên thuyết trình*** |
| MSSV: 2011368  Họ tên: Bùi Văn Du |

1. **ĐÁNH GIÁ THUYẾT TRÌNH**

**Thang điểm: Kém 0-5; Trung bình: 5-7; Khá 7-8; Tốt 8-9; Xuất sắc: 9-10**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Bảng chấm điểm nhóm thuyết trình** | | | | | | |
| Các chủ đề chính có được xác định? (A) | Slide thuyết trình có thiết kế tốt (B) | Giao tiếp với người nghe bằng ánh mắt (C) | Giao tiếp với người nghe bằng giọng nói (D) | Khả năng trình bày nội dung (E) | Thảo luận & Hỏi đáp (F) | Điểm trung bình = (A+B+C+D+E+F)/6 |
| **9,8** | **9,5** | **9** | **9** | **9** | **9** | **9,22** |

1. **BÀI TẬP NẮM KIẾN THỨC**

Dựa vào nội dung vừa trình bày, trả lời các câu hỏi sau:

1. Trình bày tóm tắt nội dung anh/ chị đã nắm được về chủ đề vừa trình bày?
2. Khái niệm

* JavaServer Pages (JSP) là một công nghệ để phát triển các Trang web hỗ trợ nội dung động. Điều này giúp các nhà phát triển chèn mã java vào các trang HTML bằng cách sử dụng các thẻ JSP đặc biệt, hầu hết các thẻ này bắt đầu bằng <% và kết thúc bằng %>.
* Thành phần Trang JavaServer là một loại Java servlet được thiết kế để hoàn thành vai trò của giao diện người dùng cho ứng dụng web Java. Các nhà phát triển web viết JSP dưới dạng tệp văn bản kết hợp mã HTML hoặc XHTML, các phần tử XML và các lệnh và hành động JSP được nhúng.

1. Ưu điểm

So với Trang máy chủ hoạt động (ASP)

* Ưu điểm của JSP là gấp đôi. Đầu tiên, phần động được viết bằng Java, không phải Visual Basic hoặc ngôn ngữ cụ thể khác của MS, vì vậy nó mạnh hơn và dễ sử dụng hơn. Thứ hai, nó có thể di động đến các hệ điều hành khác và các máy chủ Web không phải của Microsoft.

### So với Servlets thuần túy

* Việc viết (và sửa đổi!) HTML thông thường sẽ thuận tiện hơn là có nhiều câu lệnh println tạo ra HTML.

So với Bao gồm phía máy chủ (SSI)

* SSI thực sự chỉ dành cho các phần bổ sung đơn giản, không dành cho các chương trình "thực" sử dụng dữ liệu biểu mẫu, tạo kết nối cơ sở dữ liệu, v.v.

So với JavaScript

* JavaScript có thể tạo HTML động trên máy khách nhưng khó có thể tương tác với máy chủ web để thực hiện các tác vụ phức tạp như truy cập cơ sở dữ liệu và xử lý hình ảnh, v.v.

So với HTML tĩnh

* Tất nhiên, HTML thông thường không thể chứa thông tin động.

1. Kiến trúc

A diagram of a server

Description automatically generated with low confidence

* Như với một trang bình thường, trình duyệt sẽ gửi một yêu cầu HTTP đến máy chủ web.
* Máy chủ web nhận ra rằng yêu cầu HTTP dành cho một trang JSP và chuyển tiếp nó tới một công cụ JSP. Điều này được thực hiện bằng cách sử dụng URL hoặc trang JSP kết thúc bằng .jsp thay vì .html .
* Công cụ JSP tải trang JSP từ đĩa và chuyển đổi nó thành nội dung của servlet. Quá trình chuyển đổi này rất đơn giản, trong đó tất cả văn bản mẫu được chuyển đổi thành câu lệnh println( ) và tất cả các phần tử JSP được chuyển đổi thành mã Java. Mã này thực hiện hành vi động tương ứng của trang.
* Công cụ JSP biên dịch servlet thành một lớp thực thi và chuyển tiếp yêu cầu ban đầu tới một công cụ servlet.
* Một phần của máy chủ web được gọi là công cụ servlet tải lớp Servlet và thực thi nó. Trong quá trình thực thi, servlet tạo ra một đầu ra ở định dạng HTML. Đầu ra tiếp tục được chuyển đến máy chủ web bởi công cụ servlet bên trong phản hồi HTTP.
* Máy chủ web chuyển tiếp phản hồi HTTP tới trình duyệt của bạn dưới dạng nội dung HTML tĩnh.
* Cuối cùng, trình duyệt web xử lý trang HTML được tạo động bên trong phản hồi HTTP chính xác như thể đó là một trang tĩnh.

1. Cú pháp

* Cú pháp của Scriptlet

<% code fragment %>

* Cú pháp với XML

<jsp:scriptlet>

code fragment

</jsp:scriptlet>

* Viết HTML

<html>

<head><title>Hello World</title></head>

<body>

Hello World!<br/>

<%

out.println("Your IP address is " + request.getRemoteAddr());

%>

</body>

</html>

1. Client Request

* Accept

Tiêu đề này chỉ định các loại MIME mà trình duyệt hoặc các máy khách khác có thể xử lý. Giá trị của image / png hoặc image / jpeg là hai khả năng phổ biến nhất.

* Accept-Charset

Tiêu đề này chỉ định các bộ ký tự mà trình duyệt có thể sử dụng để hiển thị thông tin. Ví dụ: ISO-8859-1.

* Accept-Encoding

Tiêu đề này chỉ định các loại mã hóa mà trình duyệt biết cách xử lý. Giá trị của gzip hoặc nén là hai khả năng phổ biến nhất.

* Accept-Language

Tiêu đề này chỉ định ngôn ngữ ưa thích của máy khách trong trường hợp servlet có thể tạo ra kết quả bằng nhiều ngôn ngữ. Ví dụ: en, en-us, ru, v.v.

* Authorization

Tiêu đề này được khách hàng sử dụng để nhận dạng bản thân khi truy cập các trang web được bảo vệ bằng mật khẩu.

* Connection

Tiêu đề này cho biết liệu máy khách có thể xử lý các kết nối HTTP liên tục hay không. Kết nối liên tục cho phép máy khách hoặc trình duyệt khác truy xuất nhiều tệp với một yêu cầu duy nhất. Giá trị của Keep-Alive có nghĩa là nên sử dụng các kết nối liên tục.

* Content-Length

Tiêu đề này chỉ áp dụng cho các yêu cầu POST và cung cấp kích thước của dữ liệu POST tính bằng byte.

* Cookie

Tiêu đề này trả lại cookie cho các máy chủ đã gửi chúng trước đó đến trình duyệt.

* Host

Tiêu đề này chỉ định máy chủ lưu trữ và cổng như được đưa ra trong URL gốc.

* If-Modified-Since

Tiêu đề này chỉ ra rằng khách hàng chỉ muốn trang nếu nó đã được thay đổi sau ngày được chỉ định. Máy chủ gửi mã, 304 có nghĩa là tiêu đề Không sửa đổi nếu không có kết quả mới hơn.

* If-Unmodified-Since

Tiêu đề này là đảo ngược của If-Modified-Since; Nó chỉ định rằng hoạt động sẽ chỉ thành công nếu tài liệu cũ hơn ngày được chỉ định.

* Referer

Tiêu đề này cho biết URL của các trang web giới thiệu. Ví dụ: nếu bạn đang ở Trang web 1 và nhấp vào liên kết đến Trang web 2, URL của Trang web 1 được bao gồm trong tiêu đề Giới thiệu khi trình duyệt yêu cầu Trang web 2.

* User-Agent

Tiêu đề này xác định trình duyệt hoặc ứng dụng khách khác đưa ra yêu cầu và có thể được sử dụng để trả về nội dung khác nhau cho các loại trình duyệt khác nhau

* Cookie[] getCookies()

Trả về một mảng chứa tất cả các đối tượng Cookie mà máy khách đã gửi cùng với yêu cầu này.

* Enumeration getAttributeNames()

Trả về một Liệt kê chứa tên của các thuộc tính có sẵn cho yêu cầu này.

* Enumeration getHeaderNames()

Trả về một bảng liệt kê tất cả các tên đầu đề mà yêu cầu này chứa.

* Enumeration getParameterNames()

Trả về một bảng liệt kê các đối tượng Chuỗi chứa tên của các tham số chứa trong yêu cầu này.

* HttpSession getSession()

Trả về phiên hiện tại được liên kết với yêu cầu này hoặc nếu yêu cầu không có phiên, hãy tạo một phiên.

* HttpSession getSession(boolean create)

Trả về HttpSession hiện tại được liên kết với yêu cầu này hoặc, nếu không có phiên hiện tại và tạo là true, trả về một phiên mới.

* Locale getLocale()

Trả về Ngôn ngữ ưa thích mà máy khách sẽ chấp nhận nội dung, dựa trên tiêu đề Chấp nhận ngôn ngữ.

* Object getAttribute(String name)

Trả về giá trị của thuộc tính đã đặt tên là một Đối tượng, hoặc null nếu không có thuộc tính nào của tên đã cho tồn tại.

* ServletInputStream getInputStream()

Truy xuất phần thân của yêu cầu dưới dạng dữ liệu nhị phân bằng cách sử dụng ServletInputStream.

* String getAuthType()

Trả về tên của lược đồ xác thực được sử dụng để bảo vệ servlet, ví dụ: "BASIC" hoặc "SSL" hoặc null nếu JSP không được bảo vệ.

* String getCharacterEncoding()

Trả về tên của mã hóa ký tự được sử dụng trong nội dung của yêu cầu này.

String getContentType()

Trả về kiểu MIME của nội dung yêu cầu hoặc null nếu không xác định được kiểu.

* String getContextPath()

Trả về phần URI yêu cầu cho biết ngữ cảnh của yêu cầu.

String getHeader(String name)

Trả về giá trị của tiêu đề yêu cầu đã xác định dưới dạng Chuỗi.

* String getMethod()

Trả về tên của phương thức HTTP mà yêu cầu này được thực hiện, ví dụ: GET, POST hoặc PUT.

* String getParameter(String name)

Trả về giá trị của tham số request dưới dạng String hoặc null nếu tham số không tồn tại.

* String getPathInfo()

Trả về bất kỳ thông tin đường dẫn bổ sung nào được liên kết với URL mà máy khách đã gửi khi thực hiện yêu cầu này.

* String getProtocol()

Trả về tên và phiên bản của giao thức mà yêu cầu sử dụng.

* String getQueryString()

Trả về chuỗi truy vấn chứa trong URL yêu cầu sau đường dẫn.

* String getRemoteAddr()

Trả về địa chỉ Giao thức Internet (IP) của máy khách đã gửi yêu cầu.

* String getRemoteHost()

Trả về tên đủ điều kiện của máy khách đã gửi yêu cầu.

* String getRemoteUser()

Trả về thông tin đăng nhập của người dùng thực hiện yêu cầu này, nếu người dùng đã được xác thực hoặc null nếu người dùng chưa được xác thực.

* String getRequestURI()

Trả về một phần URL của yêu cầu này từ tên giao thức lên đến chuỗi truy vấn trong dòng đầu tiên của yêu cầu HTTP.

* String getRequestedSessionId()

Trả về ID phiên do máy khách chỉ định.

* String getServletPath()

Trả về phần URL của yêu cầu này gọi JSP.

* String[] getParameterValues(String name)

Trả về một mảng các đối tượng Chuỗi chứa tất cả các giá trị mà tham số yêu cầu đã cho có, hoặc null nếu tham số không tồn tại.

boolean isSecure()

Trả về boolean cho biết liệu yêu cầu này có được thực hiện bằng kênh bảo mật hay không, chẳng hạn như HTTPS.

* int getContentLength()

Trả về độ dài, tính bằng byte, của nội dung yêu cầu và được cung cấp bởi luồng đầu vào hoặc -1 nếu không xác định độ dài.

* int getIntHeader(String name)

Trả về giá trị của tiêu đề yêu cầu đã xác định dưới dạng int.

* int getServerPort()

Trả về số cổng mà yêu cầu này đã được nhận.

1. Server Reponse

* Allow

Tiêu đề này chỉ định các phương thức yêu cầu (GET, POST, v.v.) mà máy chủ hỗ trợ

* Cache-Control

Tiêu đề này chỉ định các trường hợp mà tài liệu phản hồi có thể được lưu trữ một cách an toàn. Nó có thể có các giá trị công khai, riêng tư hoặc không có bộ nhớ cache, v.v. Public có nghĩa là tài liệu có thể lưu vào bộ nhớ cache, Private có nghĩa là tài liệu dành cho một người dùng và chỉ có thể được lưu trữ trong bộ nhớ cache riêng tư (không chia sẻ) và không có bộ nhớ cache có nghĩa là tài liệu không bao giờ được lưu vào bộ nhớ cache.

* Connection

Tiêu đề này hướng dẫn trình duyệt có sử dụng các kết nối HTTP liên tục hay không. Giá trị close hướng dẫn trình duyệt không sử dụng các kết nối HTTP liên tục và giữ các phương tiện tồn tại bằng cách sử dụng các kết nối liên tục.

* Content-Disposition

Tiêu đề này cho phép bạn yêu cầu trình duyệt yêu cầu người dùng lưu phản hồi vào đĩa trong một tệp có tên đã cho.

* Content-Encoding

Tiêu đề này chỉ định cách trang được mã hóa trong quá trình truyền.

* Content-Language

Tiêu đề này biểu thị ngôn ngữ mà tài liệu được viết. Ví dụ: en, en-us, ru, v.v.

* Content-Length

Tiêu đề này cho biết số byte trong phản hồi. Thông tin này chỉ cần thiết nếu trình duyệt đang sử dụng kết nối HTTP liên tục (giữ sống).

* Content-Type

Tiêu đề này cung cấp loại MIME (Multipurpose Internet Mail Extension) của tài liệu phản hồi.

* Expires

Tiêu đề này chỉ định thời gian mà nội dung sẽ được coi là lỗi thời và do đó không còn được lưu vào bộ nhớ cache nữa.

* Last-Modified

Tiêu đề này cho biết khi nào tài liệu được thay đổi lần cuối. Sau đó, máy khách có thể lưu trữ tài liệu và cung cấp ngày bằng tiêu đề yêu cầu If-Modified-Since trong các yêu cầu sau.

* Location

Tiêu đề này phải được bao gồm trong tất cả các phản hồi có mã trạng thái trong những năm 300. Thao tác này sẽ thông báo cho trình duyệt về địa chỉ tài liệu. Trình duyệt tự động kết nối lại với vị trí này và truy xuất tài liệu mới.

* Refresh

Tiêu đề này chỉ định thời gian trình duyệt sẽ yêu cầu một trang cập nhật. Bạn có thể chỉ định thời gian theo số giây sau đó một trang sẽ được làm mới.

* Retry-After

Tiêu đề này có thể được sử dụng cùng với phản hồi 503 (Service Unavailable) để cho khách hàng biết bao lâu nó có thể lặp lại yêu cầu của mình.

* Set-Cookie

Tiêu đề này chỉ định một cookie được liên kết với trang.

* String encodeRedirectURL(String url)

Mã hóa URL được chỉ định để sử dụng trong phương thức sendRedirect hoặc, nếu không cần mã hóa, trả về URL không thay đổi.

* String encodeURL(String url)

Mã hóa URL được chỉ định bằng cách bao gồm ID phiên trong đó hoặc, nếu không cần mã hóa, trả về URL không thay đổi.

* boolean containsHeader(String name)

Trả về boolean cho biết tiêu đề phản hồi đã đặt tên đã được đặt chưa

* boolean isCommitted()

Trả về boolean cho biết phản hồi đã được cam kết hay chưa.

* void addCookie(Cookie cookie)

Thêm cookie được chỉ định vào phản hồi.

* void addDateHeader(String name, long date)

Thêm tiêu đề phản hồi với tên và ngày-giá trị đã cho.

* void addHeader(String name, String value)

Thêm tiêu đề phản hồi với tên và giá trị đã cho.

* void addIntHeader(String name, int value)

Thêm tiêu đề phản hồi với tên và giá trị số nguyên đã cho.

* void flushBuffer()

Buộc bất kỳ nội dung nào trong bộ đệm phải được ghi cho khách hàng.

* void reset()

Xóa mọi dữ liệu tồn tại trong bộ đệm cũng như mã trạng thái và tiêu đề.

* void resetBuffer()

Xóa nội dung của bộ đệm cơ bản trong phản hồi mà không xóa tiêu đề hoặc mã trạng thái..

* void sendError(int sc)

Gửi phản hồi lỗi cho máy khách bằng cách sử dụng mã trạng thái được chỉ định và xóa bộ đệm.

* void sendError(int sc, String msg)

Gửi phản hồi lỗi cho máy khách bằng cách sử dụng trạng thái được chỉ định.

* void sendRedirect(String location)

Gửi phản hồi chuyển hướng tạm thời đến máy khách bằng cách sử dụng URL vị trí chuyển hướng được chỉ định.

* void setBufferSize(int size)

Đặt kích thước bộ đệm ưa thích cho phần thân của phản hồi.

* void setCharacterEncoding(String charset)

Đặt mã hóa ký tự (bộ ký tự MIME) của phản hồi được gửi đến máy khách, ví dụ: thành UTF-8.

* void setContentLength(int len)

Đặt độ dài của nội dung trong phản hồi Trong HTTP servlets; phương pháp này cũng đặt tiêu đề HTTP Content-Length.

* void setContentType(String type)

Đặt loại nội dung của phản hồi được gửi đến máy khách, nếu phản hồi chưa được cam kết.

* void setDateHeader(String name, long date)

Đặt tiêu đề phản hồi với tên và giá trị ngày đã cho.

* void setHeader(String name, String value)

Đặt tiêu đề phản hồi với tên và giá trị đã cho.

* void setIntHeader(String name, int value)

Đặt tiêu đề phản hồi với tên và giá trị số nguyên đã cho.

* void setLocale(Locale loc)

Đặt ngôn ngữ của phản hồi, nếu phản hồi chưa được cam kết.

* void setStatus(int sc)

Đặt mã trạng thái cho phản hồi này.

1. Cookies processing

Cấu trúc cookie

* Cookie thường được đặt trong tiêu đề HTTP
* JSP đặt cookie có thể gửi các tiêu đề

HTTP/1.1 200 OK

Date: Fri, 04 Feb 2000 21:03:38 GMT

Server: Apache/1.3.9 (UNIX) PHP/4.0b3

Set-Cookie: name = xyz; expires = Friday, 04-Feb-07 22:03:38 GMT;

path = /; domain = tutorialspoint.com

Connection: close

Content-Type: text/html

* public void setDomain(String pattern)

Phương pháp này đặt miền mà cookie áp dụng; Ví dụ, tutorialspoint.com.

* public String getDomain()

Phương pháp này lấy miền mà cookie áp dụng; Ví dụ, tutorialspoint.com.

* public void setMaxAge(int expiry)

Phương pháp này đặt thời gian (tính bằng giây) sẽ trôi qua trước khi cookie hết hạn. Nếu bạn không đặt tùy chọn này, cookie sẽ chỉ tồn tại cho phiên hiện tại.

* public int getMaxAge()

Phương pháp này trả về tuổi tối đa của cookie, được chỉ định bằng giây, Theo mặc định, -1 cho biết cookie sẽ tồn tại cho đến khi trình duyệt tắt.

* public String getName()

Phương thức này trả về tên của cookie. Tên không thể thay đổi sau khi tạo.

* public void setValue(String newValue)

Phương pháp này đặt giá trị được liên kết với cookie.

* public String getValue()

Phương pháp này nhận được giá trị được liên kết với cookie.

* public void setPath(String uri)

Phương pháp này đặt đường dẫn mà cookie này áp dụng. Nếu bạn không chỉ định đường dẫn, cookie sẽ được trả về cho tất cả các URL trong cùng thư mục với trang hiện tại cũng như tất cả các thư mục con.

* public String getPath()

Phương pháp này lấy đường dẫn mà cookie này áp dụng.

* public void setSecure(boolean flag)

Phương pháp này đặt giá trị boolean cho biết liệu cookie chỉ nên được gửi qua các kết nối được mã hóa (tức là SSL).

* public void setComment(String purpose)

Phương pháp này chỉ định nhận xét mô tả mục đích của cookie. Nhận xét này rất hữu ích nếu trình duyệt trình bày cookie cho người dùng.

* public String getComment()

Phương thức này trả về nhận xét mô tả mục đích của cookie này hoặc null nếu cookie không có nhận xét.

1. Anh/ chị hãy đưa ra các nhận xét và góp ý cho sinh viên vừa trình bày?

* Bổ sung phần so sánh vs Thymeleaf template

Thymeleaf template và JSP (JavaServer Pages) template là hai công nghệ phổ biến trong việc phát triển giao diện người dùng trong ứng dụng web Java.

Cú pháp và đọc hiểu: Thymeleaf sử dụng cú pháp gần giống với HTML, trong khi JSP sử dụng cú pháp gần giống với Java. Do đó, Thymeleaf có cú pháp đơn giản hơn và dễ đọc hiểu hơn cho những người không quen thuộc với Java.

Độ phổ biến: JSP là một công nghệ cũ và phổ biến từ lâu trong việc phát triển giao diện người dùng Java. Thymeleaf là một công nghệ mới hơn và đã được chấp nhận rộng rãi trong cộng đồng phát triển Java.

Tích hợp với Spring: Thymeleaf được thiết kế đặc biệt để tích hợp tốt với Spring Framework. Nó cung cấp tích hợp sẵn với các tính năng của Spring như ràng buộc dữ liệu, xử lý sự kiện và quản lý layout template. JSP cũng có thể tích hợp với Spring, nhưng Thymeleaf có tích hợp mạnh mẽ hơn và cung cấp cú pháp dễ sử dụng hơn.

Tính linh hoạt: Thymeleaf cho phép bạn tạo các template độc lập với Java, có thể được xem trước và chỉnh sửa mà không cần chạy ứng dụng. Điều này giúp tăng tính linh hoạt và tốc độ phát triển. Trong khi đó, JSP yêu cầu phải biên dịch và triển khai ứng dụng để xem kết quả của template.

Hiệu suất: Thymeleaf thực hiện việc render template trực tiếp thành HTML, trong khi JSP tạo ra các servlet và sử dụng các công nghệ nền tảng web để render HTML. Do đó, trong một số tình huống, Thymeleaf có thể hiệu quả hơn về hiệu suất.

Hỗ trợ cho các công nghệ frontend: Thymeleaf cung cấp tích hợp tốt với các công nghệ frontend như CSS, JavaScript và thư viện JavaScript như Angular hoặc Vue.js. JSP cũng có thể hỗ trợ công nghệ frontend, nhưng không có tích hợp sẵn như Thymeleaf.

Tuy nhiên, việc chọn sử dụng Thymeleaf hay

JSP phụ thuộc vào nhu cầu và quen thuộc của bạn, cũng như các yêu cầu và quyết định của dự án cụ thể.