Введение в Spring MVC

№ урока: 12 **Курс:** Основы Java EE

Средства обучения: InteliJ Idea

Обзор, цель и назначение урока

Знакомство с Spring MVC. Рассмотрение шаблона MVC.

Изучив материал данного занятия, учащийся сможет:

- Создать первое приложение с использованием Spring MVC.
- Обработать форму с использованием Spring MVC.

Содержание урока

1. Знакомство с шаблоном MVC.
2. Рассмотрение контроллера.
3. Структура проекта Spring MVC.
4. Добавление CSS в Spring MVC.
5. Обработка формы в Spring MVC.

Резюме

- **Model-View-Controller** (**MVC**, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.
  Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
  Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
  Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.
- Основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:
  К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;
  Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой контроллер;
  Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики(модели), вообще не будут осведомлены о том, какое представление будет использоваться.
- **Модель** предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность. Модель не зависит от представления (не знает, как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем) просто предоставляя доступ к данным и управлению ими.
  Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом может быть встроено уведомление «наблюдателей».

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Модель, за счёт независимости от визуального представления, может иметь несколько различных представлений для одной «модели».

- **Представление** отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введённые данные пользователя.
  Представление может влиять на состояние модели, сообщая модели об этом.
- **Контроллер** обеспечивает «связи» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.
- **Функциональные возможности и расхождения**
  Поскольку MVC не имеет строгой реализации, то реализован он может быть по-разному. Нет общепринятого определения, где должна располагаться бизнес-логика. Она может находиться как в контроллере, так и в модели. В последнем случае, модель будет содержать все бизнес-объекты со всеми данными и функциями.
  Некоторые фреймворки жестко задают где должна располагаться бизнес-логика, другие не имеют таких правил.
  Также не указано, где должна находиться проверка введённых пользователем данных. Простая валидация может встречаться даже в представлении, но чаще они встречаются в контроллере или модели.
  Интернационализация и форматирование данных также не имеет четких указаний по расположению.
- Spring имеет собственную MVC-платформу веб-приложений, которая не была первоначально запланирована. Разработчики Spring решили написать её как реакцию на то, что они восприняли как неудачность конструкции (тогда) популярного Apache Struts, а также других доступных веб-фреймворков. В частности, по их мнению, было недостаточным разделение между слоями представления и обработки запросов, а также между слоем обработки запросов и моделью.[4]
  Класс DispatcherServlet является основным контроллером фреймворка и отвечает за делегирование управления различным интерфейсам, на всех этапах выполнения HTTP-запроса. Об этих интерфейсах следует сказать более подробно.
  Как и Struts, Spring MVC является фреймворком, ориентированным на запросы. В нем определены стратегические интерфейсы для всех функций современной запросно-ориентированной системы. Цель каждого интерфейса — быть простым и ясным, чтобы пользователям было легко его заново имплементировать, если они того пожелают. MVC прокладывает путь к более чистому front-end-коду. Все интерфейсы тесно связаны с Servlet API. Эта связь рассматривается некоторыми как неспособность разработчиков Spring предложить для веб-приложений абстракцию более высокого уровня. Однако эта связь оставляет особенности Servlet API доступными для разработчиков, облегчая все же работу с ним. Наиболее важные интерфейсы, определенные Spring MVC, перечислены ниже:
  HandlerMapping: выбор класса и его метода, которые должны обработать данный входящий запрос на основе любого внутреннего или внешнего для этого запроса атрибута или состояния.
  HandlerAdapter: вызов и выполнение выбранного метода обработки входящего запроса.
  Controller: включен между Моделью (Model) и Представлением (View). Управляет процессом преобразования входящих запросов в адекватные ответы. Действует как ворота, направляющие всю поступающую информацию. Переключает поток информации из модели в представление и обратно.
  View: ответственно за возвращение ответа клиенту в виде текстов и изображений. Некоторые запросы могут идти прямо во View, не заходя в Model; другие проходят через все три слоя.
  ViewResolver: выбор, какое именно View должно быть показано клиенту.
  HandlerInterceptor: перехват входящих запросов. Сопоставим, но не эквивалентен сервлет-фильтрам (использование не является обязательным и не контролируется DispatcherServlet-ом).
  LocaleResolver: получение и, возможно, сохранение локальных настроек (язык, страна, часовой пояс) пользователя.
  MultipartResolver: обеспечивает Upload — загрузку на сервер локальных файлов клиента.
  Spring MVC предоставляет разработчику следующие возможности:
  Ясное и прозрачное разделение между слоями в MVC и запросах.
  Стратегия интерфейсов — каждый интерфейс делает только свою часть работы.
  Интерфейс всегда может быть заменен альтернативной реализацией.
  Интерфейсы тесно связаны с Servlet API.

CyberBionic systematics
CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Высокий уровень абстракции для веб-приложений.
В веб-приложениях можно использовать различные части Spring, а не только Spring MVC.

Закрепление материала

- Что такое MVC?
- Как работает контроллер в Spring MVC?
- Как добавить CSS в проект Spring MVC?

Дополнительное задание

Задание
Добавить на форму checkbox «Администратор» и передать в объект User.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.
Прочитать документацию pdf разделы 15.1, 15.2

Задание 2
Выбрать любую форму со стилем.
https://designmodo.com/css3-forms/
Подключить стиль и форму на станицу

Рекомендуемые ресурсы

Spring docs pdf
https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/pdf/spring-framework-reference.pdf

Front Controller
http://www.tutorialspoint.com/design_pattern/front_controller_pattern.htm
http://www.oracle.com/technetwork/java/frontcontroller-135648.html

CyberBionic systematics
CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com