[Thực hành] ứng dụng blog quốc tế

Mục tiêu

Luyện tập sử dụng spring message bundle, luyện tập triển khai i18n và l10n.

Mô tả

Với tài nguyên là một dự án Spring MVC đang chạy, triển khai internationalize.

Có thể sử dụng mã stock tại nhánh master của project này.

Sau khi hoàn thành, học viên sẽ:

- Cấu hình được Spring sử dụng messageSource
- Sử dụng được message source
- Sử dụng được LocaleChangeInterceptor để hỗ trợ localization
- Sử dụng được SessionLocaleResolver để hỗ trợ localization

Hướng dẫn

Chuẩn bị

Hướng dẫn ở đây dựa trên mã ở nhánh master của project này.

Tạo và checkout sang nhánh `my_internationalize`.

Sử dụng `messageSource`

Trong thư mục `resources`, tạo file `message.properties`, điền các message cần đặt tên (để sử dụng trong view mà không cần hard-coded) vào, ví dụ như sau:

```
login.title=Login to Dashboard
login.heading=Login to Dashboard
login.username_label=Username:
login.password_label=Password:
login.submit_label=Login

dashboard.title=Greeting
dashboard.heading=Dashboard
dashboard.greeting=Hello
```

Khai báo cấu hình bean `messageSource`, ở đây là mã mẫu khi khai báo bằng annotation trong file cấu hình bằng java.

```
@Bean
public MessageSource messageSource() {
ResourceBundleMessageSource messageSource = new ResourceBundleMessageSource();
messageSource.setBasename("message");
return messageSource;
}
```

Trong đó, giá trị `"message"` chính là tên file trong file `message.properties`.

Nhớ thêm các imports cần thiết.

Sau đó, ở template, gọi tới tên của các message trong file `message.properties` thay vì sử dụng hard-coded, sau đây là mã tham khảo khi template là html và được resolve bằng thư viện thymeleaf, chú ý cú pháp `#{messagePath}`:

```
<!doctype html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title th:text="#{login.title}"></title>
</head>
<body>
<h1 th:text="#{login.heading}"></h1>
Language: <a href="?lang=en">English</a> | <a href="?lang=haw">Hawaii</a>
<form th:action="@{/login}" th:object="${credential}" method="post">
  <table>
    <tr>
      <td th:text="#{login.username_label}"></td>
      <td><input type="text" th:field="*{username}"/></td>
    </tr>
    <tr>
      <td th:text="#{login.password_label}"></td>
      <td><input type="password" th:field="*{password}"/></td>
    </tr>
    <tr>
      <td></td>
      <td><input type="submit" th:value="#{login.submit_label}"></td>
    </tr>
  </table>
</form>
</body>
</html>
```

Thực hiện build và deploy lại project để xem kết quả.

Bản địa hóa

Tạo thêm file `message_haw.properties` trong thư mục `resource` với các tên message giống với file `message_haw.properties` nhưng nội dung đã được bản địa hóa cho ngôn ngữ Hawaii:

```
login.title=E komo i ka papa uila
login.heading=E komo i ka papa uila
login.username_label=Inoa inoa:
login.password_label=Hua huna:
login.submit_label=Komo mai

dashboard.title=Aloha
dashboard.heading=‘O ka papa
dashboard.greeting=Aloha
```

Trong file cấu hình Spring application, cấu hình để bổ sung LocaleChangeInterceptor.

```
@Override
public void addInterceptors(InterceptorRegistry registry) {
LocaleChangeInterceptor interceptor = new LocaleChangeInterceptor();
interceptor.setParamName("lang");
registry.addInterceptor(interceptor);
}
```

Interceptor này sẽ tự động phân tích tham số `lang` đi kèm các request để xác định bản địa hiện tại.

Tiếp theo, bổ sung bean localeResolver để sử dụng thông tin về bản địa và sử dụng message bundle tương ứng:

```
@Bean
public LocaleResolver localeResolver() {
SessionLocaleResolver localeResolver = new SessionLocaleResolver();
localeResolver.setDefaultLocale(new Locale("en"));
return localeResolver;
}
```

Ở đây, ta đã khai báo một localeResolver lấy thông tin về bản địa từ session của người dùng, đồng thời dự phòng sử dụng ngôn ngữ là “en”.

Bước tiếp theo, trong template, tạo một thẻ anchor đơn giản để gửi request với tham số “lang”:

```
<!doctype html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title th:text="#{login.title}"></title>
</head>
<body>
<h1 th:text="#{login.heading}"></h1>
Language: <a href="?lang=en">English</a> | <a href="?lang=haw">Hawaii</a>
<form th:action="@{/login}" th:object="${credential}" method="post">
  <table>
    <tr>
      <td th:text="#{login.username_label}"></td>
      <td><input type="text" th:field="*{username}"/></td>
    </tr>
    <tr>
      <td th:text="#{login.password_label}"></td>
      <td><input type="password" th:field="*{password}"/></td>
    </tr>
    <tr>
      <td></td>
      <td><input type="submit" th:value="#{login.submit_label}"></td>
    </tr>
  </table>
</form>
</body>
</html>
```

Deploy lại project và kiểm tra kết quả.

Bản địa hóa dựa trên session

Hãy tiếp tục bản địa hóa trang dashboard hay một trang khác trong project:

```
<!doctype html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title th:text="#{dashboard.title}"></title>
</head>
<body>
<h1 th:text="#{dashboard.heading}"></h1>
<span th:text="#{dashboard.greeting}"></span> <span th:text="${user.userName}"></span>
</body>
</html>
```

Sau đó kiểm tra lại để thấy rằng việc bản địa hóa vẫn tiếp tục được thực hiện mà không cần tới tham số `lang` trên tất cả các request. Đó là vì ở đây đang sử dụng `LocaleResolver` là một `SessionLocaleResolver`.

Mã hoàn chỉnh của bài thực hành này có thể được tham khảo tại nhánh internationalize.