Understanding View Templates

MVC软件设计模式是一种用于在软件应用程序中分离关注度的方法

The Model-View-Controller (MVC) software design pattern is a method for separating concerns within a software application. In principal, the application logic, or controller, is separated from the technology used to display information to the user, or the view layer. The model is a communications vehicle between the controller and view layers.

原则上，应用逻辑/控制器应与向用户显示信息/视图层的技术分离。
模型是控制器和视图层之间的通信媒介。

Within an application, the view layer may use one or more different technologies to render the view. Spring web-based applications support a variety of view options, often referred to as view templates. These technologies are described as "templates" because they provide a markup language to expose model attributes within the view during server-side rendering.

View template libraries

The following view template libraries, among others, are compatible with Spring:

- JSP/JSTL
- Thymeleaf
- Tiles
- Freemarker
- Velocity

Comparing JSP and Thymeleaf

The following examples illustrate how to render the same content with JSP and Thymeleaf templates. For more detail, see this Spring blog post.

JSP

Note the JSTL (JavaServer Pages Standard Tag Library) expressions in this example.

```
<c:url var="hotelsUrl" value="/hotels"/>
<form:form modelAttribute="searchCriteria" action="${hotelsUrl}" method="get" cssClass="inline">
    <span class="errors span-18">
        <form:errors path="*"/>
    </span>
    <fieldset>
        <div class="span-8">
            <label for="searchString">SeaString:</label>
            <form:input id="searchString" path="searchString"/>
        </div>
        ...
    </fieldset>
</form:form>
```

Thymeleaf

In this example, the markup integrates with standard HTML.

```
<form action="#" th:object="${searchCriteria}" th:action="@{/hotels}" method="get" class="inline">
    <ul th:if="${#fields.hasErrors('*')}" class="errors span-18">
```

```
        <li th:each="err : ${#fields.errors('*')}" th:text="${err}">Input is incorrect</li>
    </ul>
    <fieldset>
        <div class="span-8">
            <label for="searchString">Search String:</label>
            <input type="text" id="searchString" th:field="*{searchString}" />
        </div>
        ...
    </fieldset>
</form>
```