Tìm hiểu về JTSL

1. JTSL là gì?

JSP Standard Tag Library (JSTL) là thư viện thẻ chuẩn cung cấp các thẻ để kiểm soát hành vi trang, lặp đi lặp lại và các lệnh điều khiển, các thẻ quốc tế hóa, và các thẻ SQL. JSTL hỗ trợ tới các tác vụ phổ biến và có tính cấu trúc, ví dụ như các tính lặp và điều kiện, các thẻ để thao tác tài liệu XML, các thẻ đa ngôn ngữ, và các thẻ SQL. Nó cũng cung cấp Framework để tích hợp các Custom Tags với các thẻ JSTL.

JSTL là một phần của Java EE API và trong hầu hết các container servlet. Nhưng để sử dụng JSTL trong các trang JSP của bạn, chúng ta cần phải tải về các thư viện JSTL cho bộ chứa Servlet (container servlet). Chúng ta cần phải đặt các các thư viện này vào thư mục WEB-INF/lib của project. Nếu là Maven project thì cần phải khai báo các thư viện này trong pom.xml

1. Các chức năng của JTSL

Dựa trên chức năng mà JTSL được chia ra làm 5 loại: JTSL Core Tags, JTSL Formatting and Localization Tag, JTSL SQL Tags, JTSL XML Tags, JTSL Functions Tags.

1. JTSL Core Tags và các thẻ cơ bản

Các thẻ cơ bản cung cấp hỗ trợ cho bộ lặp (iteration), các điều kiện logic, bắt ngoại lệ, url, chuyển tiếp (forward) hoặc chuyển hướng (redirect), v.v... Để sử dụng JSTL core tags, bạn cần đưa nó vào trang JSP giống dưới đây:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

Các thẻ cơ bản và thuộc tính:

* Thẻ <c:out>

Hiển thị các kết quả của một biểu thức, tương tự như cách làm việc của <%=%>, khác biệt là <c:out> có thể sử dụng "." để truy cập vào các thuộc tính của đối tượng. Ví dụ, để truy cập dept.deptName chỉ cần sử dụng thẻ là <c:out value = "${dept.deptName}" />. Thẻ <c:out> có thể tự động thoát thẻ (escape) XML để nó không bị nhầm lẫn với một thẻ thực tế.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **value** | x | String |  | Thông tin ouput, nói cách khác là nội dung sẽ trình bày |
| **default** |  | String |  | Thông tin mặc định cho output khi mà giá trị value là null |
| **escapeXml** |  | String | true | Tự động thoát thẻ (escape) XML để nó không bị nhầm lẫn với một thẻ thực tế hay không? |

* Thẻ <c:set>

Được dùng để đặt giá trị cho các biến từ các biểu thức cho việc sử dụng biến trong ứng dụng.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **value** | x | String |  | Giá trị cho biến, có thể là biểu thức hoặc giá trị cụ thể |
| **target** |  | String |  | Tên biến mà nó sẽ cập nhật giá trị |
| **property** |  | String |  | Thuộc tính của đối tượng là biến mà nó sẽ cập nhật giá trị |
| **var** |  | String |  | Tên biến |
| **scope** |  | String | Page | Phạm vi của biến |

* Thẻ <c:remove>

Được dùng để gỡ bỏ một biến từ hoặc một scope đã xác định hoặc scope đầu tiên tại đó biến được tìm thấy (nếu không có scope nào được xác định). Hoạt động này thường không hữu ích, nhưng nó có thể giúp đỡ trong việc bảo đảm một JSP xóa bất kỳ nguồn nào trong phạm vi nó chịu trách nhiệm.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **var** | x | String |  | Tên biến |
| **scope** |  | String | Tất cả các phạm vi | Phạm vi của biến |

* Thẻ <c:catch>

Được dùng để bắt bất kỳ Throwable nào xuất hiện trong phần thân của nó và trưng bày nó một cách tùy ý. Một cách đơn giản, nó được sử dụng để xử lý lỗi.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **var** | x | String |  | Tên biến lưu trữ ngoại lệ nếu nó được ném ra từ khối lệnh trong bản thân <c:catch> tương ứng |

* Thẻ <c:if>:

Được sử dụng để kiểm tra một điều kiện nào đó là đúng hay sai. Thực hiện khối lệnh bên trong nếu điều kiện là đúng.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **test** |  | String |  | Điều kiện kiểm tra |
| **var** | x | String |  | Tên biến lưu trữ kết quả kiểm tra |
| **scope** |  | String | Page | Phạm vi lưu trữ biến chứa kết quả kiểm tra |

* Thẻ <c:choose>

Cho phép lựa chọn các trường hợp thay thế, hoạt động như khối switch trong java. Với các case là <c:when> và default là <c:otherwise>. Thẻ này không có thuộc tính.

* Thẻ <c:when>

Thể hiện cho một trường hợp trong khối thẻ <c:choose>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **test** | x | String |  | Điều kiện kiểm tra |

* Thẻ <c:otherwise>

Thể hiện cho trường hợp mặc định trong khối thẻ <c:choose>. Thẻ này không có thuộc tính.

* Thẻ <c:import>

Cho phép lấy dữ liệu từ trang khác (nó render ra kết quả của trang web cần lấy trước khi nhận).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **url** | x | String |  | Đường dẫn trang chứa nội dung cần lấy |
| **context** |  | String | Application hiện tại | / được theo sau bởi tên của Web application nội bộ |
| **charEncoding** |  | String | ISO-8859-1 | Bộ ký tự được sử dụng để mã hóa nội dung nhận |
| **var** |  | String |  | Tên biến lưu trữ dội dung nhận về |
| **scope** |  | String | Page | Phạm vi của biến |
| **varReader** |  | String |  | Tên một biến thay thế để trưng bày java.io.Reader |

* Thẻ <c:forEach>

Được sử dụng để duyệt các thành phần trong một mảng các đối tượng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **items** | x | String |  | Mảng dữ liệu để duyệt |
| **begin** |  | Integer | 0 | Vị trí bắt đầu duyệt |
| **end** |  | Integer | Vị trí thành phần cuối cùng | Vị trí kết thúc duyệt |
| **step** |  | Integer | 1 | Tiến trình qua mỗi vòng lặp |
| **var** |  | String |  | Tên biến thể hiện item, thành phần, hiện tại |
| **varStatus** |  | String |  | Tên biến thể hiện trạng thái vòng lặp |

* Thẻ <c:forTokens>

Được sử dụng để phần chia một chuỗi thành các tokens và tương tác thông qua mỗi token đó

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **items** | x | String |  | Mảng dữ liệu để duyệt |
| **begin** |  | Integer | 0 | Vị trí bắt đầu duyệt |
| **end** |  | Integer | Vị trí thành phần cuối cùng | Vị trí kết thúc duyệt |
| **step** |  | Integer | 1 | Tiến trình qua mỗi vòng lặp |
| **var** |  | String |  | Tên biến thể hiện item, thành phần, hiện tại |
| **varStatus** |  | String |  | Tên biến thể hiện trạng thái vòng lặp |
| **delims** |  | String |  | Ký tự để sử dụng như là Delimiter (dấu giới hạn) |

* Thẻ <c:url>

Được dùng để định dạng một URL thành một chuỗi và lưu giữ nó vào trong một biến. Thẻ này tự động thực hiện việc viết URL khi cần thiết.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **value** | x | String |  | url cơ sở |
| **context** |  | String | Application hiện tại | / được theo sau bởi tên của Web application nội bộ |
| **var** |  | String |  | Tên biến lưu trữ URL được xử lý |
| **scope** |  | String | Page | Phạm vi của biến |

* Thẻ <c:param>

Cho phép tham số yêu cầu URL thích hợp để được xác định với URL và nó thực hiện bất kỳ mã hóa URL cần thiết nào.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **name** | x | String |  | Tên param |
| **value** | x | String |  | Giá trị |

* Thẻ <c:redirect>

Được dùng để điều hướng trình duyệt tới một URL khác bởi việc cung cấp URL tự động, nó hỗ trợ các URL liên quan tới context, và nó cũng hỗ trợ thẻ <c:param> trong JSTL.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **url** | x | String |  | Url cần điều hướng |
| **context** |  | String | Application hiện tại | / Được theo sau bởi tên của Web application nội bộ |

1. JTSL Formatting and Localization Tags và các thẻ cơ bản

Nhóm các thẻ Formatting and Localization Tags trong JSTL được sử dụng để định dạng và hiển thị văn bản, ngày tháng, thời gian, và số cho các trang web có tính chất quốc tế (Có người dùng đa quốc gia). Để sử dụng Formatting and Localization Tags, bạn cần đưa nó vào trang JSP giống dưới đây:

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/html/fmt" %>

Các thẻ cơ bản và thuộc tính:

* Thẻ <fmt:formatNumber>

Được sử dụng để định dạng số, phần trăm và tiền tệ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc định** | **Mô tả** |
| **value** | x | Number |  | Giá trị số, sẽ được định dạng. |
| **type** |  | String | number | Xác định xem giá trị được định dạng như một số, tiền tệ hoặc tỷ lệ phần trăm. Các giá trị (number, currency, percel). |
| **pattern** |  | String |  | Khuôn định dạng (Formatting pattern). |
| **currencyCode** |  | String | Lấy theo miền địa phương mặc định. | Mã tiền tệ được sử dụng nếu các loại thuộc tính là tiền tệ. |
| **currencySymbol** |  | String | Lấy theo miền địa phương mặc định. | Các ký hiệu tiền tệ được sử dụng nếu các loại thuộc tính là tiền tệ. |
| **groupingUsed** |  | String | true | Có phân tách nhóm hay không khi định dạng đầu ra. |
| **maxIntegerDigits** |  | Integer |  | Số lượng tối đa các chữ số trong phần nguyên. |
| **minIntegerDigits** |  | Integer |  | Số lượng tối thiểu của các chữ số ở phần nguyên |
| **maxFractionDigits** |  | Integer |  | Số lượng tối đa các chữ số trong phần thập phân. |
| **minFractionDigits** |  | Integer |  | Số lượng tối thiểu các chữ số trong phần thập phân. |
| **var** |  | String | In ra màn hinh. | Tên của biến lưu trữ kết quả đã được định dạng. |
| **scope** |  | String |  | Phạm vi của biến lưu trữ trong var. |

Bảng sau hiển thị các mã hóa cho pattern:

|  |  |
| --- | --- |
| **Ký hiệu** | **Miêu tả** |
| 0 | Biểu diễn một ký số |
| E | Biểu diễn trong form dạng số mũ |
| # | Biểu diễn một ký số, hiển thị là 0 khi vắng mặt |
| . | Như một placeholder cho một decimal separator |
| , | Như một placeholder cho một grouping separator |
| ; | Phân biệt riêng rẽ các định dạng |
| - | Được sử dụng như là tiền tố cho số âm theo mặc định |
| % | Phần trăm |
| ? | Multiplies by 1000 and displays as per mille. |
| ¤ | Biểu diễn ký hiệu tiền tệ; được thay thế bởi ký hiệu tiền tệ bổ sung |
| X | Chỉ rằng bất kỳ ký tự nào khác có thể được sử dụng trong prefix hoặc suffix |
| ' | Được sử dụng để trích dẫn các ký tự đặc biệt trong một prefix hoặc suffix |

* Thẻ <fmt:parseNumber>

Được sử dụng để parse số, phần trăm và tiền tệ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc định** | **Mô tả** |
| **value** | x | String |  | String được phân tích. |
| **type** |  | String | number | Quyết định xem String được cung cấp để phân tích là số, tiền tệ hoặc phần trăm, mặc định là số. (number, currency, percentage). |
| **pattern** |  | String |  | Khuôn mẫu của String đã cho trong thuộc tính value được phân tích. |
| **parseLocale** |  | String | locale mặc định | Locale để sử dụng khi phân tích các giá trị bằng cách sử dụng khuôn mẫu (pattern) đã cho. |
| **integerOnly** |  | String | false | Chỉ định có hoặc không, chỉ các phần nguyên của giá trị được phân tích. |
| **var** |  | String | In ra trang | Tên của biến để lưu trữ kết quả phân tích, kết quả trả về là một số (java.lang.Number). |
| **scope** |  | String | page | Phạm vi của **var** để lưu trữ. |

* Thẻ <fmt:formatDate>

Được dùng để định dạng ngày tháng và thời gian thành một chuỗi theo khuôn mẫu cho chỉ định:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc định** | **Mô tả** |
| **value** | x | Date |  | Ngày tháng hoặc thời gian, dùng để định dạng. |
| **type** |  | String | date | Quyết định xem giá trị trong thuộc tính value là ngày tháng hoặc thời gian, hoặc cả hai. (date, time, both) |
| **dateStyle** |  | String | default | Cách định dạng kiểu ngày tháng. Định dạng ngày tháng có thể được xác định với ngữ nghĩa tương tự trong class **java.text.DateFormat**. |
| **timeStyle** |  | String | default | Cách định dạng kiểu thời gian. Định dạng thời gian có thể được xác định với ngữ nghĩa tương tự trong class **java.text.DateFormat**. |
| **pattern** |  | String |  | Mô hình được sử dụng cho ngày tháng và thời gian khi định dạng. |
| **timeZone** |  | String | Múi giờ mặc định | Múi giờ để đại diện cho thời gian định dạng. |
| **var** |  | String | Ghi ra trang | Tên của các biến để lưu trữ kết quả định dạng. |
| **scope** |  | String | page | Phạm vi lưu trữ của biến var. |

Các ký tự định dạng:

|  |  |  |
| --- | --- | --- |
| **Mã** | **Mục đích** | **Ví dụ** |
| G | Định danh cho kỷ nguyên | AD |
| y | Năm | 2002 |
| M | Tháng | April & 04 |
| d | Ngày trong tháng | 20 |
| h | Giờ(0-12) | 12 |
| H | Giờ (0 - 23) | 0 |
| m | Phút | 45 |
| s | Giây | 52 |
| S | Mili giây | 970 |
| E | Ngày trong tuần | Tuesday |
| D | Ngày trong năm | 180 |
| F | Ngày của tuần trong tháng | 2 (2nd Wed in month) |
| w | Tuần trong năm | 27 |
| W | Tuần trong tháng | 2 |
| a | Ký hiệu a.m./p.m. | PM |
| k | The hour(12-hour time) | 24 |
| K | The hour(24-hour time) | 0 |
| z | Múi giờ | Central Standard Time |
| ' |  | The escape for text |
| '' |  | The single quote |

* Thẻ <fmt:parseDate>

Được dùng để phân tích một string mô tả của ngày tháng hoặc thời gian.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc định** | **Mô tả** |
| **value** |  | String | Nội dung trong body của thẻ | Chuỗi mô tả ngày tháng được phân tích. |
| **type** |  | String | date | Quết định xem giá trị chỉ định trong value chứa ngày tháng (date) hay thời gian (time) hoặc cả hai (both). |
| **dateStyle** |  | String | default | Chỉ định kiểu ngày tháng (SHORT, LONG, FULL, MEDIUM hoặc DEFAULT). |
| **timeStyle** |  | String | default | Chỉ định kiểu thời gian (SHORT, LONG, FULL, MEDIUM hoặc DEFAULT). |
| **pattern** |  | String |  | Chỉ định khuôn mẫu của chuỗi thời gian sẽ được phân tích. |
| **timeZone** |  | String | Múi giờ (time zone) mặc định | Múi giờ để diễn giải nếu chuỗi ngày tháng có chứa thông tin thời gian bất kỳ. |
| **parseLocale** |  | String | Miền địa phương (locale) mặc định | Locale whose date time formatting will be used to parse the date time. Miền địa phương có định dạng ngày tháng thời gian sẽ được sử dụng để phân tích. |
| **var** |  | String | In ra trang | Tên của biến dùng để lưu trữ kết quả phân tích được. |
| **scope** |  | String | page | Phạm vi lưu trữ của biến var. |

* Thẻ <fmt:bundle>

Nạp gói tài nguyên và sẽ được sử dụng trong trang. Các gói tài nguyên mà thẻ này tải chứa các cặp khóa-giá trị (key,value) được sử dụng để quốc tế hoặc nội địa hóa các ứng dụng web. Sau đó, các thẻ <fmt:message> được sử dụng để hiển thị các value ứng với các key trong các gói tài nguyên đến đầu ra.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc định** | **Mô tả** |
| **baseName** | x | String |  | Tên đầy đủ của gói tài nguyên. Giống với quy ước tên đầy đủ của Java ('.' Được sử dụng để tách các tên package). |
| **prefix** |  | String |  | Khi được sử dụng với **<fmt:message>**, thuộc tính này là một tiền tố, chỉ định giá trị được thêm vào giá trị của key, để mỗi khi sử dụng bạn không cần phải cung cấp tiền tố lặp đi lặp lại. |

* Thẻ <fmt:setBundle>

Được sử dụng để tải một Resource bundle tương tư như thẻ <fmt:bundle> và lưu giữ nó trong biến scope đã đặt tên hoặc biến cấu hình bundle.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc định** | **Mô tả** |
| **baseName** | x | String |  | Tên đầy đủ của gói tài nguyên. Giống với quy ước tên đầy đủ của Java ('.' Được sử dụng để tách các tên package |
| **var** |  | String | Replace default | Tên biến để lưu bundle |
| **scope** |  | String | Page |  |

* Thẻ <fmt:message>

Được sử dụng để hiển thị các thông báo bản địa hóa bằng cách thay thế các key (khóa) bởi các thông điệp thực tế nạp từ file tài nguyên (resource bundle). Thẻ này hoạt động kết hợp với <fmt:bundle> được sử dụng để tải các gói tài nguyên.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **key** |  | String | Giá trị trong thân thẻ | Khóa của message để tìm kiếm trên bó tài nguyên. |
| **bundle** |  | String | Default bundle | Nguồn tài nguyên để tìm khóa (key). |
| **var** |  | String | Print to Page | Tên của biến, dùng để lưu trữ message tương ứng với key. |
| **scope** |  | String | page | Phạm vi của biến var. |

* Thẻ <fmt:timeZone>

Được sử dụng để xác định timezone (múi giờ) mà tất cả thẻ bên trong phần thân của nó sẽ sử dụng.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **value** | x | String |  | Giá trị của timezone |

* Thẻ <fmt:setTimeZone>

Được sử dụng để sao chép một đối tượng timezone vào trong biến scope đã xác định.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **value** | x | String | Giá trị trong thân thẻ | Giá trị của timezone |
| **var** |  | String | Replace default | Tên của biến, dùng để lưu trữ timezone tương ứng với key. |
| **scope** |  | String | page | Phạm vi của biến var. |

* Thẻ <fmt:setLocale>

Được sử dụng để lưu giữ Locale đã cung cấp trong biến cấu hình locale.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **value** | x | String | en\_US | Xác định hai phần mã hóa: một mã hóa ngôn ngữ ISO-639 và một mã hóa quốc gia ISO-3166 |
| **variant** |  | String |  | Biến trình duyệt cụ thể |
| **scope** |  | String | page | Phạm vi của biến var. |

* Thẻ <fmt:requestEncoding>

Được sử dụng để xác định kiểu mã hóa được sử dụng bởi các form mà chuyển dữ liệu tới Web application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thuộc tính** | **Bắt buộc** | **Kiểu dữ liệu** | **Mặc đinh** | **Mô tả** |
| **key** | x | String |  | Tên mã hóa ký tự bạn muốn áp dụng khi giải mã (decode) các tham số request |

1. JTSL SQL Tags và các thẻ cơ bản

Nhóm SQL Tags trong JSTL cung cấp các thẻ để tương tác với các Relational Database (RDBMSs), ví dụ như Oracle, MySQL, hoặc Microsoft SQL Server.

Đây là cú pháp để bao nhóm SQL Tags trong JSTL trong JSP:

<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/html/sql" %>

Các thẻ cơ bản và thuộc tính:

* Thẻ <sql:setDataSource>

Thẻ <sql:setDataSource> trong JSTL thiết lập biến cấu hình nguồn dữ liệu (data source) hoặc lưu giữ thông tin data-source trong một biến scope mà có thể được sử dụng như là input tới JSTL Database action khác.

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| driver | Tên của lớp JDBC driver để được đăng ký | Không | None |
| url | JDBC URL để kết nối Database | Không | None |
| user | Username của Database | Không | None |
| password | Password của Database | Không | None |
| dataSource | Database được chuẩn bị trước | Không | None |
| var | Tên biến để biểu diễn database | Không | Set default |
| scope | Phạm vi biến để biểu diễn database | Không | Page |

Ví dụ sau minh họa cách sử dụng của thẻ <sql:setDataSource> trong JSTL:

Giả sử chúng ta có thông tin sau về thiết lập MySQL database:

* Chúng ta đang sử dụng JDBC MySQL driver.
* Chúng ta sẽ kết nối tới TEST database trên thiết bị nội bộ.
* Chúng ta sẽ sử dụng user\_id và mypassword để truy cập TEST database.

Tất cả tham số trên sẽ đa dạng dựa trên MySQL của bạn hoặc bất kỳ thiết lập database khác. Ghi nhớ các tham số trên, và dưới đây là ví dụ đơn giản để sử dụng thẻ setDataSource trong JSTL.

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<html>

<head>

<title>JSTL sql:setDataSource Tag</title>

</head>

<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost/TEST"

user="user\_id" password="mypassword"/>

<sql:query dataSource="" sql="..." var="result" />

</body>

</html>

* Thẻ <sql:query> trong JSTL

Thẻ <sql:query> trong JSTL thực thi một lệnh SQL SELECT và lưu giữ kết quả trong một biến scope.

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| sql | Lệnh SQL để thực thi (nên trả về một ResultSet) | Không | Body |
| dataSource | Kết nối Database để sử dụng (ghi đè kết nối mặc định) | Không | Default database |
| maxRows | Số kết quả tối đa để lưu giữ trong biến | Không | Unlimited |
| startRow | Số hàng trong kết quả , tại đó bắt đầu record | Không | 0 |
| var | Tên biến biểu diễn Database | Không | Set default |
| scope | Phạm vi biến biểu diễn Database | Không | Page |

Ví dụ sau minh họa cách sử dụng của thẻ <sql:query> trong JSTL:

Để hiểu khái niệm cơ bản, chúng ta tạo Employees table trong TEST database và tạo một số record trong table đó như sau:

Bước 1:

Mở một Command Prompt và tạo thay đổi đến thư mục cài đặt như sau:

C:\>

C:\>cd Program Files\MySQL\bin

C:\Program Files\MySQL\bin>

Bước 2:

Login vào database như sau:

C:\Program Files\MySQL\bin>mysql -u root -p

Enter password: \*\*\*\*\*\*\*\*

mysql>

Bước 3:

Tạo Employee table trong TEST database như sau:

mysql> use TEST;

mysql> create table Employees

(

id int not null,

age int not null,

first varchar (255),

last varchar (255)

);

Query OK, 0 rows affected (0.08 sec)

mysql>

Tạo Data Record

Cuối cùng, bạn tạo một số record trong Employees table như sau:

mysql> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');

Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO Employees VALUES (101, 25, 'Mahnaz', 'Fatma');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (102, 30, 'Zaid', 'Khan');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (103, 28, 'Sumit', 'Mittal');

Query OK, 1 row affected (0.00 sec)

mysql>

Bây giờ, chúng ta viết một JSP mà sẽ sử dụng <sql:query> trong JSTL để thực thi một lệnh SQL SELECT như sau:

<%@ page import="java.io.\*,java.util.\*,java.sql.\*"%>

<%@ page import="javax.servlet.http.\*,javax.servlet.\*" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<html>

<head>

<title>JSTL sql:query Tag</title>

</head>

<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

<sql:query dataSource="" var="result">

SELECT \* from Employees;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Emp ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Age</th>

</tr>

<c:forEach var="row" items="">

<tr>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

</tr>

</c:forEach>

</table>

</body>

</html>

Bây giờ thử truy cập JSP trên, sẽ hiển thị kết quả sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Emp ID** | **First Name** | **Last Name** | **Age** |
| 100 | Zara | Ali | 18 |
| 101 | Mahnaz | Fatma | 25 |
| 102 | Zaid | Khan | 30 |
| 103 | Sumit | Mittal | 28 |

* Thẻ <sql:update> trong JSTL

Thẻ <sql:update> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| sql | Lệnh SQL để thực thi (không trả về ResultSet) | Không | Body |
| dataSource | Kết nối Database để sử dụng (ghi đè database mặc định) | Không | Default database |
| var | Tên biến để lưu giữ lượng đếm các hàng | Không | None |
| scope | Phạm vi biến để lưu giữ lượng đếm các hàng | Không | Page |

Ví dụ

Ví dụ sau minh họa cách sử dụng của thẻ <sql:update> trong JSTL:

Để hiểu khái niệm cơ bản, chúng ta tạo **Employees** table trong TEST database và tạo một số record trong table đó như sau:

Bước 1:

Mở một **Command Prompt** và tạo thay đổi đến thư mục cài đặt như sau:

C:\>

C:\>cd Program Files\MySQL\bin

C:\Program Files\MySQL\bin>

Bước 2:

Login vào database như sau:

C:\Program Files\MySQL\bin>mysql -u root -p

Enter password: \*\*\*\*\*\*\*\*

mysql>

Bước 3:

Tạo **Employee** table trong **TEST** database như sau:

mysql> use TEST;

mysql> create table Employees

(

id int not null,

age int not null,

first varchar (255),

last varchar (255)

);

Query OK, 0 rows affected (0.08 sec)

mysql>

Tạo Data Record

Cuối cùng, bạn tạo một số record trong Employees table như sau:

mysql> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');

Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO Employees VALUES (101, 25, 'Mahnaz', 'Fatma');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (102, 30, 'Zaid', 'Khan');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (103, 28, 'Sumit', 'Mittal');

Query OK, 1 row affected (0.00 sec)

mysql>

Bây giờ, chúng ta viết một JSP mà sẽ sử dụng <sql:update> trong JSTL để thực thi một lệnh SQL INSERT để tạo một record trong table như sau:

<%@ page import="java.io.\*,java.util.\*,java.sql.\*"%>

<%@ page import="javax.servlet.http.\*,javax.servlet.\*" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<html>

<head>

<title>JSTL sql:update Tag</title>

</head>

<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

<sql:update dataSource="" var="count">

INSERT INTO Employees VALUES (104, 2, 'Nuha', 'Ali');

</sql:update>

<sql:query dataSource="" var="result">

SELECT \* from Employees;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Emp ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Age</th>

</tr>

<c:forEach var="row" items="">

<tr>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

</tr>

</c:forEach>

</table>

</body>

</html>

Bây giờ truy cập JSP trên, sẽ hiển thị kết quả sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Emp ID** | **First Name** | **Last Name** | **Age** |
| 100 | Zara | Ali | 18 |
| 101 | Mahnaz | Fatma | 25 |
| 102 | Zaid | Khan | 30 |
| 103 | Sumit | Mittal | 28 |
| 104 | Nula | Ali | 2 |

Theo cách tương tự, bạn có thể thử các lệnh SQL UPDATE, và DELETE trên bảng giống như thế.

* Thẻ <sql:param> trong JSTL

Thẻ <sql:param> trong JSTL được sử dụng như là một action được lồng cho các thẻ <sql:query> và <sql:update> để cung cấp một giá trị placeholder. Nếu giá trị null được cung cấp, giá trị được thiết lập là SQL NULL cho placeholder.

Thuộc tính

Thẻ <sql:param> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| value | Giá trị của tham số để thiết lập | Không | Body |

Ví dụ

Ví dụ sau minh họa cách sử dụng của thẻ <sql:param> trong JSTL:

Để hiểu khái niệm cơ bản, chúng ta tạo **Employees** table trong TEST database và tạo một số record trong table đó như sau:

Bước 1:

Mở một **Command Prompt** và tạo thay đổi đến thư mục cài đặt như sau:

C:\>

C:\>cd Program Files\MySQL\bin

C:\Program Files\MySQL\bin>

Bước 2:

Login vào database như sau:

C:\Program Files\MySQL\bin>mysql -u root -p

Enter password: \*\*\*\*\*\*\*\*

mysql>

Bước 3:

Tạo **Employee** table trong **TEST** database như sau:

mysql> use TEST;

mysql> create table Employees

(

id int not null,

age int not null,

first varchar (255),

last varchar (255)

);

Query OK, 0 rows affected (0.08 sec)

mysql>

Tạo Data Record

Cuối cùng, bạn tạo một số record trong Employees table như sau:

mysql> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');

Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO Employees VALUES (101, 25, 'Mahnaz', 'Fatma');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (102, 30, 'Zaid', 'Khan');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (103, 28, 'Sumit', 'Mittal');

Query OK, 1 row affected (0.00 sec)

mysql>

Bây giờ, chúng ta viết một JSP mà sẽ sử dụng <sql:param> để thực thi một lệnh SQL DELETE để xóa một record với id = 103 từ table như sau:

<%@ page import="java.io.\*,java.util.\*,java.sql.\*"%>

<%@ page import="javax.servlet.http.\*,javax.servlet.\*" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<html>

<head>

<title>JSTL sql:param Tag</title>

</head>

<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

<c:set var="empId" value="103"/>

<sql:update dataSource="" var="count">

DELETE FROM Employees WHERE Id = ?

<sql:param value="" />

</sql:update>

<sql:query dataSource="" var="result">

SELECT \* from Employees;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Emp ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Age</th>

</tr>

<c:forEach var="row" items="">

<tr>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

</tr>

</c:forEach>

</table>

</body>

</html>

Bây giờ truy cập JSP trên, sẽ hiển thị kết quả sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Emp ID** | **First Name** | **Last Name** | **Age** |
| 100 | Zara | Ali | 18 |
| 101 | Mahnaz | Fatma | 25 |
| 102 | Zaid | Khan | 30 |

Bạn có thể thử <sql:param> với các lệnh SQL UPDATE và SELECT theo cách tương tự chúng tôi đã sử dụng với lệnh DELETE.

* Thẻ <sql:dateParam>

Thẻ <sql:dateParam> trong JSTL, được sử dụng như là một action được lồng vào cho các thẻ <sql:query> và <sql:update>, cung cấp một giá trị date và time cho một giá trị placeholder. Nếu giá trị null được cung cấp, giá trị được thiết lập là SQL NULL cho placeholder.

Thuộc tính

Thẻ <sql:dateParam> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| value | Giá trị của tham số date để được thiết lập (java.util.Date) | Không | Body |
| type | DATE (chỉ có date), TIME (chỉ có time), hoặc TIMESTAMP (cả date và time) | Không | TIMESTAMP |

Ví dụ

Để hiểu khái niệm cơ bản, chúng ta tạo **Students** table trong **TEST** database và tạo một số record trong table đó như sau:

Bước 1:

Mở một **Command Prompt** và tạo thay đổi đến thư mục cài đặt như sau:

C:\>

C:\>cd Program Files\MySQL\bin

C:\Program Files\MySQL\bin>

Bước 2:

Login vào database như sau:

C:\Program Files\MySQL\bin>mysql -u root -p

Enter password: \*\*\*\*\*\*\*\*

mysql>

Bước 3:

Tạo **Students** table trong **TEST** database như sau:

mysql> use TEST;

mysql> create table Students

(

id int not null,

first varchar (255),

last varchar (255),

dob date

);

Query OK, 0 rows affected (0.08 sec)

mysql>

Tạo Data Records

Cuối cùng, bạn tạo một số record trong *Students* table như sau:

mysql> INSERT INTO Students

VALUES (100, 'Zara', 'Ali', '2002/05/16');

Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO Students

VALUES (101, 'Mahnaz', 'Fatma', '1978/11/28');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Students

VALUES (102, 'Zaid', 'Khan', '1980/10/10');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Students

VALUES (103, 'Sumit', 'Mittal', '1971/05/08');

Query OK, 1 row affected (0.00 sec)

mysql>

Bây giờ, chúng ta viết một JSP mà sẽ sử dụng <sql:update> cùng với thẻ <sql:param> và <sql:dataParam> trong JSTL để thực thi một lệnh SQL UPDATE để cập nhật ngày sinh (DoB) của Zara như sau:

<%@ page import="java.io.\*,java.util.\*,java.sql.\*"%>

<%@ page import="javax.servlet.http.\*,javax.servlet.\*" %>

<%@ page import="java.util.Date,java.text.\*" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<html>

<head>

<title>JSTL sql:dataParam Tag</title>

</head>

<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost/TEST"

user="root" password="pass123"/>

<%

Date DoB = new Date("2001/12/16");

int studentId = 100;

%>

<sql:update dataSource="" var="count">

UPDATE Students SET dob = ? WHERE Id = ?

<sql:dateParam value="<%=DoB%>" type="DATE" />

<sql:param value="<%=studentId%>" />

</sql:update>

<sql:query dataSource="" var="result">

SELECT \* from Students;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Emp ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>DoB</th>

</tr>

<c:forEach var="row" items="">

<tr>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

</tr>

</c:forEach>

</table>

</body>

</html>

Bây giờ truy cập JSP trên, sẽ hiển thị kết quả sau khi cập nhật record với id=100:

|  |  |  |  |
| --- | --- | --- | --- |
| **Emp ID** | **First Name** | **Last Name** | **DoB** |
| 100 | Zara | Ali | 2001-12-16 |
| 101 | Mahnaz | Fatma | 1978-11-28 |
| 102 | Zaid | Khan | 1980-10-10 |
| 103 | Sumit | Mittal | 1971-05-08 |

* Thẻ <sql:transaction>

Thẻ <sql:transaction> trong JSTL được sử dụng để nhóm các thẻ <sql:query> và <sql:update> vào trong **transaction** (giao tác: một tập các action cùng thực hiện một chức năng). Bạn có thể đặt số lượng các lệnh bất kỳ <sql:query> và <sql:update> bên trong <sql:transaction> để làm cho chúng thành một transaction đơn.

Nó bảo đảm rằng việc sửa đổi Database được thực hiện bởi các action được lồng vào nhau hoặc được commit hoặc được roll back nếu một *exception* được ném bởi bất kỳ action được lồng vào nào.

Thuộc tính

Thẻ <sql:transaction> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| dataSource | Kết nối Database để sử dụng (ghi đè mặc định) | Không | Default database |
| isolation | Mức độ cô lập transaction (READ\_COMMITTED, READ\_UNCOMMITTED, REPEATABLE\_READ, hoặc SERIALIZABLE) | Không | Database's default |

Ví dụ

Ví dụ sau minh họa cách sử dụng của thẻ <sql:transaction> trong JSTL:

Để hiểu khái niệm cơ bản, chúng ta tạo **Students** table trong TEST database và tạo một số record trong table đó như sau:

Bước 1:

Mở một **Command Prompt** và tạo thay đổi đến thư mục cài đặt như sau:

C:\>

C:\>cd Program Files\MySQL\bin

C:\Program Files\MySQL\bin>

Bước 2:

Login vào database như sau:

C:\Program Files\MySQL\bin>mysql -u root -p

Enter password: \*\*\*\*\*\*\*\*

mysql>

Bước 3:

Tạo **Students** table trong **TEST** database như sau:

mysql> use TEST;

mysql> create table Students

(

id int not null,

first varchar (255),

last varchar (255),

dob date

);

Query OK, 0 rows affected (0.08 sec)

mysql>

Tạo Data Records

Cuối cùng, bạn tạo một số record trong Students table như sau:

mysql> INSERT INTO Students

VALUES (100, 'Zara', 'Ali', '2002/05/16');

Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO Students

VALUES (101, 'Mahnaz', 'Fatma', '1978/11/28');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Students

VALUES (102, 'Zaid', 'Khan', '1980/10/10');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Students

VALUES (103, 'Sumit', 'Mittal', '1971/05/08');

Query OK, 1 row affected (0.00 sec)

mysql>

Bây giờ, chúng ta viết một JSP mà sẽ sử dụng <sql:update> cùng với <sql:transaction> trong JSTL để thực thi một lệnh SQL UPDATE. Tại đây, code bên trong <sql:transaction> hoặc sẽ được thực thi một cách hoàn toàn hoặc không được thực thi bất cứ cái gì.

<%@ page import="java.io.\*,java.util.\*,java.sql.\*"%>

<%@ page import="javax.servlet.http.\*,javax.servlet.\*"%>

<%@ page import="java.util.Date,java.text.\*" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<html>

<head>

<title>JSTL sql:transaction Tag</title>

</head>

<body>

<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost/TEST"

user="root" password="cohondob"/>

<%

Date DoB = new Date("2001/12/16");

int studentId = 100;

%>

<sql:transaction dataSource="">

<sql:update var="count">

UPDATE Students SET last = 'Ali' WHERE Id = 102

</sql:update>

<sql:update var="count">

UPDATE Students SET last = 'Shah' WHERE Id = 103

</sql:update>

<sql:update var="count">

INSERT INTO Students

VALUES (104,'Nuha', 'Ali', '2010/05/26');

</sql:update>

</sql:transaction>

<sql:query dataSource="" var="result">

SELECT \* from Students;

</sql:query>

<table border="1" width="100%">

<tr>

<th>Emp ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>DoB</th>

</tr>

<c:forEach var="row" items="">

<tr>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

<td><c:out value=""/></td>

</tr>

</c:forEach>

</table>

</body>

</html>

Bây giờ truy cập JSP trên, sẽ hiển thị kết quả sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Emp ID** | **First Name** | **Last Name** | **DoB** |
| 100 | Zara | Ali | 2001-12-16 |
| 101 | Mahnaz | Fatma | 1978-11-28 |
| 102 | Zaid | Ali | 1980-10-10 |
| 103 | Sumit | Shah | 1971-05-08 |
| 104 | Nuha | Ali | 2010-05-26 |

1. JTSL XML Tags và các thẻ cơ bản

Nhóm XML Tags trong JSTL cung cấp một cách để tạo và thao tác các tài liệu XML. Sau đây là cú pháp để include nhóm XML Tags trong JSP của bạn:

Thư viện thẻ XML Tags trong JSTL có các custom tags để tương tác với XML data. Điều này bao gồm việc parse XML, truyền tải XML data, và điều khiển dòng dữ liệu dựa trên các XPath Expression.

<%@ taglib prefix="x"

uri="http://java.sun.com/jsp/html/xml" %>

* Thẻ <x:out>

Thẻ <x:out> trong JSTL hiển thị kết quả của một XPath expression. Tính năng của nó tương tự như cú pháp <%= %> trong JSP.

## Thuộc tính

Thẻ <x:out> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| select | XPath expression để ước lượng như một string, thường sử dụng các biến XPath | Có | None |
| escapeXml | True nếu thẻ nên tránh các ký tự XML đặc biệt | Không | true |

## Ví dụ

Ví dụ sau sẽ sử dụng các thẻ <x:out> và thẻ <x:parse> trong JSTL.

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<html>

<head>

<title>JSTL x:out Tags</title>

</head>

<body>

<h3>Books Info:</h3>

<c:set var="xmltext">

<books>

<book>

<name>Padam History</name>

<author>ZARA</author>

<price>100</price>

</book>

<book>

<name>Great Mistry</name>

<author>NUHA</author>

<price>2000</price>

</book>

</books>

</c:set>

<x:parse xml="" var="output"/>

<b>The title of the first book is</b>:

<x:out select="$output/books/book[1]/name" />

<br>

<b>The price of the second book</b>:

<x:out select="$output/books/book[2]/price" />

</body>

</html>

Nó sẽ cho kết quả sau:

### Books Info:

**The title of the first book is**: Padam History

**The price of the second book**: 2000

* Thẻ <x:parse>

Thẻ <x:parse> trong JSTL được sử dụng để parse dữ liệu XML được xác định hoặc thông qua một thuộc tính hoặc trong phần thân thẻ.

## Thuộc tính

Thẻ <x:parse> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| var | Một biến mà chứa dữ liệu XML được parse | Không | None |
| xml | Text của tài liệu để parse (là String hoặc Reader) | Không | Body |
| systemId | URI của System identifier để parse tài liệu | Không | None |
| filter | Filter để được áp dụng cho tài liệu nguồn (source document) | Không | None |
| doc | Tài liệu XML để được parse | Không | Page |
| scope | Phạm vi biến được xác định trong thuộc tính var | Không | Page |
| varDom | Biến mà chứa dữ liệu XML đã được parse | Không | Page |
| scopeDom | Phạm vi biến được xác định trong thuộc tính varDom | Không | Page |

## Ví dụ

Ví dụ sau minh họa cách thẻ <x:parse> trong JSTL được sử dụng để đọc XML file ngoại vi.

Chúng ta đã thấy cách chúng ta có thể parse XML từ phần thân của tài liệu đã cho. Bây giờ, đặt nội dung sau trong books.xml file:

<books>

<book>

<name>Padam History</name>

<author>ZARA</author>

<price>100</price>

</book>

<book>

<name>Great Mistry</name>

<author>NUHA</author>

<price>2000</price>

</book>

</books>

Thử main.jsp, giữ trong cùng thư mục:

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<html>

<head>

<title>JSTL x:parse Tags</title>

</head>

<body>

<h3>Books Info:</h3>

<c:import var="bookInfo" url="http://localhost:8080/books.xml"/>

<x:parse xml="" var="output"/>

<b>The title of the first book is</b>:

<x:out select="$output/books/book[1]/name" />

<br>

<b>The price of the second book</b>:

<x:out select="$output/books/book[2]/price" />

</body>

</html>

Bây giờ thử truy cập JSP trên bởi sử dụng http://localhost:8080/main.jsp, nó sẽ cho kết quả sau:

### Books Info:

**The title of the first book is**:Padam History

**The price of the second book**: 2000

* Thẻ <x:set>

Thẻ <x:set> trong JSTL thiết lập một biến tới giá trị của một XPath expression.

Nếu XPath expression cho kết quả là một Boolean, thì <x:set> thiết lập một đối tượng là java.lang.Boolean; cho một string, nó thiết lập là java.lang.String; và cho một number, nó thiết lập là java.lang.Number.

## Thuộc tính

Thẻ <x:set> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| var | Một biến mà được thiết lập tới giá trị của XPath expression | Có | Body |
| select | XPath expression để được ước lượng | Không | None |
| scope | Pham vi biến được xác định trong thuộc tính var | Không | Page |

## Ví dụ

Ví dụ sau minh họa cách sử dụng của thẻ <x:set> trong JSTL:

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<html>

<head>

<title>JSTL Tags</title>

</head>

<body>

<h3>Books Info:</h3>

<c:set var="xmltext">

<books>

<book>

<name>Padam History</name>

<author>ZARA</author>

<price>100</price>

</book>

<book>

<name>Great Mistry</name>

<author>NUHA</author>

<price>2000</price>

</book>

</books>

</c:set>

<x:parse xml="" var="output"/>

<x:set var="fragment" select="$output//book"/>

<b>The price of the second book</b>:

<c:out value="" />

</body>

</html>

Thử truy cập JSP trên, nó sẽ cho kết quả sau:

### Books Info:

**The price of the second book**:[[book: null], [book: null]]

* Thẻ <x:if>

Thẻ <x:if> trong JSTL ước lượng một XPath expression để kiểm tra và nếu là true, nó xử lý phần thân của nó. Nếu điều kiện kiểm tra là false, phần thân bị bỏ qua.

## Thuộc tính

Thẻ <x:if> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| select | XPath expression để được ước lượng | Có | None |
| var | Tên biến để lưu giữ kết quả của điều kiện | Không | None |
| scope | Phạm vi biến được xác định trong thuộc tính var | Không | Page |

## Ví dụ

Ví dụ sau minh họa cách sử dụng của thẻ <x:if> trong JSTL:

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<html>

<head>

<title>JSTL x:if Tags</title>

</head>

<body>

<h3>Books Info:</h3>

<c:set var="xmltext">

<books>

<book>

<name>Padam History</name>

<author>ZARA</author>

<price>100</price>

</book>

<book>

<name>Great Mistry</name>

<author>NUHA</author>

<price>2000</price>

</book>

</books>

</c:set>

<x:parse xml="" var="output"/>

<x:if select="$output//book">

Document has at least one <book> element.

</x:if>

<br />

<x:if select="$output/books[1]/book/price > 100">

Book prices are very high

</x:if>

</body>

</html>

Thử truy cập JSP trên, nó sẽ cho kết quả sau:

### Books Info:

Document has at least one <book> element.

Book prices are very high

Thẻ <x:forEach> trong JSTL được sử dụng để lặp qua các node trong một tài liệu XML.

## Thuộc tính

Thẻ <x:forEach> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| select | XPath expression để được ước lượng | Có | None |
| var | Tên biến để lưu giữ item hiện tại cho mỗi vòng lặp | Không | None |
| begin | Index bắt đầu cho tính lặp | Không | None |
| end | Index kết thúc cho tính lặp | Không | None |
| step | Kích thước của việc lượng gia index trong khi lặp qua tập hợp | Không | None |
| varStatus | Tên biến trong đó trạng thái của vòng lặp được lưu giữ | Không | None |

## Ví dụ

Ví dụ sau minh họa cách sử dụng của thẻ <x:forEach> trong JSTL:

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<html>

<head>

<title>JSTL x:if Tags</title>

</head>

<body>

<h3>Books Info:</h3>

<c:set var="xmltext">

<books>

<book>

<name>Padam History</name>

<author>ZARA</author>

<price>100</price>

</book>

<book>

<name>Great Mistry</name>

<author>NUHA</author>

<price>2000</price>

</book>

</books>

</c:set>

<x:parse xml="" var="output"/>

<ul class="list">

<x:forEach select="$output/books/book/name" var="item">

<li>Book Name: <x:out select="$item" /></li>

</x:forEach>

</ul>

</body>

</html>

Nó sẽ cho kết quả:

### Books Info:

* Book Name: Padam History

* Book Name: Great Mistry
* Thẻ <x:choose>

Thẻ <x:choose> trong JSTL giống như một lệnh **switch** trong Java, cho phép bạn lựa chọn giữa các tùy chọn thay thế nhau. Ở đây, lệnh switch có các lệnh **case**, thì thẻ <x:choose> có các thẻ <x:when>. Một lệnh switch có mệnh đề **default** để xác định một hành động mặc định và tương tự như vậy, thẻ <x:choose> có các mệnh đề mặc định là thẻ <x:otherwise>.

## Thuộc tính

* Thẻ <x:choose> không có bất kỳ thuộc tính nào.
* Thẻ <x:when> có một thuộc tính, được liệt kê dưới đây.
* Thẻ <x:otherwise> không có bất kỳ thuộc tính nào.

Thẻ <x:when> trong JSTL có thuộc tính:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| select | Điều kiện đề ước lượng | Có | None |

## Ví dụ

Ví dụ sau minh họa cách sử dụng của thẻ <x:choose>, <x:when>, <x:otherwise> trong JSTL:

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<html>

<head>

<title>JSTL x:choose Tags</title>

</head>

<body>

<h3>Books Info:</h3>

<c:set var="xmltext">

<books>

<book>

<name>Padam History</name>

<author>ZARA</author>

<price>100</price>

</book>

<book>

<name>Great Mistry</name>

<author>NUHA</author>

<price>2000</price>

</book>

</books>

</c:set>

<x:parse xml="" var="output"/>

<x:choose>

<x:when select="$output//book/author = 'ZARA'">

Book is written by ZARA

</x:when>

<x:when select="$output//book/author = 'NUHA'">

Book is written by NUHA

</x:when>

<x:otherwise>

Unknown author.

</x:otherwise>

</x:choose>

</body>

</html>

Nó sẽ cho kết quả:

### Books Info:

Book is written by ZARA

* Thẻ <x:transform>

Thẻ <x:transform> trong JSTL áp dụng một XSL transformation (chuyển đổi) trên một tài liệu XML.

## Thuộc tính

Thẻ <x:transform> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| doc | Tài liệu XML nguồn cho sự chuyển đổi XSLT | Không | Body |
| docSystemId | URI của tài liệu XML ban đầu | Không | None |
| xslt | XSLT stylesheet cung cấp các chỉ thị chuyển đổi | Có | None |
| xsltSystemId | URI của tài liệu XSLT ban đầu | Không | None |
| result | Đối tượng result để chấp nhận kết quả của sự chuyển đổi | Không | Print to page |
| var | Biến được thiết lập tới tài liệu XML được chuyển đổi | Không | Print to page |
| scope | Phạm vi biến để trưng bày kết quả của sự chuyển đổi | Không | None |

## Ví dụ

Ví dụ sau minh họa cách sử dụng của thẻ <x:transform> trong JSTL:

Bạn theo dõi XSLT stylesheet có tên là style.xsl:

<?xml version="1.0"?>

<xsl:stylesheet xmlns:xsl=

"http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:output method="html" indent="yes"/>

<xsl:template match="/">

<html>

<body>

<xsl:apply-templates/>

</body>

</html>

</xsl:template>

<xsl:template match="books">

<table border="1" width="100%">

<xsl:for-each select="book">

<tr>

<td>

<i><xsl:value-of select="name"/></i>

</td>

<td>

<xsl:value-of select="author"/>

</td>

<td>

<xsl:value-of select="price"/>

</td>

</tr>

</xsl:for-each>

</table>

</xsl:template>

</xsl:stylesheet>

Bây giờ xem xét JSP file sau:

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<html>

<head>

<title>JSTL x:transform Tags</title>

</head>

<body>

<h3>Books Info:</h3>

<c:set var="xmltext">

<books>

<book>

<name>Padam History</name>

<author>ZARA</author>

<price>100</price>

</book>

<book>

<name>Great Mistry</name>

<author>NUHA</author>

<price>2000</price>

</book>

</books>

</c:set>

<c:import url="http://localhost:8080/style.xsl" var="xslt"/>

<x:transform xml="" xslt=""/>

</body>

</html>

Nó sẽ cho kết quả:

### Books Info:

|  |  |  |
| --- | --- | --- |
| *Padam History* | ZARA | 100 |
| *Great Mistry* | NUHA | 2000 |

* Thẻ <x:param>

Thẻ <x:param> trong JSTL được sử dụng cùng với thẻ transform để thiết lập một tham số trong XSLT stylesheet.

## Thuộc tính

Thẻ <x:param> trong JSTL có các thuộc tính sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Miêu tả** | **Cần thiết** | **Mặc định** |
| name | Tên tham số XSLT để thiết lập | Có | Body |
| value | Giá trị của tham số XSLT để thiết lập | Không | None |

## Ví dụ

Ví dụ sau minh họa cách sử dụng của thẻ <x:param> trong JSTL:

Bạn theo dõi XSLT stylesheet có tên là style.xsl. Chú ý cách sử dụng của thẻ <xsl:param...> và biến {$bgColor}:

<?xml version="1.0"?>

<xsl:stylesheet xmlns:xsl=

"http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:output method="html" indent="yes"/>

<xsl:param name="bgColor"/>

<xsl:template match="/">

<html>

<body>

<xsl:apply-templates/>

</body>

</html>

</xsl:template>

<xsl:template match="books">

<table border="1" width="50%" bgColor="{$bgColor}">

<xsl:for-each select="book">

<tr>

<td>

<i><xsl:value-of select="name"/></i>

</td>

<td>

<xsl:value-of select="author"/>

</td>

<td>

<xsl:value-of select="price"/>

</td>

</tr>

</xsl:for-each>

</table>

</xsl:template>

</xsl:stylesheet>

Bây giờ theo dõi JSP file sau, ở đây, chúng tôi định nghĩa giá trị của bgColor bởi sử dụng thẻ <x:param> bên trong thẻ <x:transform> trong JSTL:

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<html>

<head>

<title>JSTL x:transform Tags</title>

</head>

<body>

<h3>Books Info:</h3>

<c:set var="xmltext">

<books>

<book>

<name>Padam History</name>

<author>ZARA</author>

<price>100</price>

</book>

<book>

<name>Great Mistry</name>

<author>NUHA</author>

<price>2000</price>

</book>

</books>

</c:set>

<c:import url="http://localhost:8080/style.xsl" var="xslt"/>

<x:transform xml="" xslt="">

<x:param name="bgColor" value="grey"/>

</x:transform>

</body>

</html>

Nó sẽ cho kết quả sau:

### Books Info:

|  |  |  |
| --- | --- | --- |
| *Padam History* | ZARA | 100 |
| *Great Mistry* | NUHA | 2000 |

1. JTSL Functions Tags và các thẻ cơ bản

JSTL bao gồm một số hàm chuẩn, mà hầu hết là các hàm thao tác chuỗi phổ biến. Sau đây là cú pháp để bao nhóm JSTL Functions trong chương trình JSP của bạn:

z

<%@ taglib prefix="fn"

uri="http://java.sun.com/jsp/html/functions" %>

* Hàm fn:contains()

Hàm fn:contains() trong JSTL quyết định có hay không một chuỗi input chứa một chuỗi phụ đã xác định.

## Cú pháp:

Hàm fn:contains() trong JSTL có cú pháp như sau:

boolean contains(java.lang.String, java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:contains() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="theString" value="I am a test String"/>

<c:if test="${fn:contains(theString, 'test')}">

<p>Found test string<p>

</c:if>

<c:if test="${fn:contains(theString, 'TEST')}">

<p>Found TEST string<p>

</c:if>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Found test string

* Hàm fn:containsIgnoreCase()

Hàm fn:containsIgnoreCase() trong JSTL xác định có hay không một chuỗi input chứa một chuỗi phụ đã xác định. Trong khi thực hiện tìm kiếm, nó bỏ qua kiểu chữ.

## Cú pháp:

Hàm fn:containsIgnoreCase() trong JSTL có cú pháp như sau:

boolean containsIgnoreCase(java.lang.String, java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:containsIgnoreCase() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="theString" value="I am a test String"/>

<c:if test="${fn:containsIgnoreCase(theString, 'test')}">

<p>Found test string<p>

</c:if>

<c:if test="${fn:containsIgnoreCase(theString, 'TEST')}">

<p>Found TEST string<p>

</c:if>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Found test string

Found TEST string

* Hàm fn:endsWith()

Hàm fn:endsWith() trong JSTL xác định có hay không một chuỗi input kết thúc với một suffix đã xác định.

## Cú pháp:

Hàm fn:endsWith() trong JSTL có cú pháp như sau:

boolean endsWith(java.lang.String, java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:endsWith() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="theString" value="I am a test String 123"/>

<c:if test="${fn:endsWith(theString, '123')}">

<p>String ends with 123<p>

</c:if>

<c:if test="${fn:endsWith(theString, 'TEST')}">

<p>String ends with TEST<p>

</c:if>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

String ends with 123

* Hàm fn:escapeXml()

Hàm fn:escapeXml() trong JSTL tránh các ký tự mà có thể được phiên dịch như là XML Markup.

## Cú pháp:

Hàm fn:escapeXml() trong JSTL có cú pháp như sau:

java.lang.String escapeXml(java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:escapeXml() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="This <abc>is second String.</abc>"/>

<p>With escapeXml() Function:</p>

<p>string (1) : ${fn:escapeXml(string1)}</p>

<p>string (2) : ${fn:escapeXml(string2)}</p>

<p>Without escapeXml() Function:</p>

<p>string (1) : </p>

<p>string (2) : </p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

With escapeXml() Function:

string (1) : This is first String.

string (2) : This <abc>is second String.</abc>

Without escapeXml() Function:

string (1) : This is first String.

string (2) : This is second String.

* Hàm fn:indexOf()

Hàm fn:indexOf() trong JSTL trả về index bên trong một chuỗi của một chuỗi phụ đã xác định.

## Cú pháp:

Hàm fn:indexOf() trong JSTL có cú pháp như sau:

int indexOf(java.lang.String, java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:indexOf() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="This <abc>is second String.</abc>"/>

<p>Index (1) : ${fn:indexOf(string1, "first")}</p>

<p>Index (2) : ${fn:indexOf(string2, "second")}</p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Index (1) : 8

Index (2) : 13

* Hàm fn:join()

Hàm fn:join() trong JSTL trong JSTL nối chuỗi tất cả phần tử của một mảng thành một chuỗi với một Separator đã xác định.

## Cú pháp:

Hàm fn:join() trong JSTL có cú pháp như sau:

String join (java.lang.String[], java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:join() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="${fn:split(string1, ' ')}" />

<c:set var="string3" value="${fn:join(string2, '-')}" />

<p>Final String : </p>

</body>

</html>

**Ghi chú:** fn:split() trả về một mảng chia thành các phần tử khác nhau.

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Final String : This-is-first-String.

* Hàm fn:length()

Hàm fn:length() trong JSTL trả về độ dài chuỗi hoặc số item trong một collection.

## Cú pháp:

Hàm fn:length() trong JSTL có cú pháp như sau:

int length(java.lang.Object)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:length() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="This is second String." />

<p>Length of String (1) : ${fn:length(string1)}</p>

<p>Length of String (2) : ${fn:length(string2)}</p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Length of String (1) : 21

Length of String (2) : 22

* Hàm fn:replace()

Hàm fn:replace() trong JSTL thay thế tất cả sự xuất hiện của một chuỗi với một chuỗi khác.

## Cú pháp:

Hàm fn:replace() trong JSTL có cú pháp như sau:

boolean replace(java.lang.String, java.lang.String, java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:replace() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="${fn:replace(string1,

'first', 'second')}" />

<p>Final String : </p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Final String : This is second String.

* Hàm fn:split()

Hàm fn:split() trong JSTL chia một chuỗi thành một mảng các chuỗi phụ dựa trên một chuỗi delimiter.

## Cú pháp:

Hàm fn:split() trong JSTL có cú pháp như sau:

java.lang.String[] split(java.lang.String, java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:split() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="${fn:split(string1, ' ')}" />

<c:set var="string3" value="${fn:join(string2, '-')}" />

<p>String (3) : </p>

<c:set var="string4" value="${fn:split(string3, '-')}" />

<c:set var="string5" value="${fn:join(string4, ' ')}" />

<p>String (5) : </p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

String (3) : This-is-first-String.

String (5) : This is first String.

* Hàm fn:startsWith()

Hàm fn:startsWith() trong JSTL quyết định có hay không một chuỗi input bắt đầu với một chuỗi phụ đã xác định.

## Cú pháp:

Hàm fn:startsWith() trong JSTL có cú pháp như sau:

boolean startsWith(java.lang.String, java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:startsWith() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string" value="Second: This is first String."/>

<c:if test="${fn:startsWith(string, 'First')}">

<p>String starts with First</p>

</c:if>

<br />

<c:if test="${fn:startsWith(string, 'Second')}">

<p>String starts with Second</p>

</c:if>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

String starts with Second

* Hàm fn:substring()

Hàm fn:substring() trong JSTL trả về một tập con của một chuỗi được xác định bởi chỉ mục bắt đầu và chỉ mục kết thúc.

## Cú pháp:

Hàm fn:substring() trong JSTL có cú pháp như sau:

java.lang.String substring(java.lang.String, int, int)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:substring() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="${fn:substring(string1, 5, 15)}" />

<p>Final sub string : </p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Final sub string : is first S

* Hàm fn:substringAfter()

Hàm fn:substringAfter() trong JSTL trả về một phần của chuỗi mà ở sau một chuỗi phụ đã xác định.

## Cú pháp:

Hàm fn:substringAfter() trong JSTL có cú pháp như sau:

java.lang.String substringAfter(java.lang.String, java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:substringAfter() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="${fn:substringAfter(string1, 'is')}" />

<p>Final sub string : </p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Final sub string : is first String.

* Hàm fn:substringBefore()

Hàm fn:substringBefore() trong JSTL trả về một phần của chuỗi mà ở trước một chuỗi phụ đã xác định.

## Cú pháp:

Hàm fn:substringBefore() trong JSTL có cú pháp như sau:

java.lang.String substringBefore(java.lang.String,

java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:substringBefore() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="${fn:substringBefore(string1,

'first')}" />

<p>Final sub string : </p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Final sub string : This is

* Hàm fn:toLowerCase()

Hàm fn:toLowerCase() trong JSTL biến đổi tất cả ký tự của một chuỗi thành chữ thường.

## Cú pháp:

Hàm fn:toLowerCase() trong JSTL có cú pháp như sau:

java.lang.String toLowerCase(java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:toLowerCase() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="${fn:toLowerCase(string1)}" />

<p>Final string : </p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Final string : this is first string.

* Hàm fn:toUpperCase()

Hàm fn:toUpperCase() trong JSTL biến đổi tất cả ký tự của một chuỗi thành chữ hoa.

## Cú pháp:

Hàm fn:toUpperCase() trong JSTL có cú pháp như sau:

java.lang.String tolowercase(java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:toUpperCase() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String."/>

<c:set var="string2" value="${fn:toUpperCase(string1)}" />

<p>Final string : </p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

Final string : THIS IS FIRST STRING.

* Hàm fn:trim()

Hàm fn:trim() trong JSTL gỡ bỏ các khoảng trống trăng từ cả hai đầu chuỗi: phần đầu và kết thúc.

## Cú pháp:

Hàm fn:trim() trong JSTL có cú pháp như sau:

java.lang.String trim(java.lang.String)

## Ví dụ

Ví dụ sau giải thích tính năng của hàm fn:trim() trong JSTL:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>

<c:set var="string1" value="This is first String "/>

<p>String (1) Length : ${fn:length(string1)}</p>

<c:set var="string2" value="${fn:trim(string1)}" />

<p>String (2) Length : ${fn:length(string2)}</p>

<p>Final string : ${string2}</p>

</body>

</html>

**Chú ý:** Ký tự $ và { là viết liền nhau.

Nó sẽ cho kết quả sau:

String (1) Length : 29

String (2) Length : 20

Final string : This is first String

1. Sử dụng JTSL trong webapp
2. Tài liệu tham khảo

* <http://vietjack.com/jsp/jsp_standard_tag_library.jsp>
* <http://o7planning.org/web/fe/default/vi/document/1264541/huong-dan-su-dung-jsp-standard-tag-library-jstl>
* <http://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm>
* Jstl api 1.2.5