eBook Gratuit

APPRENEZ thymeleaf

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#thymeleaf

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: thymeleaf

It is an unofficial and free thymeleaf ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official thymeleaf.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec thymeleaf

Remarques

Thymeleaf est un moteur de template, une bibliothèque écrite en JAVA. Il permet à un développeur de définir un modèle de page HTML, XHTML ou HTML5 et de le remplir ultérieurement de données pour générer la page finale. Par conséquent, il réalise une partie *Model-View* d'un modèle Model-View-Controller .

Le principe de conception important de Thymeleaf est qu'un modèle lui-même doit être correctement écrit (X) HTML.

Versions

Version	Rendez-vous amoureux	Dernière version	Rendez-vous amoureux
3.xx	2016-05-08	3.0.6	2017-05-07
2.xx	2012-02-09	2.1.5	2016-07-11

Examples

Configuration

Pour commencer avec Thymeleaf, visitez la page de téléchargement officielle .

Dépendance de Maven

```
<dependency>
  <groupId>org.thymeleaf</groupId>
  <artifactId>thymeleaf</artifactId>
  <version>3.0.1.RELEASE</version>
</dependency>
```

Gradle dépendance

```
compile group: 'org.thymeleaf', name: 'thymeleaf', version: '3.0.1.RELEASE'
```

Exemple de configuration

À partir de la version 3.0, Thymeleaf ne prend en charge que la configuration Java.

```
public ViewResolver viewResolver() {
    ThymeleafViewResolver resolver = new ThymeleafViewResolver();
    resolver.setTemplateEngine(templateEngine());
    resolver.setCharacterEncoding("UTF-8");
```

```
    resolver.setContentType("text/html; charset=UTF-8");
    return resolver;
}
```

Dans la méthode `viewResolver()` , vous pouvez configurer, par exemple, le codage et le type de contenu pour les vues. Plus d'information

```
public TemplateEngine templateEngine() {
    SpringTemplateEngine engine = new SpringTemplateEngine();
    engine.setTemplateResolver(templateResolver());
    return engine;
}
```

Dans `templateEngine()` , vous pouvez ajouter des dialectes personnalisés. Par exemple, pour ajouter un dialecte de sécurité Spring, vous pouvez le faire comme ceci `engine.addDialect(new SpringSecurityDialect());`

```
public ITemplateResolver templateResolver() {
    SpringResourceTemplateResolver resolver = new SpringResourceTemplateResolver();
    resolver.setApplicationContext(applicationContext);
    resolver.setPrefix("/views/");
    resolver.setSuffix(".html");
    resolver.setTemplateMode(TemplateMode.HTML);
    resolver.setCharacterEncoding("UTF-8");
    return resolver;
}
```

Regardez le setter pour le préfixe et le suffixe dans la méthode `templateResolver()` . Il indique à Thymeleaf que chaque fois que le contrôleur renverra la vue, Thymeleaf regardera ces noms html dans `webapp/views/` directory et ajoutera le suffixe `.html` pour vous.

Exemple

```
@RequestMapping(value = "/")
public String homePage() {
    return "foo/my-index";
}
```

Thymeleaf recherchera le HTML nommé `my-index.html` dans le `webapp/views/foo/` . Selon l'exemple de configuration ci-dessus.

Utiliser des cases à cocher

Exemple de méthode dans le contrôleur

```
@RequestMapping(value = "/test")
public String showCheckbox(Model model) {
    boolean myBooleanVariable = false;
    model.addAttribute("myBooleanVariable", myBooleanVariable);
    return "sample-checkbox";
}
```

Voir: sample-checkbox.html

```
<input
      type="checkbox"
      name="myBooleanVariable"
      th:checked="${myBooleanVariable}"/>
```

Ne pas utiliser `th:name` for checboxes, juste `name`

Ajax avec Jquery

Pour soumettre le formulaire via Ajax avec Jquery:

```
    <div id="yourPanel" th:fragment="yourFragment">
        <form id="yourForm" method="POST"
              th:action="@{/actions/postForm}"
              th:object="${yourFormBean}">
        <div class="form-group">
            <label for="param1"></label>
            <input class="form-component" type="text" th:field="*{param1}" />
        </div>
        <div class="form-group">
            <label for="param2"></label>
            <input class="form-component" type="text" th:field="*{param2}" />
        </div>
        <div class="form-group">
            <label for="param3"></label>
            <input class="form-component" type="checkbox" th:field="*{param3}" />
        </div>

        <button type="submit" class="btn btn-success">Save</button>
        <a href='#' class="btn btn-default">Cancel</a>
    </form>
    </div>

<script th:inline="javascript">
    /*<![CDATA[*/
    $(document).ready(function () {
        /*[+
         var postUrl = [[@{/actions/postForm(
         additionalParam=${#httpServletRequest.getParameter('additionalParam')}
         )}]];
         +]*/
        $("#yourForm").submit(function (e) {
            e.preventDefault();
            $.post(postUrl,
                    $(this).serialize(),
                    function (response) {
                        var isErr = 'hasError';
                        // when there are an error then show error
                        if (response.indexOf(isErr) > -1) {
                            $("#yourPanel").html(response);
                        } else {
                            var formData = $("#yourForm").serializeArray(),
                                    len = formData.length,
                                    urlEnd = '';
                            for (i = 0; i < len; i++) {
                                urlEnd += formData[i].name + '=' +
```

```
encodeURIComponent(formData[i].value) + '&';
                            }

                            /*[+
                             var urlReplacement = [[@{/another/page(

additionalParam=${#httpServletRequest.getParameter('additionalParam')}
                             )}]] + urlEnd;
                             +]*/

                            window.location.replace(urlReplacement);
                        }
                    }
            );
            return false;
        });
    });
    /*]]>*/
</script>
```

Classe YourFormBean:

```
@lombok.Getter
@lombok.Setter
@lombok.NoArgsConstructor
public class YourFormBean {
    private String param1;
    private String param2;
    private boolean param3;
}
```

Code du contrôleur:

```
@RequestMapping(value = "/actions/postForm", method = RequestMethod.POST)
public String saveForm(Model model,
        @RequestParam("additionalParam") Integer additionalParam,
        @Valid @ModelAttribute("yourFormBean") YourFormBean yourFormBean,
        BindingResult bindingResult,
        RedirectAttributes redirectAttributes) {
    if (bindingResult.hasErrors()) {
        model.addAttribute("hasError", true);
        return "your/template :: yourFragment";
    }
    redirectAttributes.addAttribute("additionalParam", additionalParam);

    return "redirect:/another/page";
}
```

Remplacer des fragments par ajax

Si vous souhaitez remplacer des parties de votre site Web, ajax est un moyen facile de le faire.

Le **site Web.html** où vous souhaitez remplacer le contenu en fonction de la valeur sélectionnée:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
```

```
  xmlns:th="http://www.thymeleaf.org">

    <head>
        <title>Index</title>
    </head>

    <body>
        <select id="selection">
            <option>Content 1</option>
            <option>Content 2</option>
        </select>

        <div id="replace_div">
            Content goes here
        </div>

        <!-- JQury from Google CDN -->
        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>

        <script>
            $(document).ready(function () {

                //call function when page is loaded
                getContent();

                //set on change listener
                $('#selection').change(getContent);

                function getContent() {

                    //create url to request fragment
                    var url = /content/;
                    if ($('#selection').val() === "Content 1") {
                        url = url + "content1";
                    } else {
                        url = url + "content2";
                    }

                    //load fragment and replace content
                    $('#replace_div').load(url);
                }
            })
        </script>
    </body>
</html>
```

Et le **content.html** avec les fragments que vous voulez inclure en fonction de la valeur sélectionnée:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
    <head>
    </head>

    <body>
        <div th:fragment="content1">
            This is Content 1
        </div>
```

```
        <div th:fragment="content2">
            This is Content 2
         </div>
    </body>
</html>
```

Last but not least le Spring MVC **ContentController.java** :

```
@Controller
@RequestMapping("content")
public class ContentController {

    @RequestMapping("")
    public String loadContent() {
        return "website";
    }

    @RequestMapping("content1")
    public String getContent1() {
        return "content :: content1";
    }

    @RequestMapping("content2")
    public String getContent2() {
        return "content :: content2";
    }
}
```

Soumission de formulaire

Objet de formulaire

```
package formSubmission;

public class Person {

    private String name;
    private int age;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name= name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }

}
```

Manette

```
package formSubmission;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class FriendsController {

    @GetMapping("/friends")
    public String friendForm(Model model) {
        model.addAttribute("personForm", new Person());
        return "friendsForm";
    }

    @PostMapping("/friends")
    public String submissionResult(@ModelAttribute("personForm") Person person) {
        return "result";
    }

}
```

friendsForm.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Friend form</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <h1>Friend Form</h1>
    <form th:action="@{/friends}" th:object="${personForm}" method="post">
        <p>Name: <input type="text" th:field="*{name}"/></p>
        <p>Age: <input type="number" th:field="*{age}"/></p>
        <p><input type="submit" value="Submit"/></p>
    </form>
</body>
</html>
```

result.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Submission result</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <h1>th:text="'My friend ' + ${personForm.name} + ' is ' + ${personForm.age} + ' years
old'"</h1>
</body>
</html>
```

Lire Démarrer avec thymeleaf en ligne: https://riptutorial.com/fr/thymeleaf/topic/1895/demarrer-avec-thymeleaf

Chapitre 2: Externalisation du texte dans Thymeleaf

Examples

Localisation

1. Créer des fichiers pour vos messages

```
messages.properties
messages_en.properties
messages_fr.properties
...
```

2. Ecrire des messages dans ces fichiers comme ceci

```
header.label.title=Title
```

3. Configurez le chemin d'accès à ces fichiers (dans ce cas, dans le dossier D: / project / messages) dans des propriétés d'application telles que:

```
messages.basename.path=D:/project/messages/messages
```

4. Configurer MessageSource

```
@Value("${messages.basename.path}")
private String messagesBasename;

@Bean
public MessageSource messageSource() {
       ReloadableResourceBundleMessageSource messageSource = new
ReloadableResourceBundleMessageSource();
       messageSource.setFallbackToSystemLocale(false);
       messageSource.setBasenames("file:" + messagesBasename);
       return messageSource;
   }
```

5. Utiliser les messages sur les pages

```
<p th:text="#{header.label.title}">Title</p>
```

Messages de localisation avec paramètres

Ecrire un message dans messages.properties

```
welcome.message=Hello, {0}!
```

Remplacez {0} par le nom d'utilisateur dans la balise thymeleaf

```
<h3 th:text="#{welcome.message(${some.variable})}">Hello, Placeholder</h3>
```

Lire Externalisation du texte dans Thymeleaf en ligne:
https://riptutorial.com/fr/thymeleaf/topic/10668/externalisation-du-texte-dans-thymeleaf

Chapitre 3: Objets utilitaires d'expression

Examples

Date de format

```
<p>
  Today: <span th:text="${#calendars.format(today,'dd MMMM yyyy')}">30 May 2017</span>
</p>
```

Longueur de chaine

```
<div th:if="*{userMessage!=null and #strings.length(userMessage)>0}">
     <label th:text = "*{userMessage}"/>
</div>
```

La chaîne contient

```
<div th:if="${#strings.contains(#httpServletRequest.requestURI, 'email')}">
    <div th:replace="fragments/email::welcome">
</div>
```

Date d'analyse

Obtenez l'année à partir de la date

```
<p>
  Year: <span th:text="${#dates.year(today)}">2017</span>
</p>
```

Obtenez le mois

```
<p>
  Month number: <span th:text="${#dates.month(today)}">8</span>
  Month: <span th:text="${#dates.monthName(today)}">August</span>
  Month short name: <span th:text="${#dates.monthNameShort(today)}">Aug</span>
</p>
```

Obtenez le jour

```
<p>
  Day: <span th:text="${#dates.day(today)}">26</span>
</p>
```

Obtenez le jour de la semaine

```
<p>
```

```
    Day: <span th:text="${#dates.dayOfWeek(today)}">1</span>
    Day: <span th:text="${#dates.dayOfWeekName(today)}">Monday</span>
    Day: <span th:text="${#dates.dayOfWeekNameShort(today)}">Mo</span>
</p>
```

Obtenir du temps

```
<p>
    Hour: <span th:text="${#dates.hour(today)}">10</span>
    Minute: <span th:text="${#dates.minute(today)}">50</span>
    Second: <span th:text="${#dates.second(today)}">48</span>
    Millisecond: <span th:text="${#dates.millisecond(today)}">48</span>
<p>
```

Format décimal

```
<p>
    Order sum: <span th:text="${#numbers.formatDecimal(orderSum, 0, 'COMMA', 2,
'POINT')}">1,145,000.52</span>
</p>
```

Lire Objets utilitaires d'expression en ligne: https://riptutorial.com/fr/thymeleaf/topic/10675/objets-utilitaires-d-expression

Chapitre 4: Spring Security et Thymeleaf

Examples

Sécurisez votre WebApp avec Login et Logout

Cet exemple est une application Spring Boot très simple.

Dépendances Maven

Ajoutez d'abord les dépendances suivantes à votre projet. Spring Initializr est recommandé lorsque vous créez un nouveau projet.

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.1.RELEASE</version>
    <relativePath/>
</parent>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.thymeleaf.extras</groupId>
        <artifactId>thymeleaf-extras-springsecurity4</artifactId>
    </dependency>
</dependencies>
```

Créez votre WebApp

Créez une application Web avec des sites Web et un contrôleur. Par exemple, cette très petite application Web avec une seule page (index.html) et une entrée pour la page de connexion.

```
@Configuration
public class MvcConfig extends WebMvcConfigurerAdapter{

    @Override
    public void addViewControllers(ViewControllerRegistry registry) {
```

```
        registry.addRedirectViewController("/", "index");
        registry.addViewController("/index").setViewName("index");
        registry.addViewController("/login").setViewName("login");
    }
}
```

Sécurisez votre WebApp

Configurez Spring Security pour sécuriser votre application Web. Par exemple. autoriser toute demande par les utilisateurs authentifiés uniquement. Autorise les ressources statiques comme js et css, sinon elles ne seront pas chargées pour les utilisateurs non authentifiés. Exclure la page de connexion et de déconnexion de cette règle et créer un utilisateur de test:

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/css/*.css", "/js/*.js").permitAll()
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/login")
                .permitAll()
                .and()
                .logout()
                .permitAll();
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.inMemoryAuthentication()
                .withUser("user").password("password").roles("USER");
    }
}
```

Créer la page de connexion

La page de connexion doit avoir un formulaire qui envoie une demande de publication à "/ login":

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">

    <head>
        <title>Login</title>
        <link th:href="@{/css/stylesheet.css}" rel="stylesheet" type="text/css"/>
        <script type="text/javascript" th:src="@{/js/login.js}"></script>
    </head>
    <body>
        <!-- show notification on error -->
```

```
        <div th:if="${param.error}">
            Invalid username or password.
        </div>

        <!-- show notification of logout -->
        <div th:if="${param.logout}">
            You have been logged out.
        </div>

        <!-- login form -->
        <div>
            <form th:action="@{/login}" method="post">
                <h2 >Please sign in</h2>
                <label >User Name</label>
                <input type="text" name="username" th:required="required"
th:autofocus="autofocus"/>
                <br/>

                <label>Password</label>
                <input type="password" name="password" th:required="required" />
                <br/>

                <input type="submit" value="Sign In"/>
            </form>
        </div>
    </body>
</html>
```

Lorsque l'utilisateur entre le mauvais nom d'utilisateur / mot de passe, le paramètre d'erreur est défini. Lorsque l'utilisateur se déconnecte, le paramètre de déconnexion est défini. Ceci est utilisé pour afficher les messages correspondants.

Accéder aux propriétés de l'utilisateur

Après une connexion réussie, l'utilisateur est dirigé vers le **fichier index.html** . Le Spring Security Dialect nous permet d'accéder aux propriétés de l'utilisateur comme son nom d'utilisateur:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Index</title>
    </head>
    <body>
        <div>
            <h3>Welcome <span th:text="${#authentication.name}"/></h3>
            <form th:action="@{/logout}" method="post">
                <input type="submit" value="Logout"/>
            </form>
        </div>
    </body>
</html>
```

Une déconnexion est réalisée via post request to "/ logout"

Affichage de quelque chose pour les utilisateurs authentifiés uniquement

```
<div sec:authorize="isAuthenticated()">
    This text is displayed for authenticated users.
</div>
```

Afficher le nom d'utilisateur

Vous pouvez afficher le nom d'utilisateur pour les utilisateurs authentifiés

```
<div sec:authorize="isAuthenticated()">
    Welcome, <span sec:authentication="name">Username</span>
</div>
```

Afficher un contenu différent pour différents rôles

L'attribut sec: authorize affiche son contenu lorsque l'expression d'attribut est évaluée à true

```
<div sec:authorize="hasRole('ROLE_ADMIN')">
    Content for administrators
</div>
<div sec:authorize="hasRole('ROLE_USER')">
    Content for users
</div>
```

L'attribut sec: authentication est utilisé pour imprimer les rôles d'utilisateur journalisés:

```
Roles: <span sec:authentication="principal.authorities">ROLE_USER, ROLE_ADMIN</span>
```

Lire Spring Security et Thymeleaf en ligne: https://riptutorial.com/fr/thymeleaf/topic/9190/spring-security-et-thymeleaf

Chapitre 5: Utilisation des listes avec Thymeleaf

Examples

En utilisant la liste dans select

Vous pouvez utiliser la variable de liste pour former des éléments `<select>`

```
 <select th:field="*{countries}">
     <option th:each="country: ${countries}"
             th:value="${country.id}"
             th:text="#{${'selected.label.' + country.name}}"/>
</select>
```

Tableau de forme

```
<table id="countryList">
    <thead>
        <tr>
            <th th:text="#{country.label.name}"> Country </th>
            <th th:text="#{country.label.capital}"> Capital </th>
            <th th:text="#{country.label.square}"> Square </th>
        </tr>
    </thead>
    <tbody>
        <tr th:each="country : ${countryList}">
            <td th:text="${country.name}"></td>
            <td th:text="${country.capital}"></td>
            <td th:text="${country.square}"></td>
        </tr>
    </tbody>
</table>
```

Lire Utilisation des listes avec Thymeleaf en ligne:
https://riptutorial.com/fr/thymeleaf/topic/10676/utilisation-des-listes-avec-thymeleaf

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec thymeleaf	Aboodz, Alexander, Community, guille11, Jakub Pomykała, Maciek Łoziński, ppeterka, Prabhat, Rob Streeter, sanluck
2	Externalisation du texte dans Thymeleaf	Elizabeth
3	Objets utilitaires d'expression	Elizabeth
4	Spring Security et Thymeleaf	Alexander, Elizabeth, Sébastien Temprado
5	Utilisation des listes avec Thymeleaf	Elizabeth