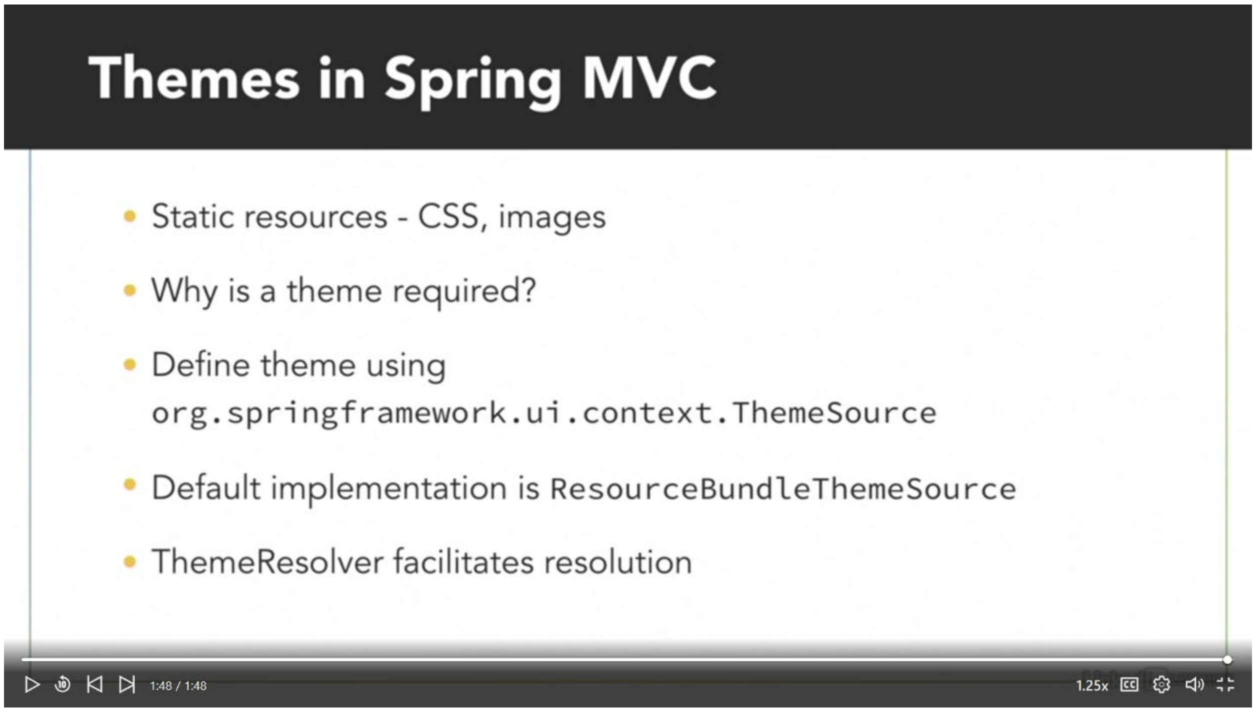
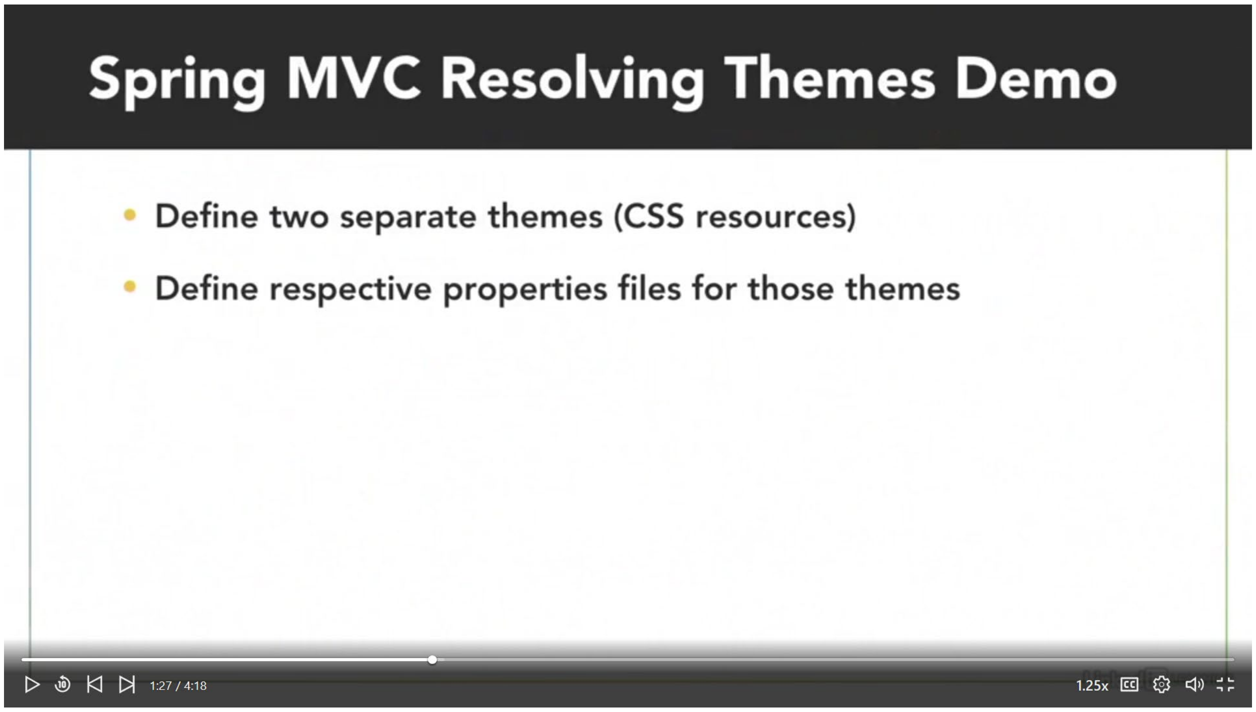
Themes in Spring MVC



Themes in Spring MVC

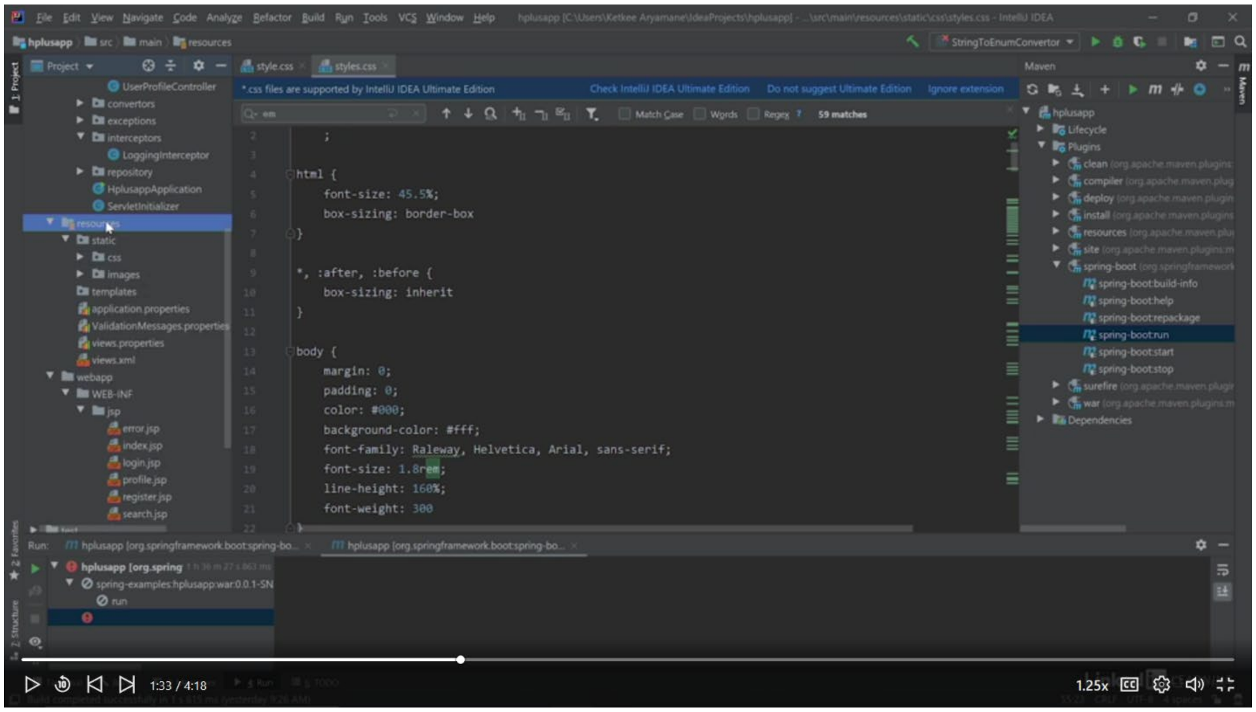
- Static resources - CSS, images
- Why is a theme required?
- Define theme using `org.springframework.ui.context.ThemeSource`
- Default implementation is `ResourceBundleThemeSource`
- ThemeResolver facilitates resolution



Spring MVC Resolving Themes Demo

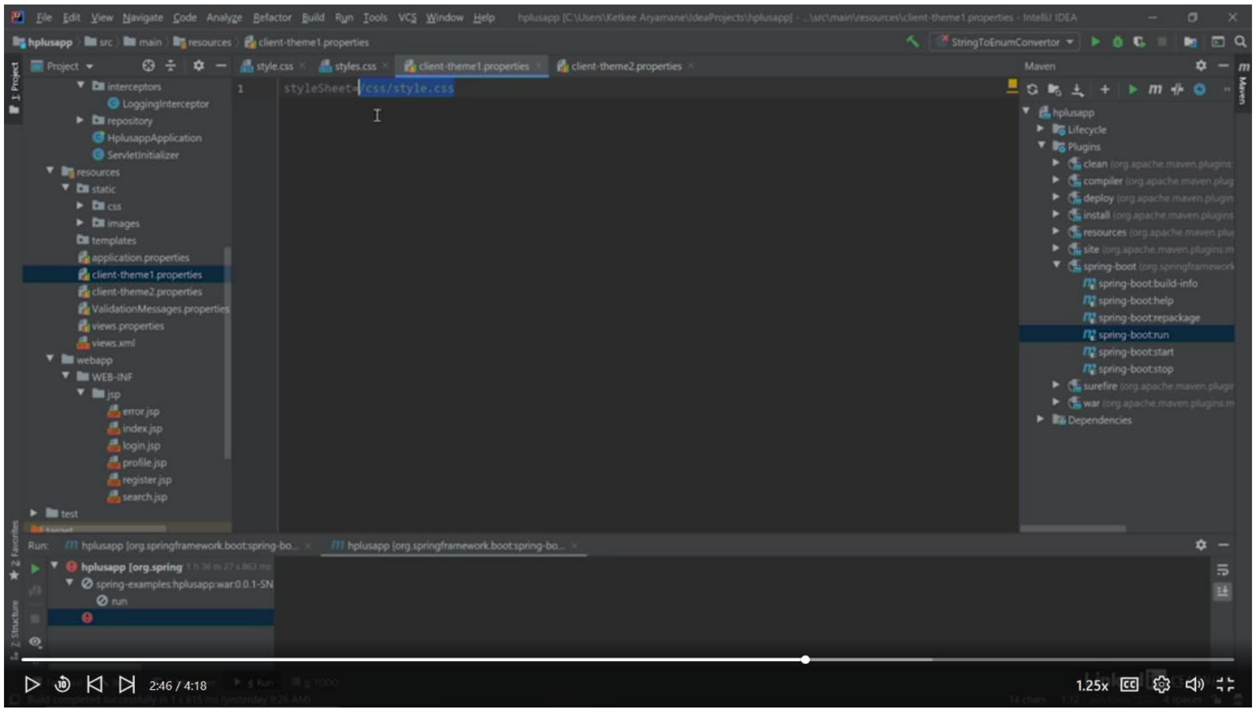
- **Define two separate themes (CSS resources)**
- **Define respective properties files for those themes**

Different style.css namely style.css and styles.css

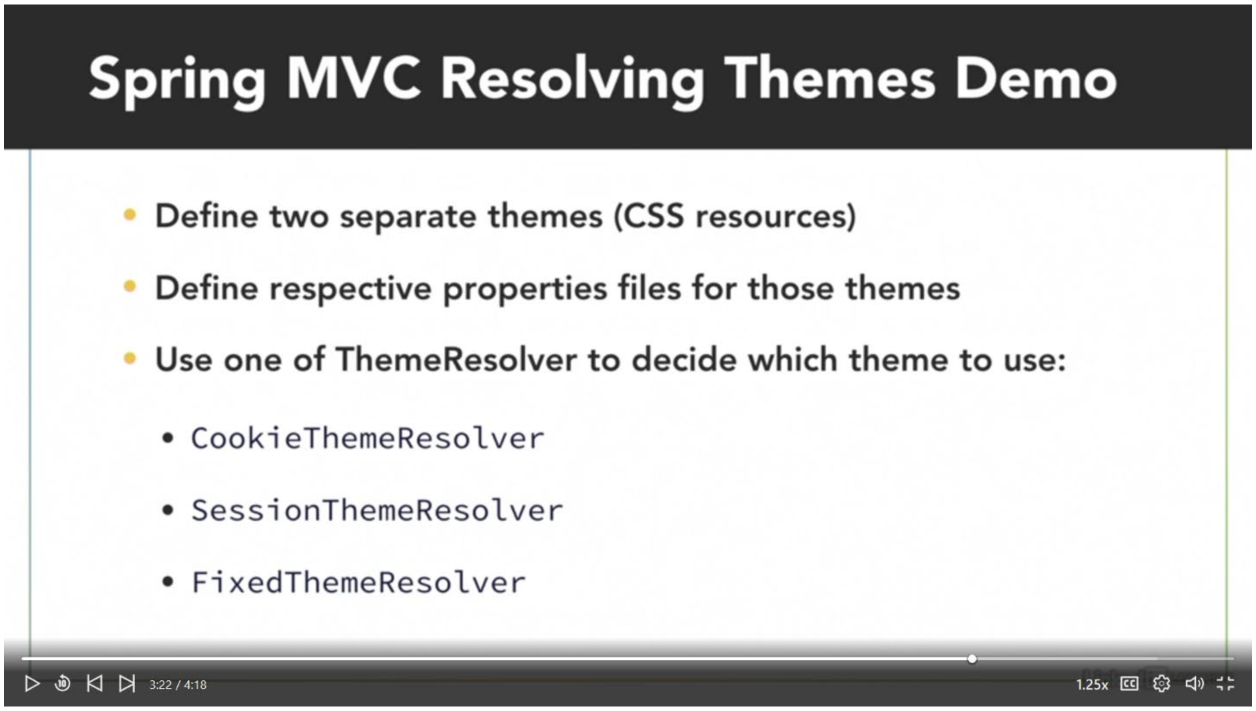


```
    ;

html {
    font-size: 45.5%;
    box-sizing: border-box
}

*, :after, :before {
    box-sizing: inherit
}

body {
    margin: 0;
    padding: 0;
    color: #000;
    background-color: #fff;
    font-family: Raleway, Helvetica, Arial, sans-serif;
    font-size: 1.8rem;
    line-height: 160%;
    font-weight: 300
```

Create two different properties file client-theme1 and client-theme2 with stylesheet pointing to the two different css style sheet.



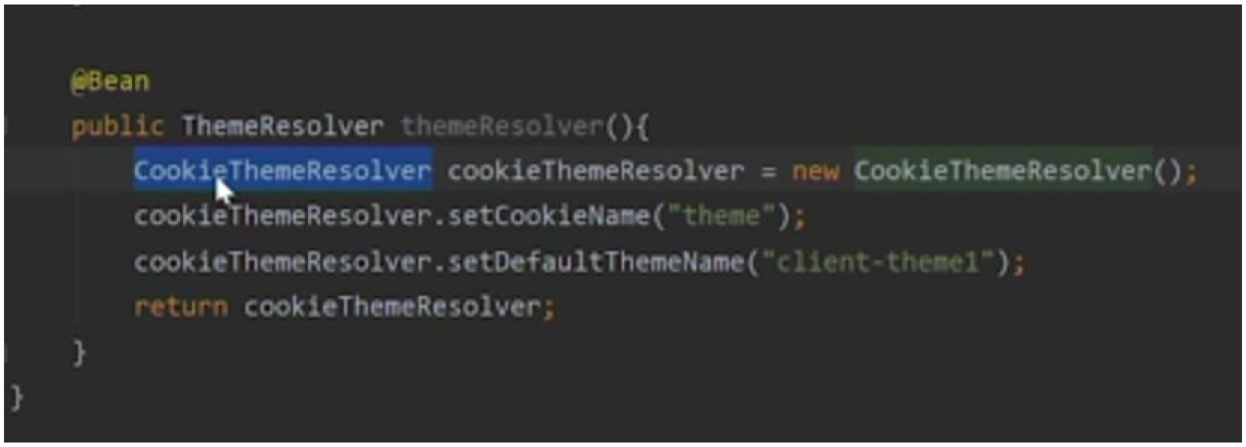
```
styleSheet=/css/style.css
```



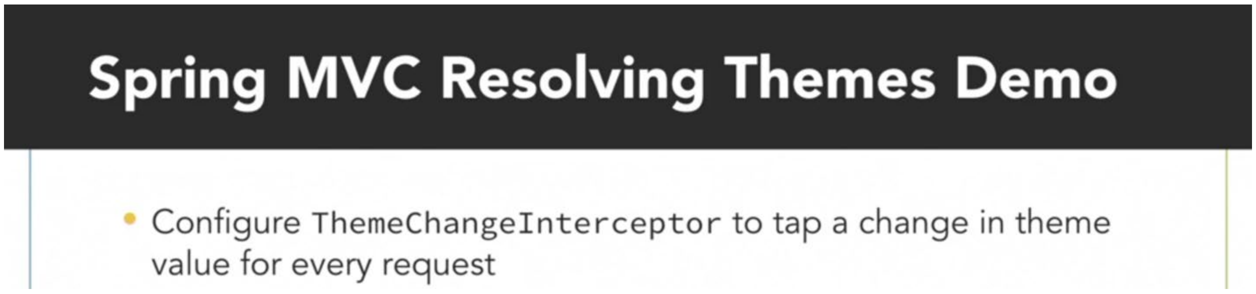
Spring MVC Resolving Themes Demo

- Define two separate themes (CSS resources)
- Define respective properties files for those themes
- Use one of ThemeResolver to decide which theme to use:
  - `CookieThemeResolver`
  - `SessionThemeResolver`
  - `FixedThemeResolver`

We are going to use CookieThemeResolver. Goto ApplicationConfig class and create a Bean that will give a ThemeResolver



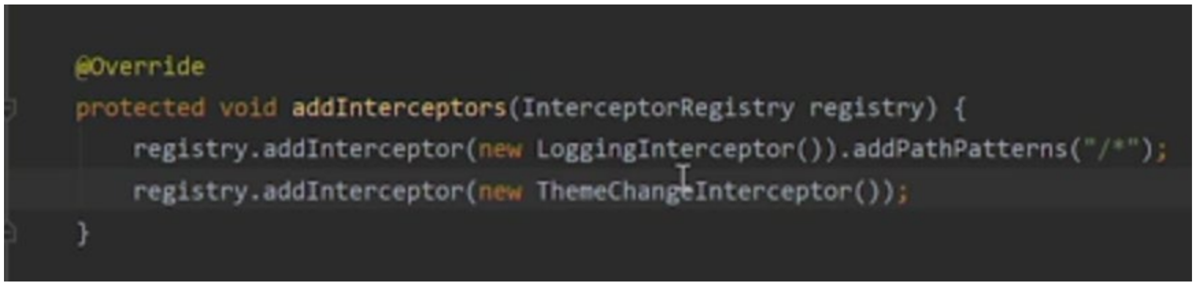
```
    @Bean
    public ThemeResolver themeResolver(){
        CookieThemeResolver cookieThemeResolver = new CookieThemeResolver();
        cookieThemeResolver.setCookieName("theme");
        cookieThemeResolver.setDefaultThemeName("client-theme1");
        return cookieThemeResolver;
    }
}
```



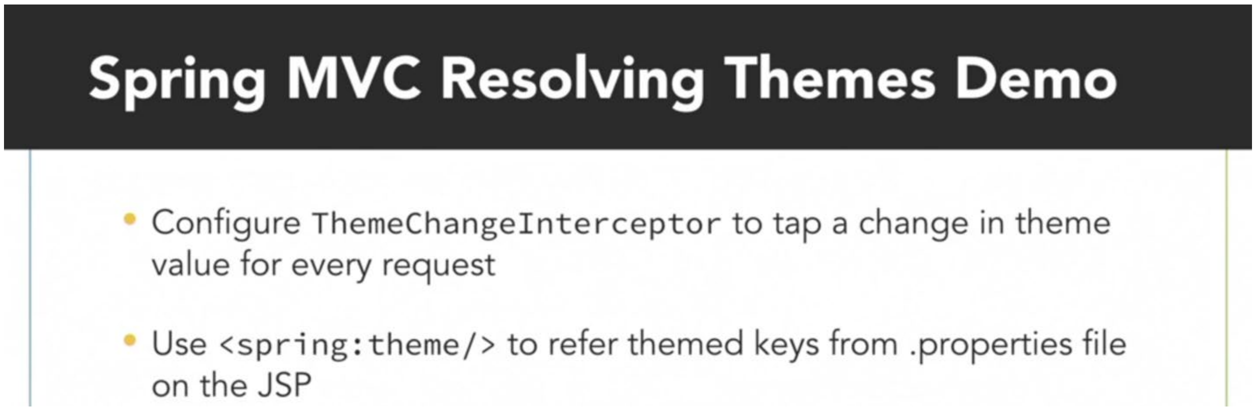
Spring MVC Resolving Themes Demo

- Configure `ThemeChangeInterceptor` to tap a change in theme value for every request

Register a ThemeChanngeInterceptor inside ApplicationConfig class an implementation provided by Spring itself



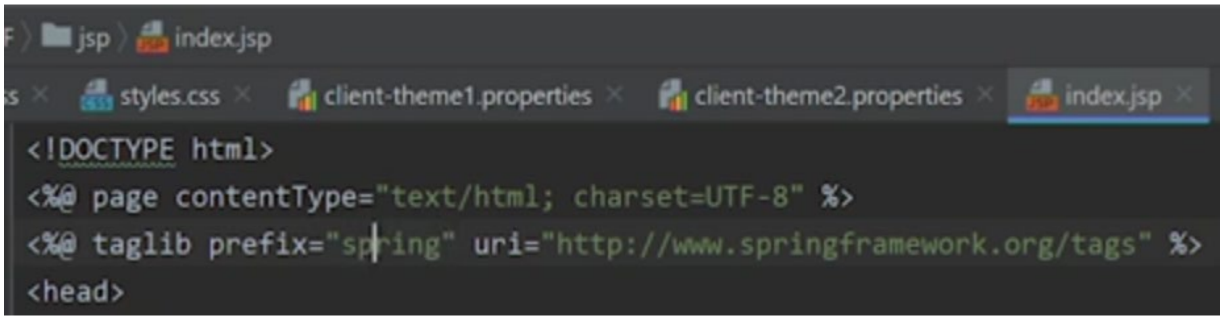
```
@Override
protected void addInterceptors(InterceptorRegistry registry) {
    registry.addInterceptor(new LoggingInterceptor()).addPathPatterns("/*");
    registry.addInterceptor(new ThemeChangeInterceptor());
}
```



Spring MVC Resolving Themes Demo

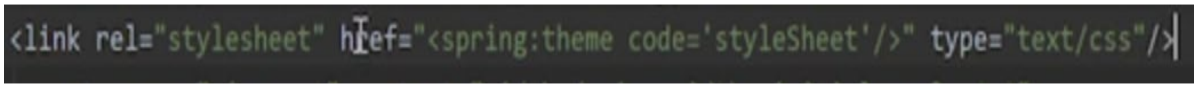
- Configure `ThemeChangeInterceptor` to tap a change in theme value for every request
- Use `<spring:theme/>` to refer themed keys from .properties file on the JSP

Add the spring tag library in index.jsp

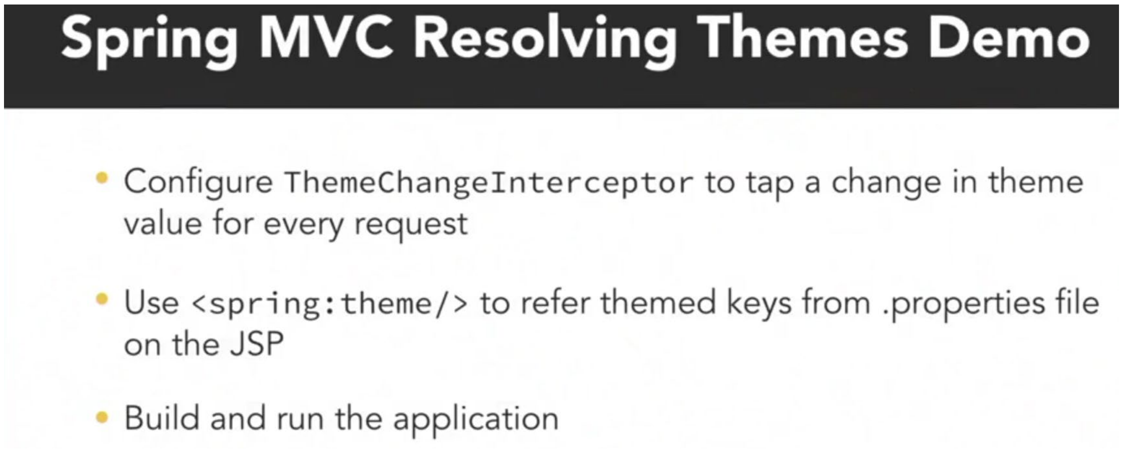


```
<!DOCTYPE html>
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<head>
```

Change the link rel tag to respond the change theme



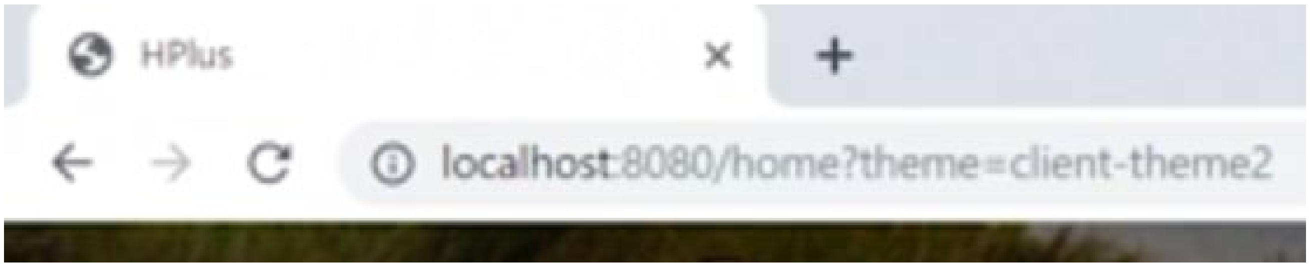
```
<link rel="stylesheet" href="<spring:theme code='styleSheet'/>" type="text/css"/>
```



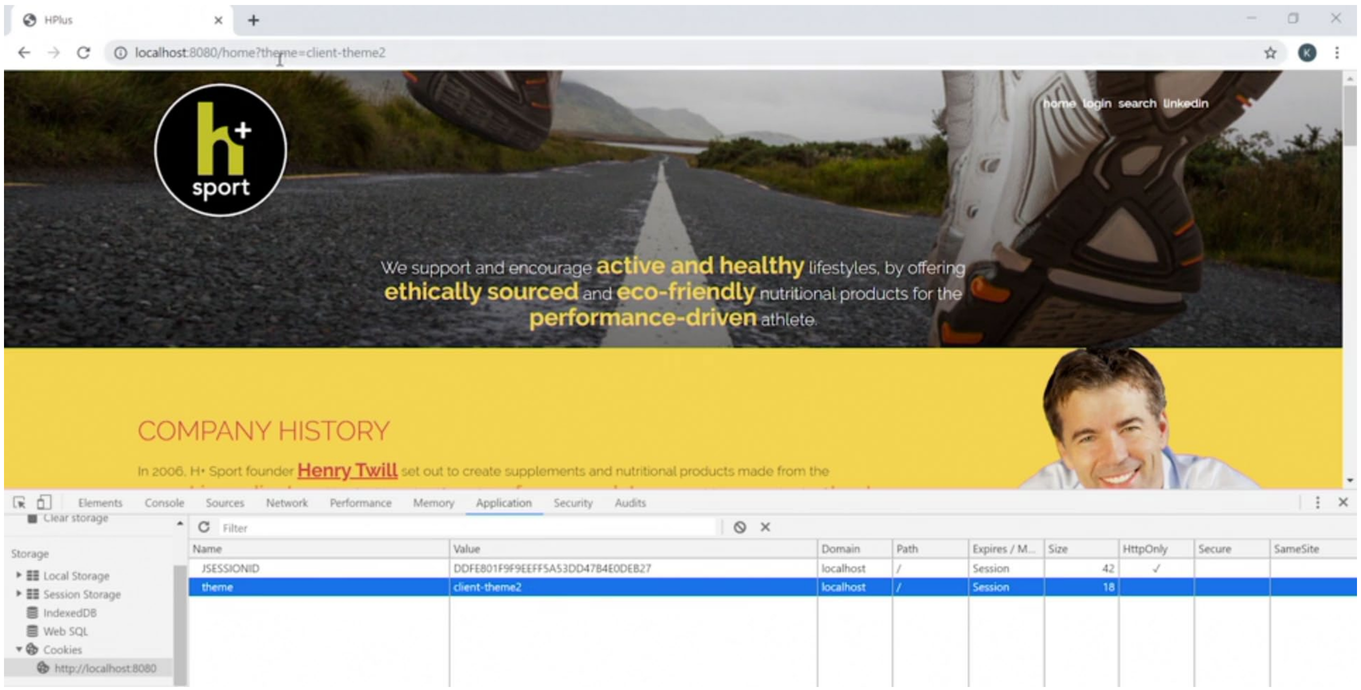
Spring MVC Resolving Themes Demo

- Configure `ThemeChangeInterceptor` to tap a change in theme value for every request
- Use `<spring:theme/>` to refer themed keys from .properties file on the JSP
- Build and run the application

Change the url query pattern as



```
localhost:8080/home?theme=client-theme2
```



```
localhost:8080/home?theme=client-theme2
```

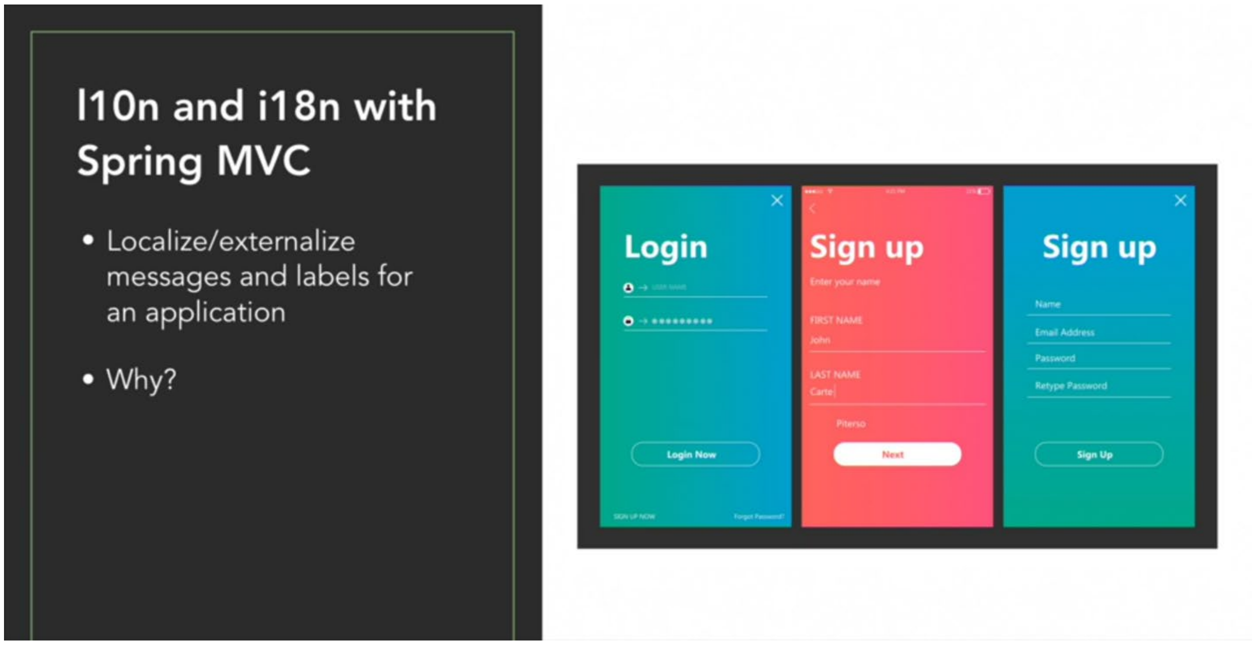
We support and encourage **active and healthy** lifestyles, by offering **ethically sourced** and **eco-friendly** nutritional products for the **performance-driven** athlete.

COMPANY HISTORY

In 2006, H+ Sport founder **Henry Twill** set out to create supplements and nutritional products made from the

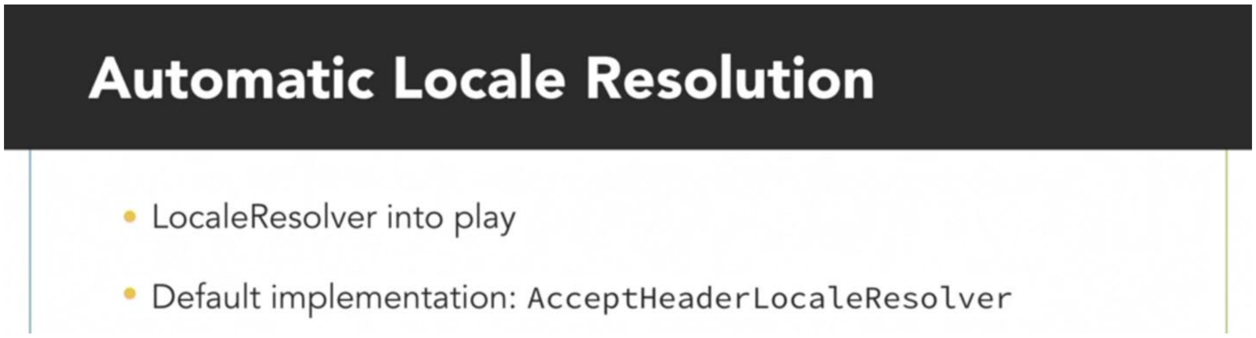
Name	Value	Domain	Path	Expires / M...	Size	HttpOnly	Secure	SameSite
JSESSIONID	DDFE801F9F9EEFF5A53DD47B4E0DEB27	localhost	/	Session	42	✓		
theme	client-theme2	localhost	/	Session	18			

L10n and i18n with Spring MVC



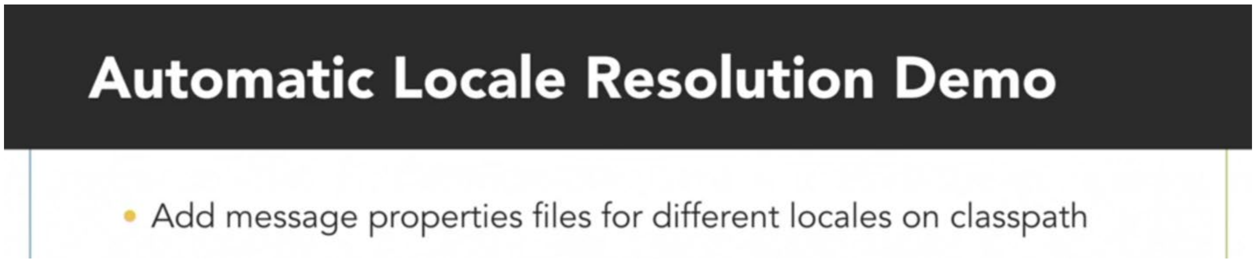
l10n and i18n with Spring MVC

- Localize/externalize messages and labels for an application
- Why?



Automatic Locale Resolution

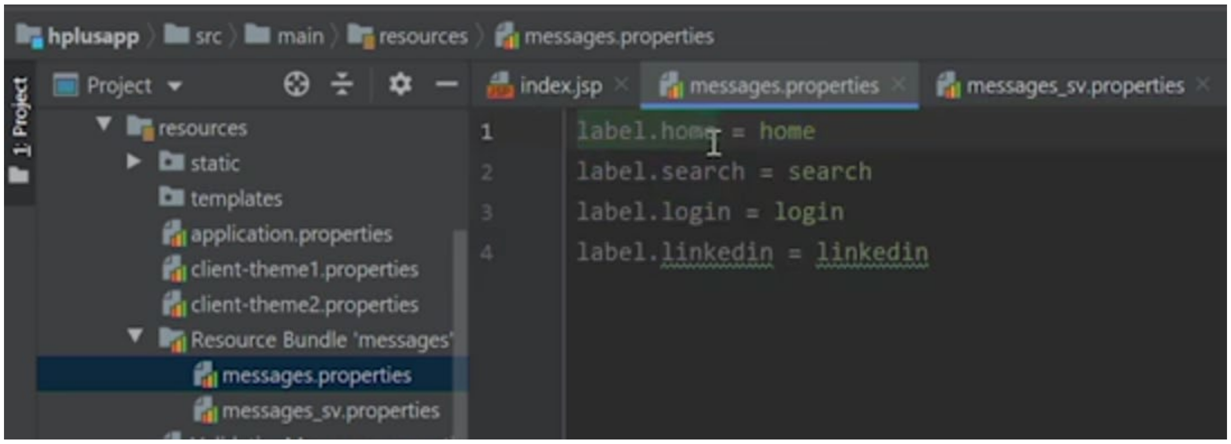
- LocaleResolver into play
- Default implementation: `AcceptHeaderLocaleResolver`



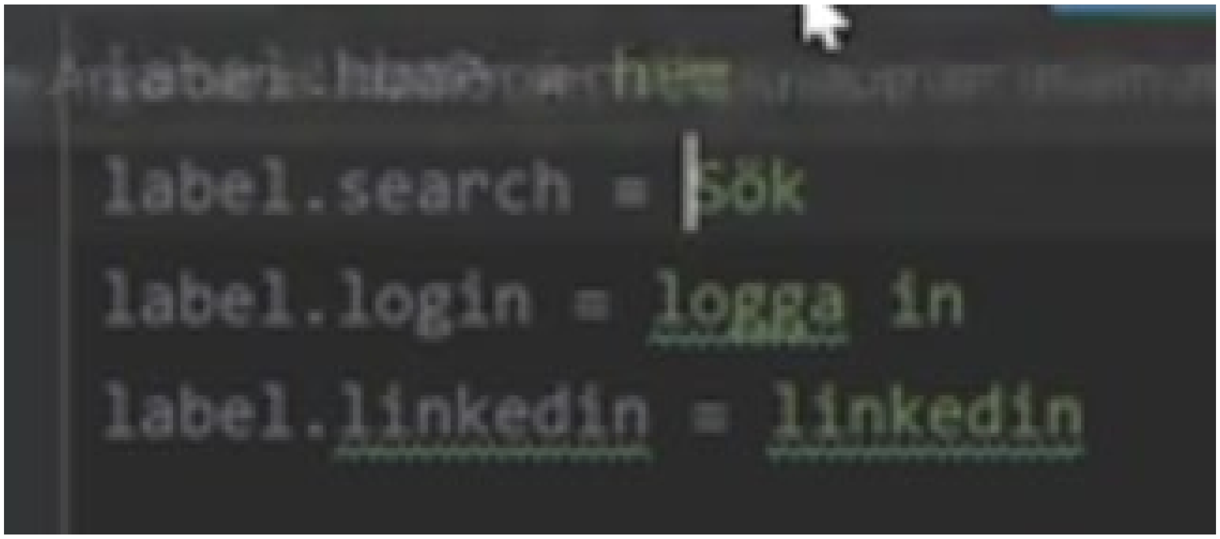
Automatic Locale Resolution Demo

- Add message properties files for different locales on classpath

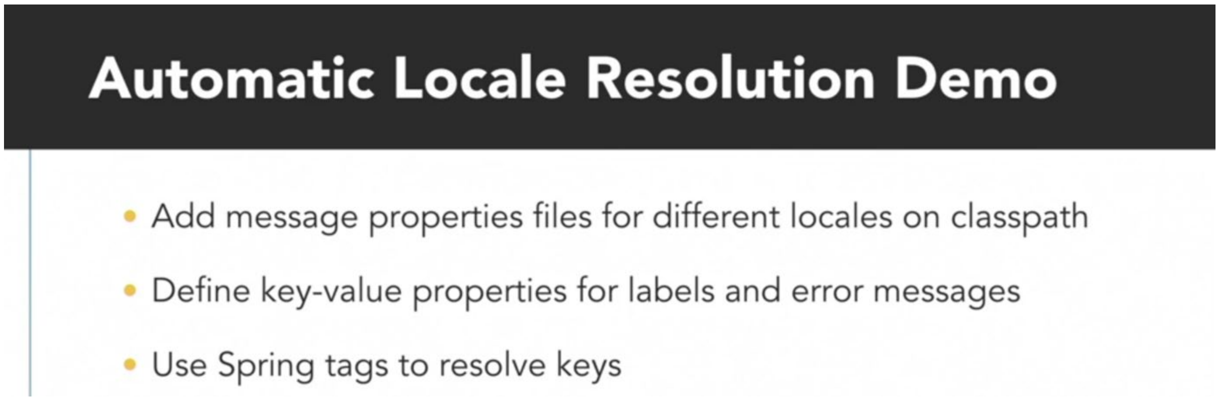
Adding two properties file for messaging one messages.properties for English language and one messages_sv.properties for Swidish



```
label.home = home
label.search = search
label.login = login
label.linkedin = linkedin
```



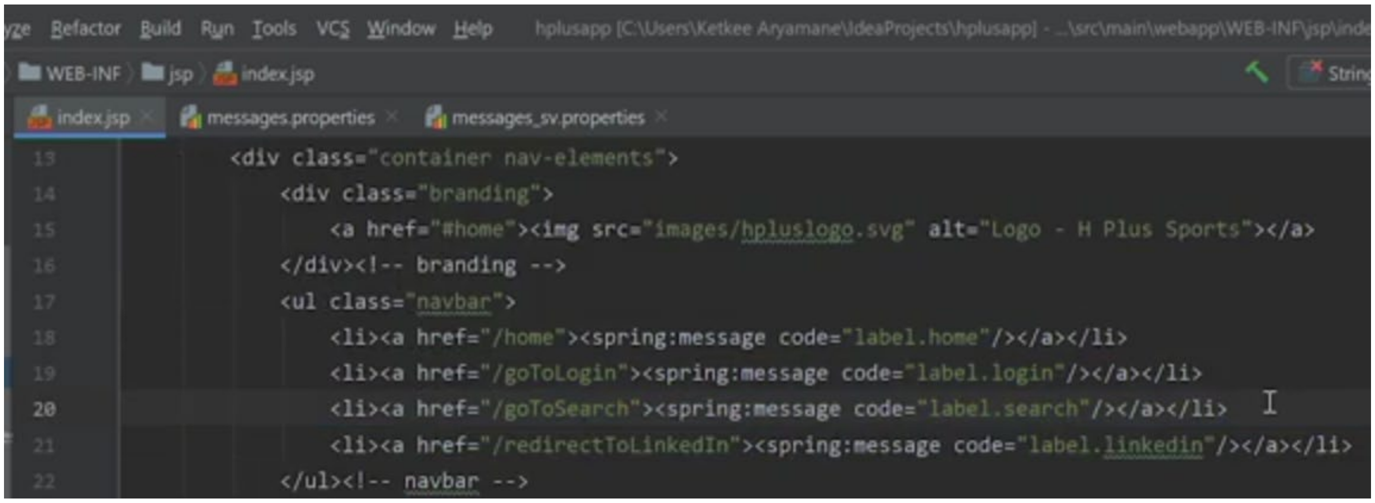
```
label.search = Sök
label.login = logga in
label.linkedin = linkedin
```



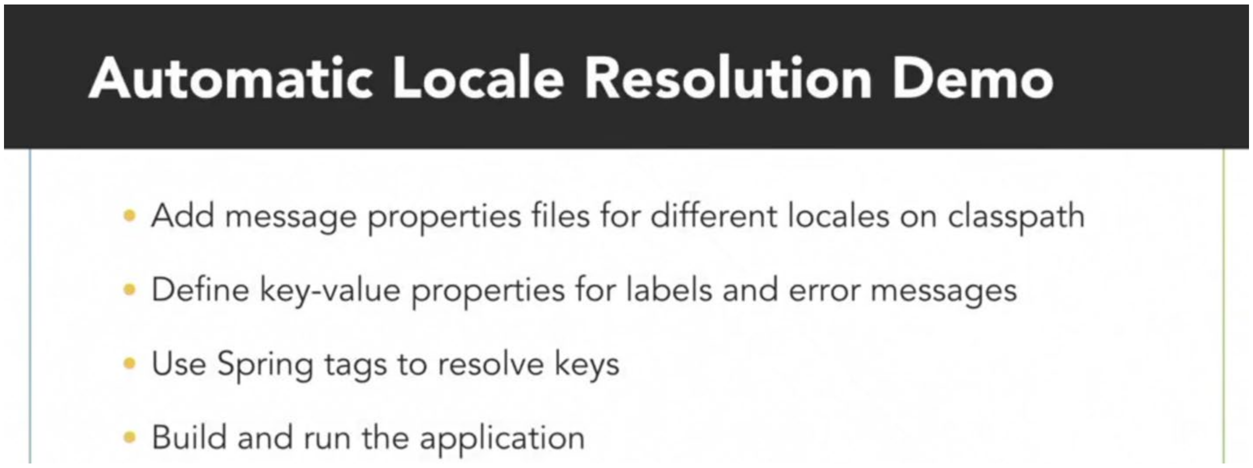
Automatic Locale Resolution Demo

- Add message properties files for different locales on classpath
- Define key-value properties for labels and error messages
- Use Spring tags to resolve keys

Adding Spring:message tag for internationalization in index.jsp


yze Refactor Build Run Tools VCS Window Help hplusapp [C:\Users\Ketkee Aryamane\IdeaProjects\hplusapp] - ...\src\main\webapp\WEB-INF\jsp\inde
WEB-INF jsp index.jsp
index.jsp messages.properties messages_sv.properties

```
            <div class="container nav-elements">
                <div class="branding">
                    <a href="#home"><img src="images/hpluslogo.svg" alt="Logo - H Plus Sports"></a>
                </div><!-- branding -->
                <ul class="navbar">
                    <li><a href="/home"><spring:message code="label.home"/></a></li>
                    <li><a href="/goToLogin"><spring:message code="label.login"/></a></li>
                    <li><a href="/goToSearch"><spring:message code="label.search"/></a></li>
                    <li><a href="/redirectToLinkedIn"><spring:message code="label.linkedin"/></a></li>
                </ul><!-- navbar -->
```



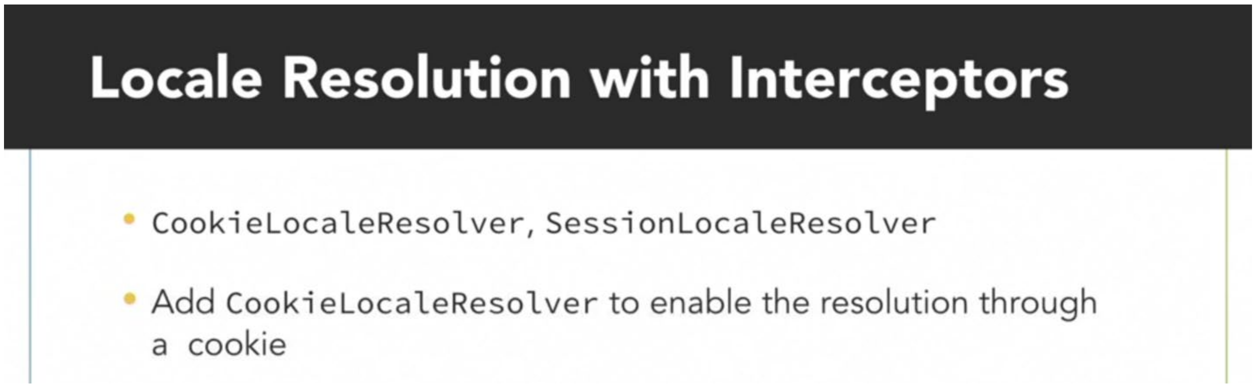
Automatic Locale Resolution Demo

- Add message properties files for different locales on classpath
- Define key-value properties for labels and error messages
- Use Spring tags to resolve keys
- Build and run the application

Now in browser add Swedish language by going to the setting of the browser and access the link

http://localhost:8080/home?theme=client-theme2

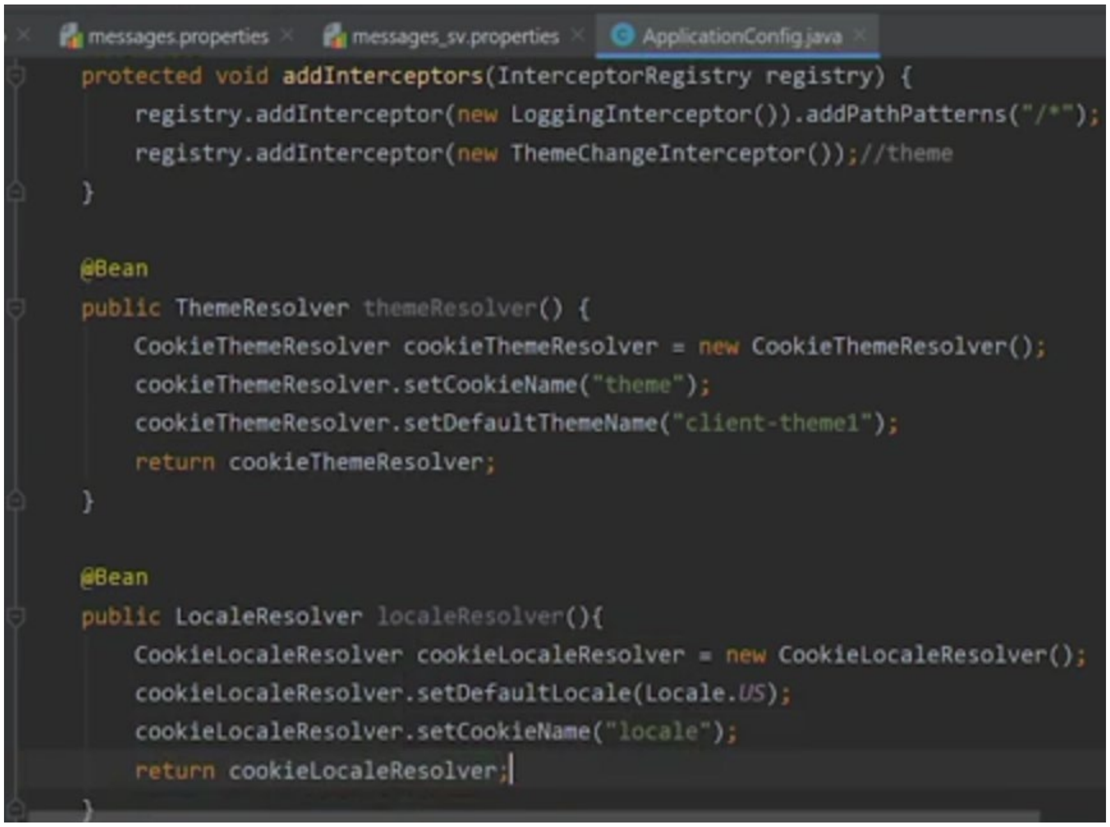
http://localhost:8080/home



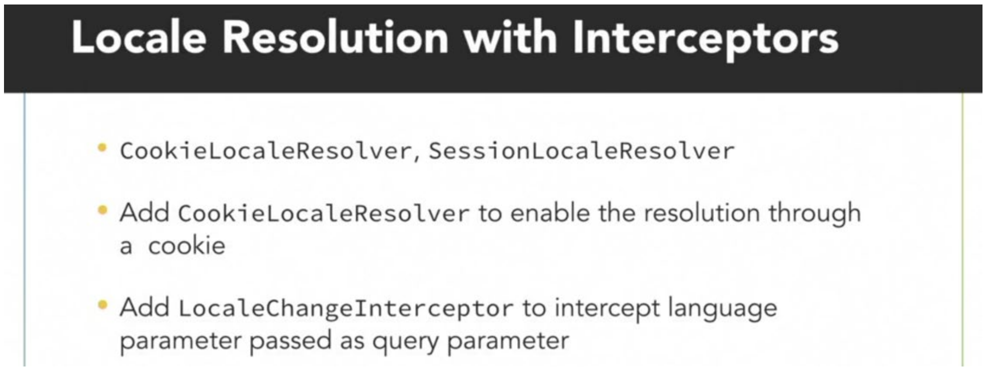
Locale Resolution with Interceptors

- `CookieLocaleResolver`, `SessionLocaleResolver`
- Add `CookieLocaleResolver` to enable the resolution through a cookie

Create a LocaleResolver Bean in ApplicationConfig class



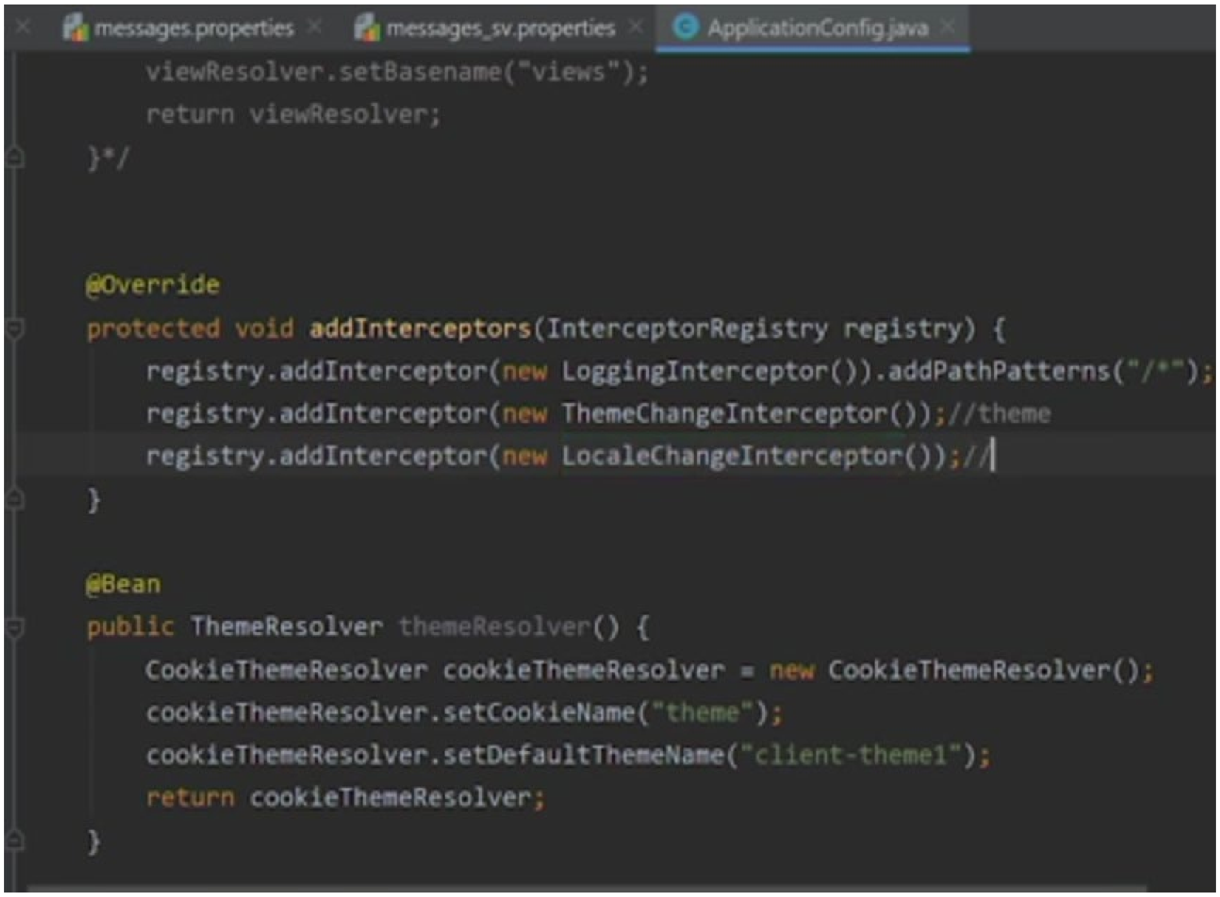
```java
protected void addInterceptors(InterceptorRegistry registry) {
    registry.addInterceptor(new LoggingInterceptor()).addPathPatterns("/*");
    registry.addInterceptor(new ThemeChangeInterceptor());//theme
}

@Bean
public ThemeResolver themeResolver() {
    CookieThemeResolver cookieThemeResolver = new CookieThemeResolver();
    cookieThemeResolver.setCookieName("theme");
    cookieThemeResolver.setDefaultThemeName("client-theme1");
    return cookieThemeResolver;
}

@Bean
public LocaleResolver localeResolver(){
    CookieLocaleResolver cookieLocaleResolver = new CookieLocaleResolver();
    cookieLocaleResolver.setDefaultLocale(Locale.US);
    cookieLocaleResolver.setCookieName("locale");
    return cookieLocaleResolver;
}
```



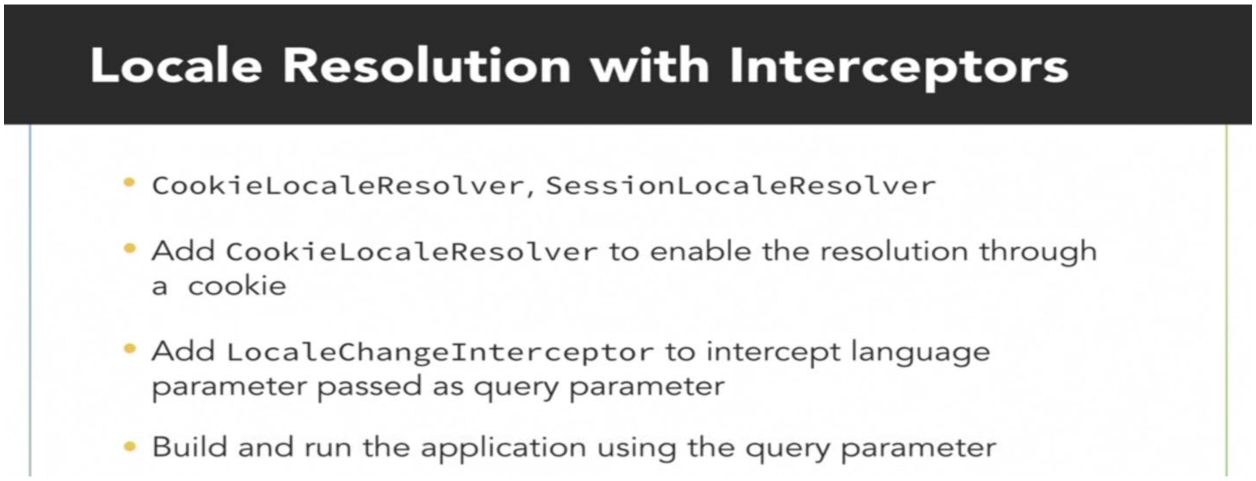
Locale Resolution with Interceptors

- `CookieLocaleResolver`, `SessionLocaleResolver`
- Add `CookieLocaleResolver` to enable the resolution through a cookie
- Add `LocaleChangeInterceptor` to intercept language parameter passed as query parameter

Register the interceptor via addInterceptor method in ApplicationConfig class



```java
        viewResolver.setBasename("views");
        return viewResolver;
    }*/


    @Override
    protected void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new LoggingInterceptor()).addPathPatterns("/*");
        registry.addInterceptor(new ThemeChangeInterceptor());//theme
        registry.addInterceptor(new LocaleChangeInterceptor());//
    }

    @Bean
    public ThemeResolver themeResolver() {
        CookieThemeResolver cookieThemeResolver = new CookieThemeResolver();
        cookieThemeResolver.setCookieName("theme");
        cookieThemeResolver.setDefaultThemeName("client-theme1");
        return cookieThemeResolver;
    }
```



Locale Resolution with Interceptors

- CookieLocaleResolver, SessionLocaleResolver
- Add CookieLocaleResolver to enable the resolution through a cookie
- Add LocaleChangeInterceptor to intercept language parameter passed as query parameter
- Build and run the application using the query parameter

Now in browser add Swedish language by going to the setting of the browser and access the link

http://localhost:8080/home?locale=sv