Java Pro 7.9

TEL-RAN
by Starta Institute

Spring MVC (часть 2)

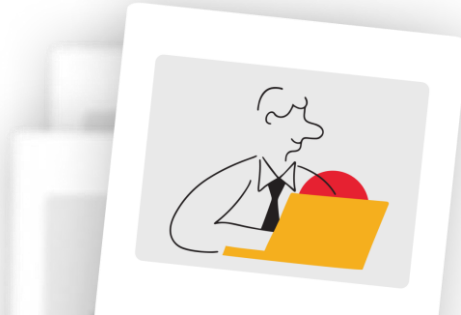
ПРЕПОДАВАТЕЛЬ

TEL-RAN
by Starta Institute

Юрий Костяной

Java/Kotlin backend-разработчик

- 3+ года опыта в коммерческой разработке
- 2+ года опыта в преподавании
- Проекты по интеграции сторонних платформ, CRM
- Проблемно-ориентированный подход в преподавании



ВАЖНО:

- Камера должна быть включена на протяжении всего занятия.
- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

TEL-RAN
by Starta Institute

ПЛАН ЗАНЯТИЯ

1. Повторение
2. Основной блок
3. Вопросы по основному блоку
4. Домашняя работа

TEL-RAN
by Starta Institute

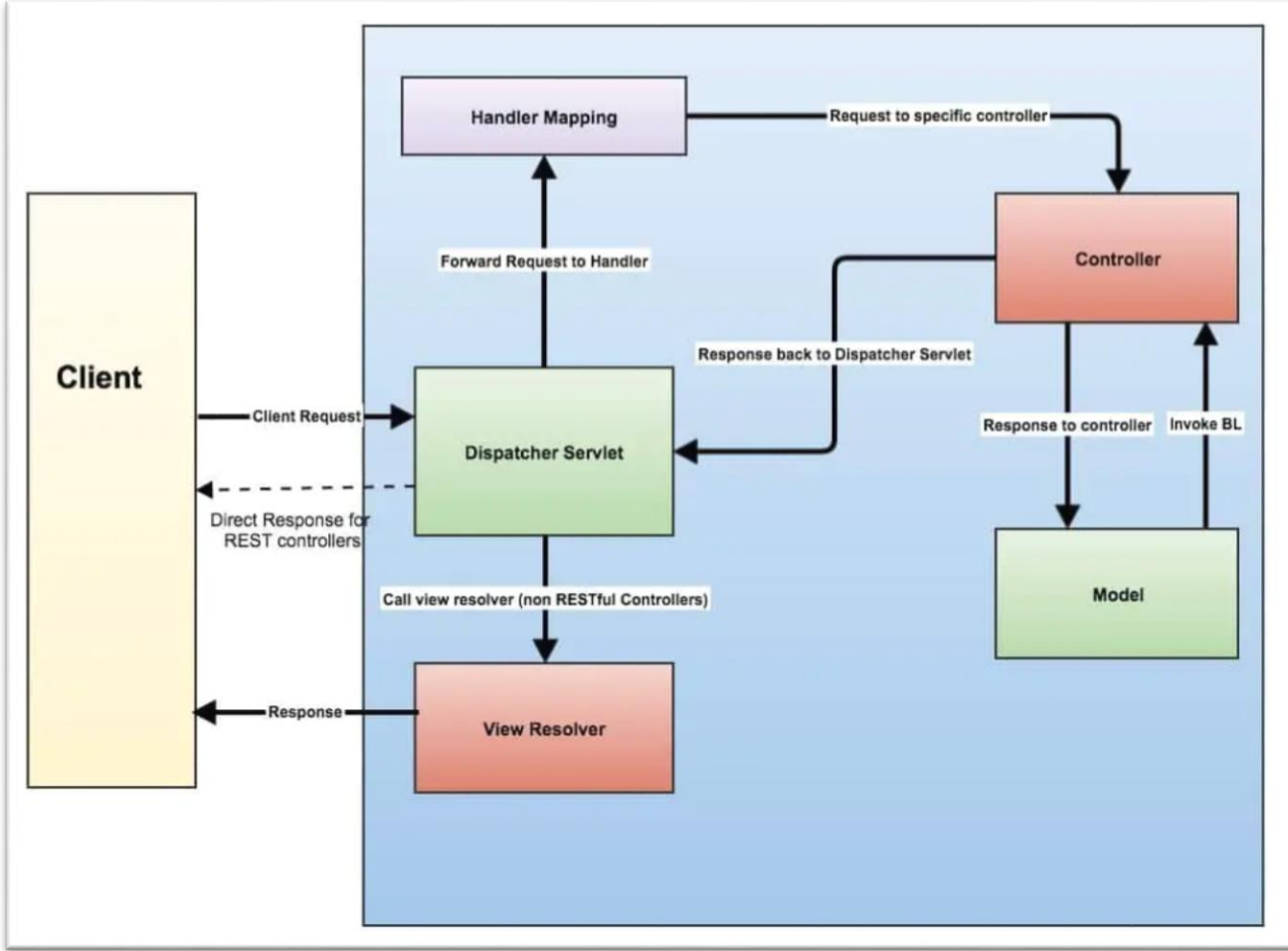
1 ПОВТОРЕНИЕ ИЗУЧЕННОГО

Повторение

Как устроено MVC приложение?

Повторение

Как устроено MVC приложение?



Повторение

Какие аннотации необходимо использовать в классе, реализующем контроллер?

Повторение

Какие аннотации необходимо использовать в классе, реализующем контроллер?

@Controller для указания, что класс реализует логику контролера.

@GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping для указания, какой метод протокола http соответствует методу класса монтроллера.

Повторение

Какие аннотации потребуются, чтобы извлечь информацию из http-запроса?

Повторение

Какие аннотации потребуются, чтобы извлечь информацию из http-запроса?

@RequestBody – взять данные из тела сообщения;

@RequestParam(“Имя_параметра”) – из параметра URL;

@PathVariable(“Имя_переменной”) – из переменной пути;

@RequestHeader(“Имя_заголовка”) – из заголовка.

Или никакие, если внедрить бин запроса напрямую в метод.

```
@GetMapping("/get-session-count")
public String testSessionListner(HttpServletRequest request, HttpServletResponse response){ }
```

Повторение

В чём отличие использование @Controller и @RestController?

Что может сделать поведение @Controller таким же, как у @RestController?

Повторение

В чём отличие использование @Controller и @RestController?

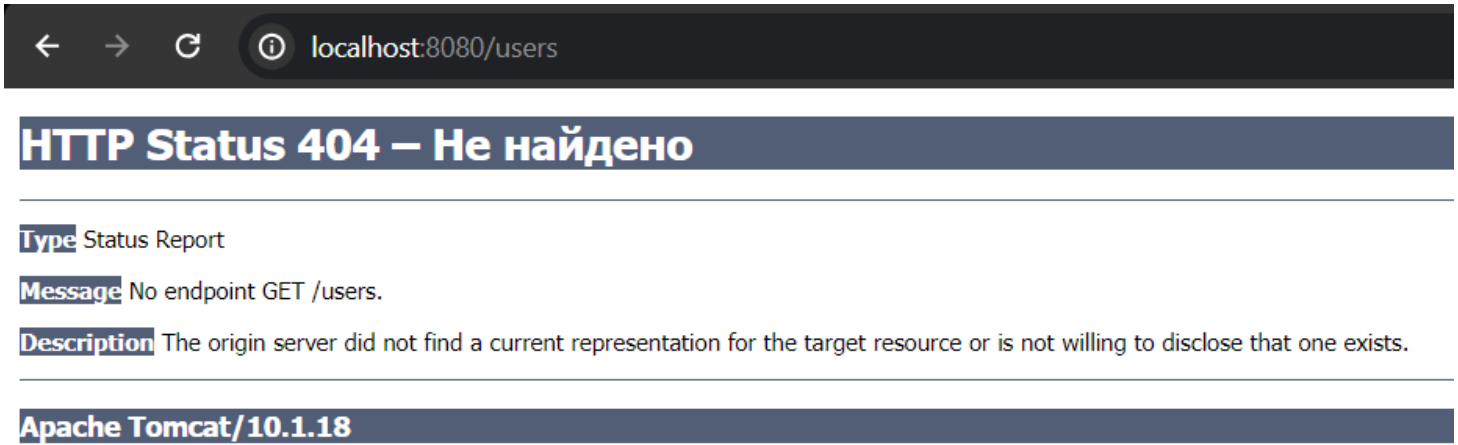
Что может сделать поведение @Controller таким же, как у @RestController?

@Controller используется, чтобы пометить класс классического MVC-приложения. Результатом выполнения методов внутри такого контроллера является строка с именем представления. Если же мы хотим вернуть, не имя представления, а тело ответа, то нужно использовать @ResponseBody над методом или перед возвращаемым типом;
@RestController используется для построения RESTful API. Его метод всегда возвращают объекты, которые отправляются как есть или сериализуются (обычно в JSON). Сериализация возвращаемого объекта происходит автоматически, без необходимости написания доп. кода.

Повторение

Почему не работает запрос?

```
@Controller
@RequestMapping(value = "/users")
public class UserController {
    @GetMapping("/users")
    public List<User> getUsers() { … }
}
```

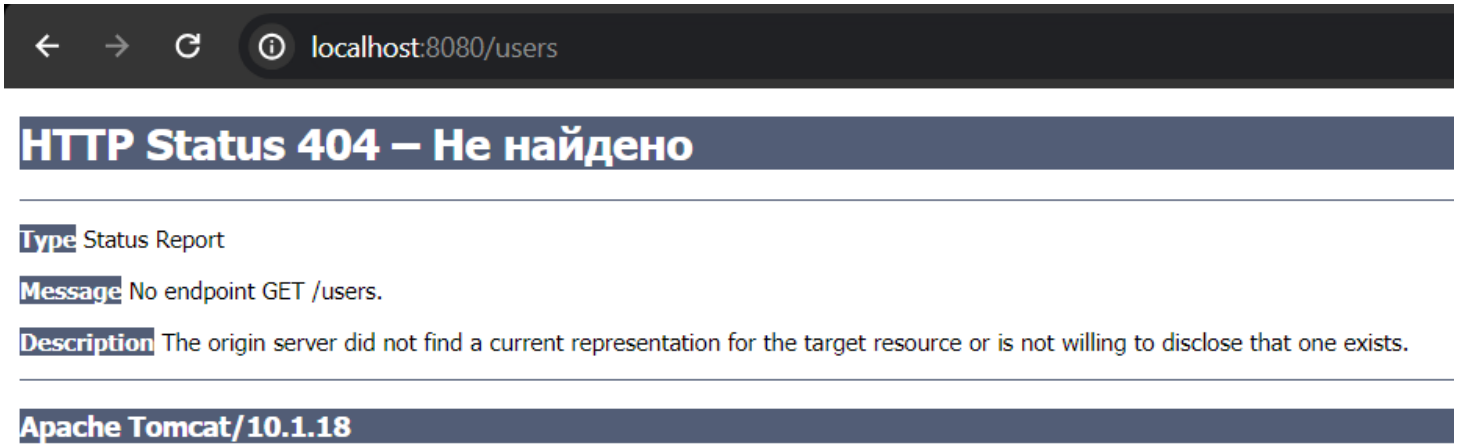


Повторение

Почему не работает запрос?

Из-за дублирования в аннотациях @RequestMapping и @GetMapping. Нужно либо обращаться по пути /users/users, либо убрать @RequestMapping, либо указать @GetMapping(“/”)

```
@Controller
@RequestMapping(value = "/users")
public class UserController {
    @GetMapping("/users")
    public List<User> getUsers() { … }
}
```



Повторение

В чём прикол мема?

TEL-RAN
by Starta Institute

2 ОСНОВНОЙ БЛОК

Введение

TEL-RAN
by Starta Institute

- Начнём с собаки
- Классика жанра по MVC

Проблема

TEL-RAN
by Starta Institute

Как перейти от xml-конфигурации к конфигурации в коде?

Начнём с собаки

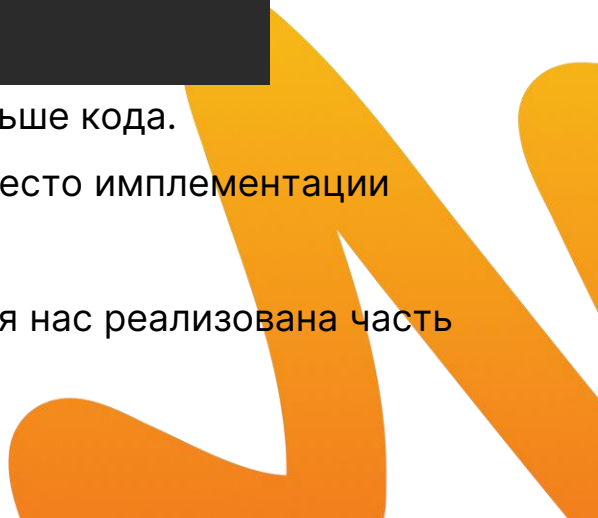
Конфигурация Spring MVC в Java-коде

Начиная с версии 3.0 Spring Framework вместо создания web.xml можно в коде создать класс, реализующий интерфейс *org.springframework.web.WebApplicationInitializer*.

```
public class MyWebAppInitializer implements WebApplicationInitializer {
    @Override
    public void onStartup(ServletContext servletContext) {
        // код, делающий то, что ранее было в конфигурации web.xml
    }
}
```

Такой подход даёт больший контроль, но заставляет писать больше кода.

Поэтому в версии 3.2 команда Spring Framework предложила вместо имплементации интерфейса наследовать абстрактный класс *AbstractAnnotationConfigDispatcherServletInitializer*, в котором для нас реализована часть логики из *WebApplicationInitializer*.



Начнём с собаки

Конфигурация Spring MVC в Java-коде

Скопируем один из ранее созданных проектов и переименуем его.

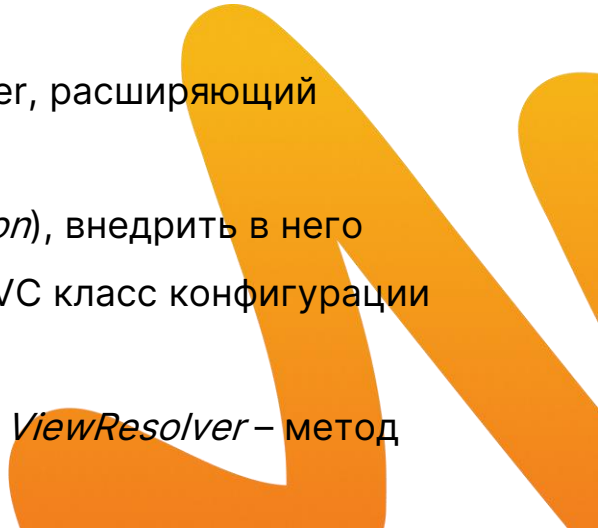
Класс *AbstractAnnotationConfigDispatcherServletInitializer* использует взаимодействие с JEE, поэтому нужно добавить новую зависимость

https://mvnrepository.com/artifact/jakarta.servlet/jakarta.servlet-api

(для версий Spring до 5 - https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api)

Шаги перехода:

1. Создать класс-инициализатор. Например, MyWebAppInitializer, расширяющий *AbstractAnnotationConfigDispatcherServletInitializer*;
2. Создать класс конфигурации контекста Spring (*@Configuration*), внедрить в него ApplicationContext, перенести бины. Для настройки Spring MVC класс конфигурации должен имплементировать интерфейс *WebMvcConfigurer*.
3. Переопределить нужные методы. Для замены стандартного *ViewResolver* – метод *configureViewResolvers.*



Начнём с собаки

Конфигурация Spring MVC в Java-коде

4. Переопределить методы в *MyWebAppInitializer*:

- *getRootConfigClasses* - если требуется задать базовую конфигурацию Spring, указываем её класс. В противном случае возвращаем null;
- *getServletConfigClasses* – указываем класс конфигурации сервлета;
- *getServletMappings* – сопоставляем обрабатываемые пути из URL нашему *DispatcherServlet*;

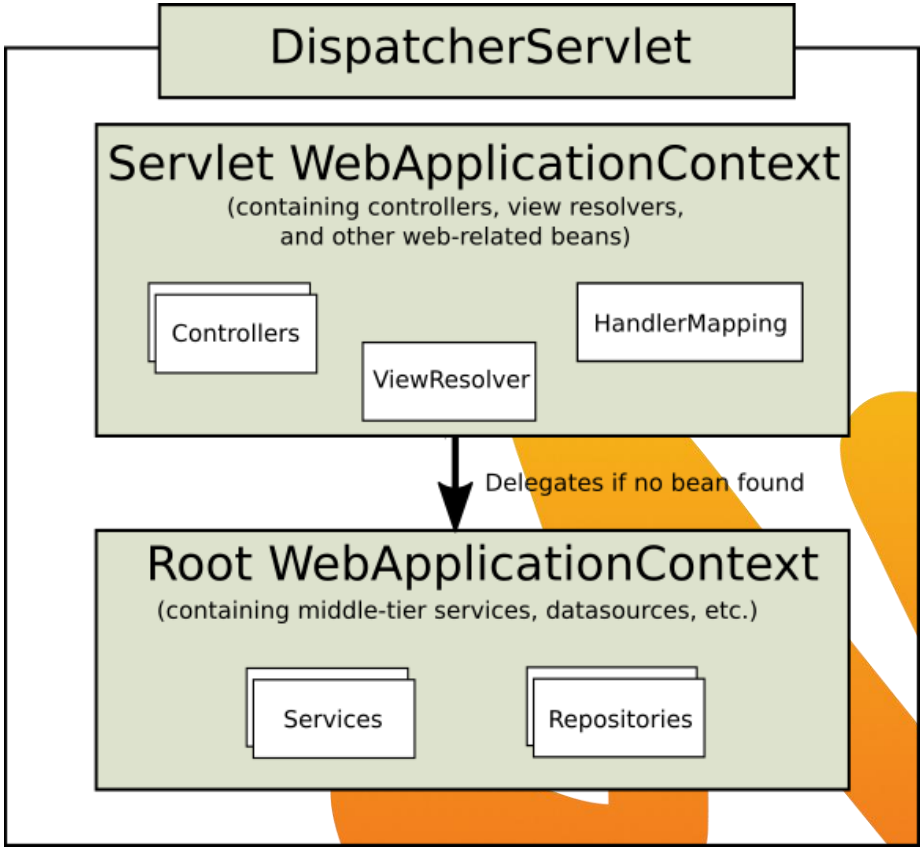
5. Можно удалить xml-файлы;

Начнём с собаки

Связь контекстов

Стоит отдельно отметить, что Spring MVC приложение строится на базовом контексте Spring (*RootWebApplicationContext*) и контексте сервлета (*Servlet WebApplicationContext*). Каждый из контекстов обладает своим набором бинов.

Сначала поиск бинов происходит в контексте сервлета. Если бин не был обнаружен, то поиск производится в базовом контексте приложения.



Начнём с собаки

Конфигурация Spring MVC в Java-коде

Пример конфигурации в Java-коде.

SpringMvcInitializerExample.zip

Задание

TEL-RAN
by Starta Institute

Создайте Spring MVC приложение с конфигурацией в Java-коде. Приложение генерирует случайные данные по запросу:

- дату
- время
- дату и время
- UUID
- email
- имя
- номер телефона

Для упрощения воспользуйтесь библиотекой

https://mvnrepository.com/artifact/com.github.javafaker/javafaker

Проблема

TEL-RAN
by Starta Institute

Классическое MVC-приложение умеет формировать различные представления в зависимости от направленного запроса. Предположим, что нам нужно добавить форму регистрации пользователя и форму отображения его данных в системе (профиль).

Каким образом приложение изменяет шаблоны в соответствии с данными модели для каждого пользователя?

Классика жанра по MVC

Управление представлением

Для управления представлением, как ни странно, используется класс **Model**. Объект *Model* используется для передачи данных между контроллерами и представлениями в *Spring MVC*. Это интерфейс, который предоставляет API для хранения атрибутов модели и их последующей передачи в представление для отображения.

По сути *Model* представляет из себя хранилище пар ключ-значение, где ключ – имя атрибута. По ключу можно получить значение в представлении (*html*-страницах).

Spring может инжектировать бин *Model* в любой из методов контроллера.

```
@GetMapping("/profile/{userId}")
public String getProfileView(@PathVariable("userId") long userId, Model model) {
    User user = userService.getUser(userId);
    model.addAttribute("id", user.id());
    return "user_profile";
}
```

```
<html lang="en">
    <body>
        <h2>Account</h2>
            <p>id: <span th:text = "${id}"></span></p>
    </body>
</html>
```

Классика жанра по MVC

Управление представлением

Аннотация **@ModelAttribute** используется для автоматического связывания входящих параметров запроса с объектами модели или для добавления общих атрибутов к модели перед выполнением метода контроллера.

Аннотация используется над методом или перед аргументом, который принимает метод.

Аннотация над методом:

```
// Показывает, что метод подготавливает атрибуты модели.
// Выполняется до начала работы контроллера
@ModelAttribute
public void addAttributes(Model model) {
    model.addAttribute("msg", “Hello world!");
}
```

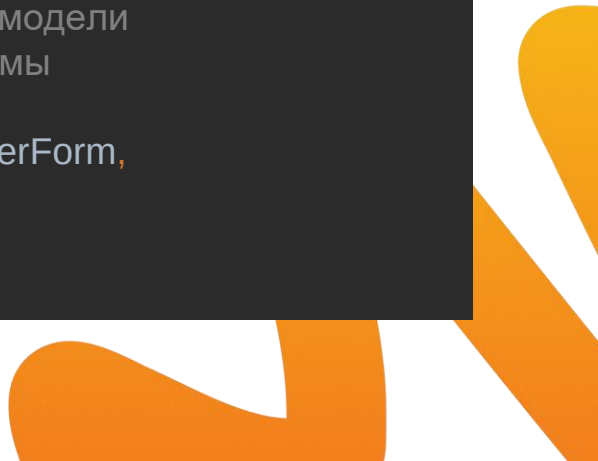
Классика жанра по MVC

Управление представлением

Аннотация перед аргументом:

```
@GetMapping("register")
public String getRegisterForm(Model model) {
    // подготавливаем болванку для заполнения объекта данными формы, кладём в модель
    model.addAttribute("userForm", new UserFormData());
    return "user_registration";
}

// Аннотация @ModelAttribute говорит Spring внедрить атрибут из модели
// При внедрении поля объекта будут заполнены данными из формы
@PostMapping("register")
public String register(@ModelAttribute("userForm") UserFormData userForm,
BindingResult result,
Model model) {
}
```



Классика жанра по MVC

Синтаксис Thymeleaf в html

Для обращения к переменным *Thymeleaf* используется следующий синтаксис:

${x} вернет переменную x, сохраненную в контексте *Thymeleaf* (в *Model*), или как атрибут запроса.

${param.x} вернет параметр запроса с именем x (который может быть многозначным).

${session.x} вернет атрибут сессии с именем x.

${application.x} вернет атрибут контекста сервлета с именем x.

```
<body>
    <p th:utext="#{home.welcome}">Welcome to our grocery store!</p
    <p>Today is: <span th:text="${today}">13 February 2011</span></p>
</body>
```

Код содержит выражение на языке **OGNL** (*Object-Graph Navigation Language*), который будет выполняться на контекстных переменных из *Model*. При использовании *Thymeleaf* в *Spring MVC* язык *OGNL* заменяют на **SpEL** (*Spring Expression Language*), который во многом похож на *OGNL*.

Классика жанра по MVC

Синтаксис Thymeleaf в html

Атрибут "th:text" заменяет тело тега. При использовании тегов html внутри значения атрибута следует использовать "th:utext".

Thymeleaf поддерживает:

- Простые выражения, токены и литералы ('one text', 34, 3.0, null, true, false, _)
- Операции со строками (соединение, подстроки)
- Переменные: ${...}, выбранная переменная *{...}, сообщение: #{...}, ссылка URL: @{...}, фрагмент: ~{...}
- Арифметические, логические, реактивные операции (+, -, *, /, %, and, or, ! или not, >, <, >=, <=, ==, !=
- Условные: If-then: (if)? (then), If-then-else: (if)? (then): (else), (value) ?: (defaultvalue)

Выражения могут комбинироваться и вкладываться:

'User is of type ' + (${user.isAdmin()} ? 'Administrator' : (${user.type} ?: 'Unknown'))

Подробнее здесь: https://habr.com/ru/articles/350870/

Классика жанра по MVC

Управление представлением

Для получения данных из контекста можно использовать следующие обращения:

- **#ctx**: контекст.
- **#vars**: переменные контекста.
- **#locale**: локаль контекста.
- **#request**: (только в Web Contexts) объект HttpServletRequest.
- **#response**: (только в Web Contexts) объект HttpServletResponse.
- **#session**: (только в Web Contexts) объект HttpSession.
- **#servletContext**: (только в Web Contexts) объект ServletContext.

Документация Thymeleaf https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html

Классика жанра по MVC

Пример управления представлениями

SpringMvcModelToViewExample.zip

Задание

TEL-RAN
by Starta Institute

Создайте Spring MVC приложение с конфигурацией в Java-коде. Приложение является онлайн-калькулятором: получает три параметра в URL – оператор, первое число, второе число. В ответ приложение отправляет представление с сообщением типа:

7 минус 8 равно -1

TEL-RAN
by Starta Institute

3 Домашнее задание

Домашнее задание

TEL-RAN
by Starta Institute

Разработайте приложение для отображения и обработки простой формы обратной связи о товаре (оценка от 0 до 5, сообщение с отзывом). Приложение должно предоставлять представление формы обратной связи, а также сохранять отправленные пользователем данные. У товара есть идентификатор и название. Создайте метод, который по id товара будет возвращать представление вида «Средняя оценка товара Расчёска: 4.35 из 5 на основании 132 отзывов».

ЗАКЛЮЧЕНИЕ

TEL-RAN
by Starta Institute