Skillbox

# Макеты шаблонов. Фрагменты

**Кирилл Кошаев**

Java-разработчик в «Газпром информ»

# Определение и ссылка

В шаблоны часто необходимо включать части из других шаблонов такие, как: футеры, заголовки, меню и тому подобное.
Для этого Thymeleaf предлагает концепцию фрагментов для включения их в шаблоны с помощью атрибута th: fragment

## Код фрагмента footer.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <body>
    <div th:fragment="basicFooter">
      <!--footer content--></div>
  </body>
</html>
```

## Код thymeleaf-шаблона, где применяется фрагмент footer.html

```
<body>
  ...
  <div th:insert="~{footer :: basicFooter}"></div>
 </body>
```

# th:replace vs. th:insert

**Код фрагмента footer.html**

```
...<footer th:fragment="basicFooter">
      <!--footer content-->
</footer>...
```

**Код thymeleaf-шаблона, где применяется фрагмент footer.html**

```
...<body>
  <div th:insert="footer :: basicFooter"></div>
  <div th:replace="footer :: basicFooter"></div>
 </body>...
```

**Результирующий код web-страницы**

```
...<body>
  <div>
    <footer><!--footer content--></footer>
  </div>
  <footer><!--footer content--></footer>
</body>...
```

# Fragment expression ~{...}

Синтаксис фрагментных выражений довольно прост.
Есть три разных формата:

- **`"~{templatename::selector}"`**
- **`"~{templatename}"`**
- **`~{::selector}"`** или **`"~{this::selector}"`**

Части templatename и selector могут быть сами по себе полноценными выражениями и даже условиями

```
<div th:insert="footer :: (${user.isAdmin}? #{footer.admin} :
#{footer.user})">
</div>
```

# th:fragment

Благодаря мощи селекторов, возможно включать фрагменты, не использующие атрибуты th:fragment, что позволяет использовать фрагмент, просто вызывая его по атрибуту id как CSS-селектор

## Код фрагмента footer.html

```
<div id="basicFooter">
      <!--footer content-->
</div>
```

## Код thymeleaf-шаблона, где применяется фрагмент footer.html

```
<body>
  ...
  <div th:insert="~{footer :: #basicFooter}"></div>
  <div th:replace="~{footer :: #basicFooter}"></div>
 </body>
```

Skillbox

# Параметризация сигнатур фрагментов

## Определение фрагмента

```
<div th:fragment="fragment (param1,param2)"
th:assert="${!#strings.isEmpty(param1.name)}">
    <p th:text="${param1} + ' - ' + ${param2}">...</p>
 </div>
```

## Вставка фрагмента

```
<div th:insert="::frag (${value1},${value2})">...</div>
```

Skillbox

# Гибкие макеты

С помощью выражений-фрагментов мы можем указывать в качестве параметров для фрагментов не только простой текст, числа или какие-то объекты, но и другие фрагменты

# Гибкие макеты

### Код фрагмента base.html

```
<head th:fragment="header_fragment(title,links)">
  <title th:replace="${title}">Title</title>
  <!--/*Fragment content*/-->
  <div th:replace="${links}">Content</div>
</head>
```

### Код thymeleaf-шаблона, где применяется фрагмент

```
...
<head th:replace="base::header_fragment(~{::title},~{::div})">
  <title>Main title</title>
  <div>Content</div>
  <p>Another content</p>
</head>
...
```

### Результирующий код web-страницы

```
<head>
  <title>Main title</title>
  <div>Content</div>
</head>
```

Skillbox

# Наследование макетов

**Макет layout.html**

```
<!DOCTYPE html>
<html th:fragment="layout (title, content)"
xmlns:th="http://www.thymeleaf.org">
<head>
    <title th:replace="${title}">Layout Title</title>
</head>
<body>
    <h1>Layout H1</h1>
    <div th:replace="${content}">
        <p>Layout content</p>
    </div>
    <footer>
        Layout footer
    </footer>
</body>
 </html>
```

Skillbox

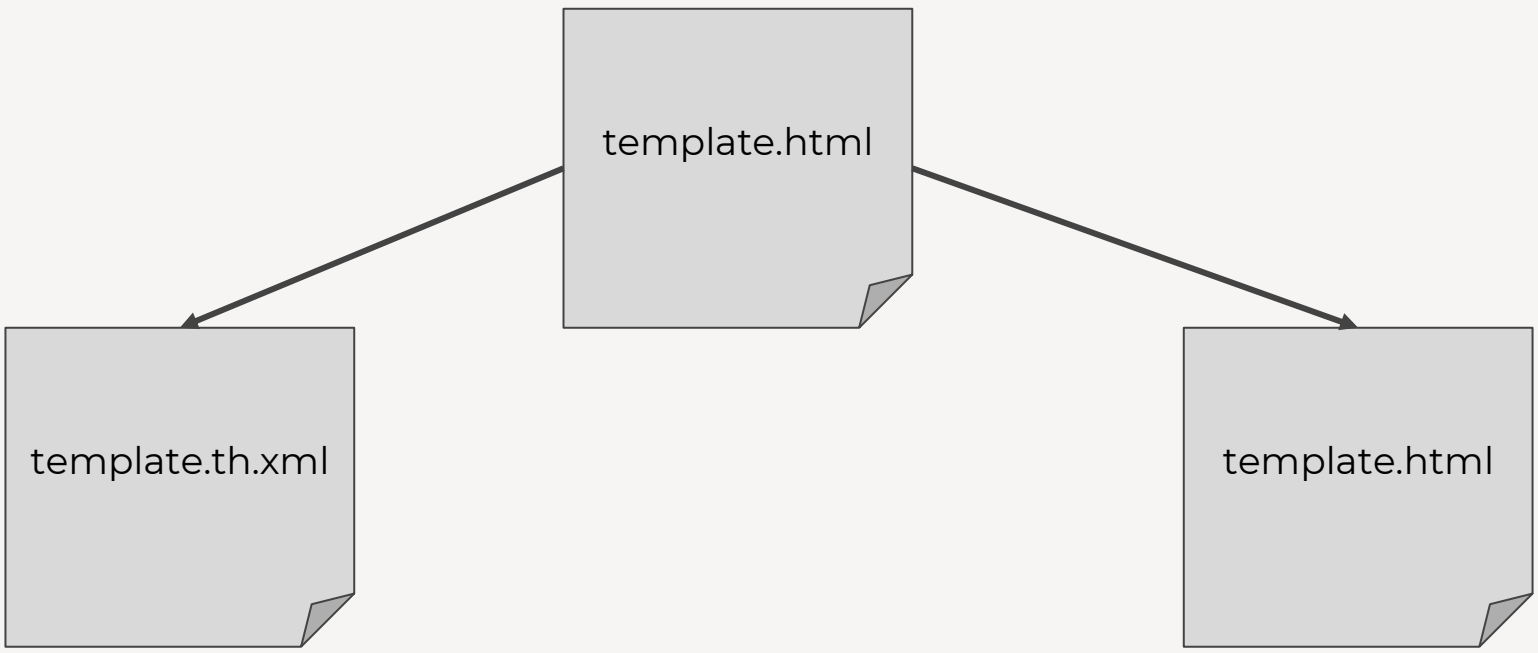
# Наследование макетов

**Шаблон, который наследует макет layout**

```
<!DOCTYPE html>
<html th:replace="~{layoutFile :: layout(~{::title}, ~{::section})}">
<head>
    <title>Page Title</title>
</head>
<body>
<section>
    <p>Page content</p>
    <div>Included on page</div>
</section>
</body>
</html>
```

# Слабосвязанная архитектура

Thymeleaf предлагает использовать другой подход, который позволяет нам полностью отделить разметку шаблона от его логики. Основная идея заключается в том, что логика шаблона будет определена в отдельном файле, с новым расширением .th.xml, а разметка в другом файле, с расширением html



# Слабосвязанная архитектура

Такой шаблон будет полностью понятен дизайнеру, не обладающему знаниями о Thymeleaf или шаблонах, и это не помешает ему создавать, редактировать и анализировать разметку

Skillbox

# Слабосвязанная архитектура

**template_markup.html**

```
<!DOCTYPE html>
<html>
  <body>
    <table id="usersTable">
      <tr>
        <td class="username">Jeremy Grapefruit</td>
        <td class="usertype">Normal User</td>
      </tr>
      <tr>
        <td class="username">Alice Watermelon</td>
        <td class="usertype">Administrator</td>
      </tr>
    </table>
  </body>
</html>
```

# Слабосвязанная архитектура

Здесь мы видим множество тегов <attr> внутри блока thlogic. Теги <attr> выполняют внедрение атрибутов в узлы исходного шаблона template.html, выбранные с помощью их атрибутов sel, которые содержат селекторы разметки Thymeleaf

```
<?xml version="1.0"?>
<thlogic>
  <attr sel="#usersTable">
    <attr sel="/tr[0]" th:each="user : ${users}">
      <attr sel="td.username" th:text="${user.name}" />
      <attr sel="td.usertype" th:text="${user.type}" />
    </attr>
  </attr>
</thlogic>
```

# Слабосвязанная архитектура

Преимущество разделённых шаблонов состоит в том, что мы можем предоставить им полную независимость от Thymeleaf и, следовательно, лучшую ремонтопригодность с точки зрения команды web-дизайнеров

```
<!DOCTYPE html>
<html>
  <body>
    <table id="usersTable">
      <tr th:each="user : ${users}">
        <td class="username" th:text="${user.name}">Jeremy Grape</td>
        <td class="usertype" th:text="${user.type}}">User</td>
      </tr>
      <tr>
        <td class="username">Jerry Waters</td>
        <td class="usertype">Administrator</td>
      </tr>
    </table>
  </body>
</html>
```

Skillbox

# Спасибо за внимание!