

*Spring MVC and Thymeleaf: how to access data from templates*

*Written by Rafał Borowiec —*[*http://blog.codeleak.pl*](http://blog.codeleak.pl/)

In a typical Spring MVC application, **@Controller** classes are responsible for preparing a model map with data and selecting a view to be rendered. This *model map* allows for the complete abstraction of the view technology and, in the case of Thymeleaf, it is transformed into a Thymeleaf context object (part of the Thymeleaf *template execution context*) that makes all the defined variables available to expressions executed in templates.

*Spring model attributes*

Spring MVC calls the pieces of data that can be accessed during the execution of views *model attributes*. The equivalent term in Thymeleaf language is *context variables*.

There are several ways of adding model attributes to a view in Spring MVC. Below you will find some common cases:

Add attribute to **Model** via its **addAttribute** method:

@RequestMapping(value = "message", method = RequestMethod.GET)

public String messages(Model model) {

model.addAttribute("messages", messageRepository.findAll());

return "message/list";

}

Return **ModelAndView** with model attributes included:

@RequestMapping(value = "message", method = RequestMethod.GET)

public ModelAndView messages() {

ModelAndView mav = new ModelAndView("message/list");

mav.addObject("messages", messageRepository.findAll());

return mav;

}

Expose common attributes via methods annotated with **@ModelAttribute**:

@ModelAttribute("messages")

public List<Message> messages() {

return messageRepository.findAll();

}

As you may have noticed, in all the above cases the **messages** attribute is added to the model and it will be available in Thymeleaf views.

In Thymeleaf, these model attributes (or *context variables* in Thymeleaf jargon) can be accessed with the following syntax: **${attributeName}**, where **attributeName** in our case is **messages**. This is a [Spring EL](http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html/expressions.html) expression. In short, Spring EL (Spring Expression Language) is a language that supports querying and manipulating an object graph at runtime.

You can access model attributes in views with Thymeleaf as follows:

<tr th:each="message : ${messages}">

<td th:text="${message.id}">1</td>

<td><a href="#" th:text="${message.title}">Title ...</a></td>

<td th:text="${message.text}">Text ...</td>

</tr>

*Request parameters*

Request parameters can be easily accessed in Thymeleaf views. Request parameters are passed from the client to server like:

https://example.com/query?q=Thymeleaf+Is+Great!

Let’s assume we have a **@Controller** that sends a redirect with a request parameter:

@Controller

public class SomeController {

@RequestMapping("/")

public String redirect() {

return "redirect:/query?q=Thymeleaf+Is+Great!";

}

}

In order to access the **q** parameter you can use the **param.** prefix:

<p th:text="${param.q}">Test</p>

In the above example if parameter **q** is not present, empty string will be displayed in the above paragraph otherwise the value of **q** will be shown.

Since parameters can be multivalued (e.g. `https://example.com/query?q=Thymeleaf%20Is%20Great!&q=Really%3F) you may access them using brackets syntax:

<p th:text="${param.q[0] + ' ' + param.q[1]}" th:unless="${param.q == null}">Test</p>

Note: If you access multivalued parameter with **${param.q}** you will get a serialized array as a value.

Another way to access request parameters is by using the special **#request** object that gives you direct access to the **javax.servlet.http.HttpServletRequest** object:

<p th:text="${#request.getParameter('q')}" th:unless="${#request.getParameter('q') == null}">Test</p>

*Session attributes*

In the below example we add **mySessionAttribute** to session:

@RequestMapping({"/"})

String index(HttpSession session) {

session.setAttribute("mySessionAttribute", "someValue");

return "index";

}

Similarly to the request parameters, session attributes can be accessed by using the **session.** prefix:

<p th:text="${session.mySessionAttribute}" th:unless="${session == null}">[...]</p>

Or by using **#session**, that gives you direct access to the **javax.servlet.http.HttpSession** object: **${#session.getAttribute('mySessionAttribute')}**

*ServletContext attributes*

The ServletContext attributes are shared between requests and sessions. In order to access ServletContext attributes in Thymeleaf you can use the **#servletContext.** prefix:

<table>

<tr>

<td>My context attribute</td>

<!-- Retrieves the ServletContext attribute 'myContextAttribute' -->

<td th:text="${#servletContext.getAttribute('myContextAttribute')}">42</td>

</tr>

<tr th:each="attr : ${#servletContext.getAttributeNames()}">

<td th:text="${attr}">javax.servlet.context.tempdir</td>

<td th:text="${#servletContext.getAttribute(attr)}">/tmp</td>

</tr>

</table>

*Spring beans*

Thymeleaf allows accessing beans registered at the Spring Application Context with the **@beanName** syntax, for example:

<div th:text="${@urlService.getApplicationUrl()}">...</div>

In the above example, **@urlService** refers to a Spring Bean registered at your context, e.g.

@Configuration

public class MyConfiguration {

@Bean(name = "urlService")

public UrlService urlService() {

return () -> "domain.com/myapp";

}

}

public interface UrlService {

String getApplicationUrl();

}

This is fairly easy and useful in some scenarios.

*References*

* [Thymeleaf + Spring 3](http://www.thymeleaf.org/doc/tutorials/3.0/thymeleafspring.html)
* [Expression Basic Objects](http://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html#appendix-a-expression-basic-objects)