thymeleaf-ee (0.0.2)

Maksim Kostromin

Version 0.0.2, 2019-09-18 21:13:47 UTC

Table of Contents

Travis CI status: [Build Status]

Chapter 1. Introduction

This repo contains example of Thymeleaf and Java EE integration

Read GitHub pages project reference documentation

generated by generator-jvm yeoman generator (java-ee-thymeleaf)

Chapter 2. Implementation

2.1. create Java EE project

I'm using my yo `generator-jvm` *generator package (type:* `java-ee`*, name* `thymeleaf-ee`*):*

```
npm i -g yo generator generator-jvm
yo jvm -t java-ee -n thymeleaf-ee
```

beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
       bean-discovery-mode="all">
</beans>
```

2.2. add thymeleaf and mvc 1.0 spec dependencies

maven

```
<dependencies>
  <dependency>
    <groupId>javax.mvc</groupId>
    <artifactId>javax.mvc-api</artifactId>
    <version>${javax.mvc-api.version}</version>
  </dependency>
  <dependency>
    <groupId>org.mvc-spec.ozark</groupId>
    <artifactId>ozark-resteasy</artifactId>
    <version>${ozark-resteasy.version}</version>
  </dependency>
  <dependency>
    <groupId>org.mvc-spec.ozark.ext</groupId>
    <artifactId>ozark-thymeleaf</artifactId>
    <version>${ozark-thymeleaf.version}</version>
    <exclusions>
      <exclusion>
        <groupId>org.thymeleaf</groupId>
        <artifactId>thymeleaf</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf</artifactId>
    <version>${thymeleaf.version}</version>
  </dependency>
  <dependency>
    <groupId>org.webjars</groupId>
    <artifactId>material-design-icons</artifactId>
    <version>${material-design-icons.version}</version>
  </dependency>
  <dependency>
    <groupId>org.webjars</groupId>
    <artifactId>jquery</artifactId>
    <version>${jquery.version}</version>
  </dependency>
  <dependency>
    <groupId>org.webjars</groupId>
    <artifactId>materializecss</artifactId>
    <version>${materializecss.version}</version>
  </dependency>
</dependencies>
```

gardle

```
allprojects {
  dependencies {
    implementation 'org.webjars:material-design-icons:3.0.1'
    implementation 'org.webjars:materializecss:1.0.0'

    implementation 'javax.mvc:javax.mvc-api:1.0-pr'
    implementation "org.mvc-spec.ozark:ozark-resteasy:$ozarkVersion"
    implementation "org.mvc-spec.ozark.ext:ozark-thymeleaf:$ozarkVersion", {
      exclude group: 'org.thymeleaf', module: 'thymeleaf'
    }
    implementation 'org.thymeleaf:thymeleaf:3.0.11.RELEASE'
  }
}
```

2.3. configure thymeleaf

configure Thymeleaf view resolver

```
@Slf4j
@ApplicationScoped
public class ThymeleafViewEngine extends ViewEngineBase {

  @Inject ServletContext servletContext;
  TemplateEngine engine;

  @Override
  public boolean supports(final String view) {
    return !view.contains(".");
  }

  @PostConstruct
  public void postConstruct() {
    final ServletContextTemplateResolver resolver = new
ServletContextTemplateResolver(servletContext);
    resolver.setPrefix("/WEB-INF/layouts/");
    resolver.setSuffix(".html");
    resolver.setTemplateMode(TemplateMode.HTML);
    resolver.setCacheable(false);
    engine = new TemplateEngine();
    engine.setTemplateResolver(resolver);
  }

  @Override
  public void processView(ViewEngineContext context) throws ViewEngineException {
    Try.run(() -> {

      final HttpServletRequest request = context.getRequest();
      final HttpServletResponse response = context.getResponse();
      final WebContext webContext = new WebContext(request, response, servletContext,
request.getLocale());

      webContext.setVariables(context.getModels());
      request.setAttribute("view", context.getView());
      engine.process(/*default layout*/ "default", webContext, response.getWriter());

    }).getOrElseThrow(ViewEngineException::new);
  }
}
```

2.4. configure JAX-RS serve static files and webjars

StaticResourcesResource.java *file:*

```
@Slf4j
@Path("")
@RequestScoped
public class StaticResourcesResource {

  @Inject ServletContext context;

  /**
   * Serving webjar dependencies
   *
   * see: https://www.webjars.org
   */

  @GET
  @Path("{path: ^webjars\\/.*}")
  public Response webjars(@PathParam("path") final String path) {
    log.debug("handling webjars: '{}'", path);
    String absolutePath = format("/META-INF/resources/%s", path);
    InputStream resource = getClass().getClassLoader().getResourceAsStream
(absolutePath);
    return Objects.isNull(resource)
        ? Response.status(NOT_FOUND).build()
        : Response.ok().entity(resource).build();
  }

  /**
   * Serving static files form folders:
   *
   * /WEB-INF/resources
   * /WEB-INF/static
   * /WEB-INF/public
   * /WEB-INF/assets
   */

  @GET
  @Path("{path: ^(assets|public|static|resources)\\/.*}")
  public Response staticResources(@PathParam("path") final String path) {
    log.debug("handling assets: '{}'", path);
    InputStream resource = context.getResourceAsStream(format("/WEB-INF/%s", path));
    return null == resource
        ? Response.status(NOT_FOUND).build()
        : Response.ok().entity(resource).build();
  }
}
```

2.5. create MVC controller

resteasy controller

```
@Path("")
@Controller
@Produces(TEXT_HTML)
public class IndexPage {

  @Inject Models models;

  @GET
  @Path("")
  public String indexView() {
    models.put("message", "Hello, World!");
    models.put("data", asList("ololo", "trololo"));
    return "index";
  }
}
```

2.6. add default (basic) pages layout

Thymeleaf default layout template in `src/main/resources/webapp/WEB-INF/layouts/default.html` *file:*

```
<!doctype html>
<html lang="en"
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="IE=Edge" />
  <title>Template | Thymeleaf EE</title>
  <link rel="icon" type="image/x-icon" th:href="@{/assets/favicon.ico}" />
  <link rel="stylesheet" type="text/css" th:href="@{/webjars/material-design-
icons/3.0.1/material-icons.css}"
        href="/webjars/material-design-icons/3.0.1/material-icons.css"/>
  <link rel="stylesheet" type="text/css" th:href=
"@{/webjars/materializecss/1.0.0/css/materialize.min.css}"
        href="/webjars/materializecss/1.0.0/css/materialize.css"/>
  <link rel="stylesheet" type="text/css" th:href="@{/assets/styles.css}" />
  <!-- custom styles will be added here...-->
  <th:block th:include="${view} :: header" />
</head>
<body>

<div th:text="${message}">Here should be some basic message...</div>
<ul th:each="item : ${data}">
  <li th:text="${item}"></li>
</ul>

<!-- custom view specific stuff: -->
<div th:replace="${view} :: content"></div>

<script th:src="@{/webjars/materializecss/1.0.0/js/materialize.min.js}"
        src="/webjars/materializecss/1.0.0/js/materialize.js"></script>
<script th:src="@{/assets/scripts.js}"></script>
<!-- custom scripts will be added here...-->
<th:block th:include="${view} :: footer" />
</body>
</html>
```

2.7. and finally create view for index page

Thymeleaf index view template (file `src/main/resources/webapp/WEB-INF/layouts/index.html`*):*

```
<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org"
      xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Hello | Thymeleaf EE</title>
  <div th:fragment="header" th:remove="tag">
    <!-- additional styles -->
  </div>
</head>
<body th:fragment="content">
<h1>Hello!</h1>
<div th:fragment="footer" th:remove="tag">
  <!-- additional scripts -->
</div>
</body>
</html>
```

Chapter 3. Links

- GitHub repo
- GitHub pages

Chapter 4. Other links

- DZone: Thymeleaf With JavaEE 8 article
- JSR 371: Model-View-Controller Specification
- Resteasy docs
- Ozark: MVC for JavaEE