Spring Start Here

Chapter-7:
Understanding Spring Boot and Spring MVC

Upcode Software
Engineer Team

-2024-

КОНТЕНТ

1. Что такое веб-приложение (1/3)

- **Определение веб-приложения:** Веб-приложение это программа, доступная через веб-браузер, которая выполняет определенные функции или предоставляет услуги пользователям.

- **Примеры веб-приложений**: Включают онлайн-магазины, социальные сети, электронные почтовые клиенты и многое другое.

- **Удобство и доступность**: Веб-приложения удобны, потому что не требуют установки на устройство пользователя и доступны с любого устройства с доступом к интернету.

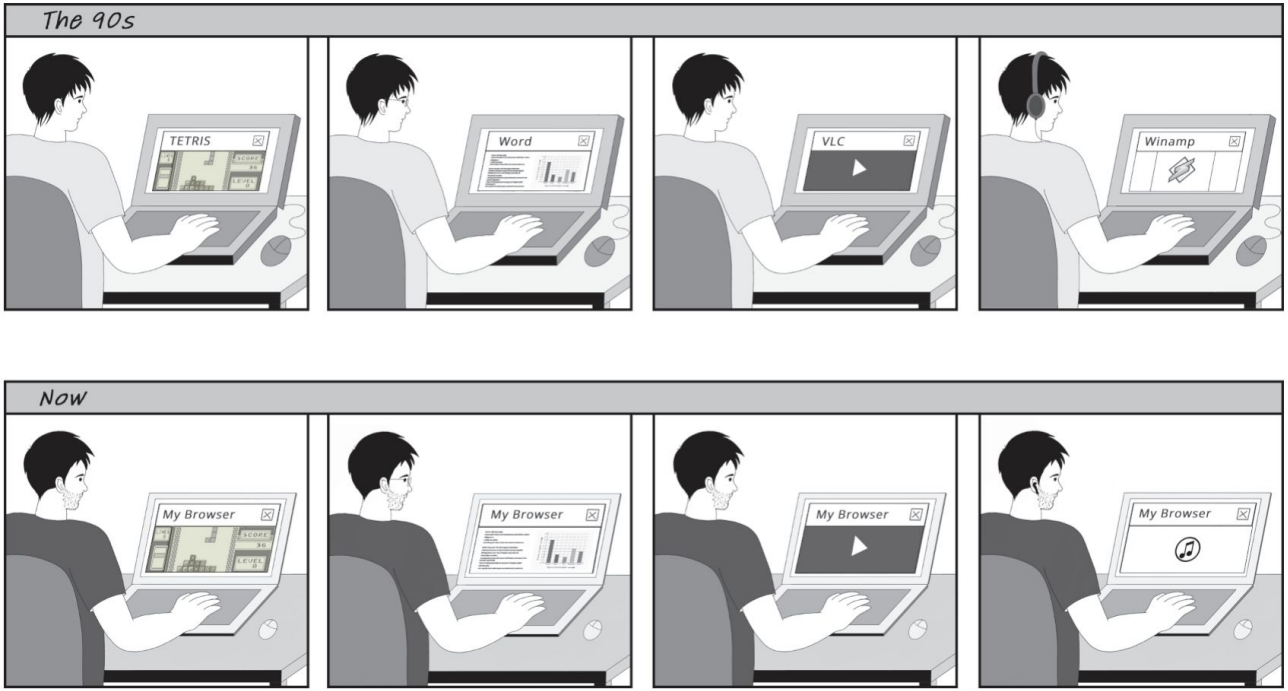
1. Что такое веб-приложение (2/3)

- **Эволюция от десктопных приложений к веб-приложениям**: В прошлом десктопные программы были основным способом доступа к приложениям, но сегодня большинство приложений доступны через веб-браузер.

- **Преимущества веб-приложений:** Гибкость, масштабируемость, обновления без установки, доступность с разных устройств.

- **Заключение**: Веб-приложения сегодня стали стандартом, обеспечивая пользователей удобством, доступностью и функциональностью на любом устройстве. Например с планшета или телефона.

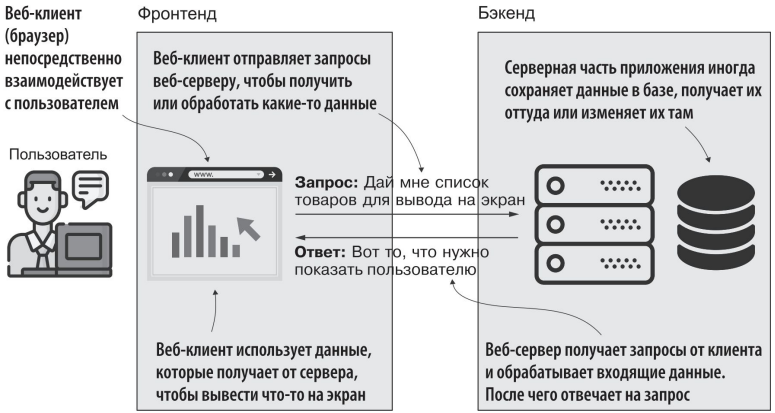
1. Что такое веб-приложение(3/3



2. Основные сведение о веб-приложенях(1/2)

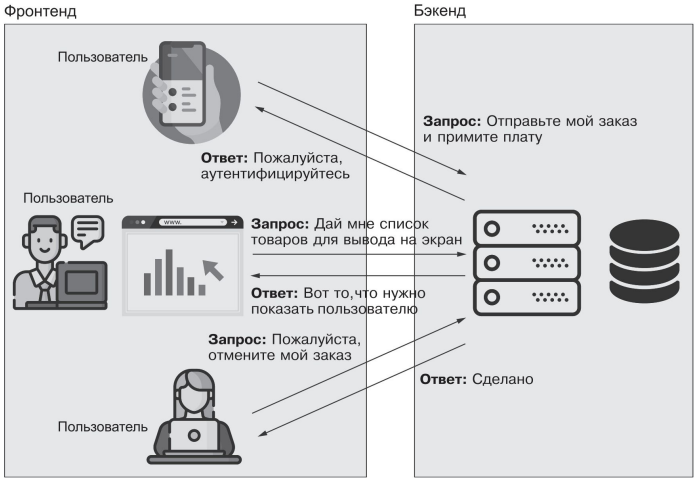
Любое веб- приложение состоит из двух частей, таких как.

- Клиентская часть, или фронтенд, это интерфейс приложения, с которым взаимодействует пользователь через веб-браузер. Браузер отправляет запросы на сервер, получает ответы и обеспечивает пользователю способ взаимодействия с приложением.
- **Бэкенд**, это часть веб-приложения, которая принимает запросы от клиента, обрабатывает их, иногда сохраняет данные и отправляет обратно клиенту ответы.



2. Основные сведение о веб-приложенях(2/2)

- Бэкенд веб-приложения обслуживает несколько клиентов одновременно на конкурентной основе. Это означает, что множество пользователей могут одновременно использовать одно приложение на разных платформах, таких как компьютеры, телефоны, планшеты и другие.



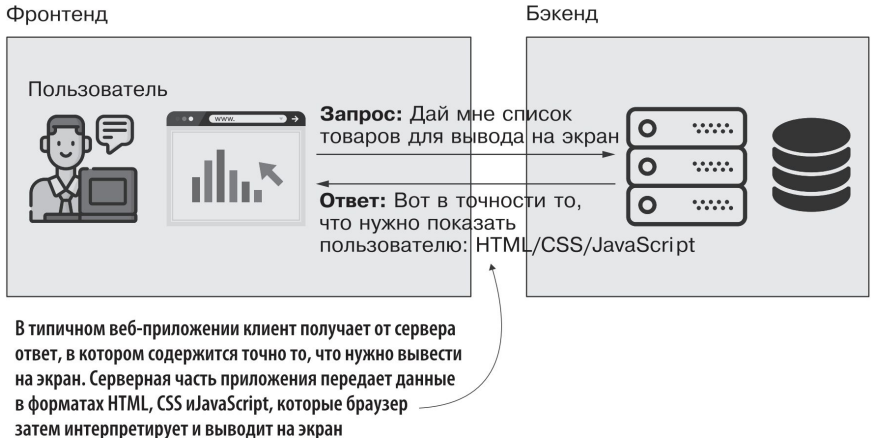
3. Способы реализации веб-приложений на основе Spring(1/3)

Веб-приложение по способу функционирования бывают разные:

- **Приложения, в которых бэкенд в ответ на запрос клиента дает полностью готовое представление.**
  - Этот подход называется серверным рендерингом. Бэкенд обрабатывает запросы клиента и формирует полностью готовое представление данных, которые сразу отображаются в браузере пользователя. Этот метод позволяет браузеру получать данные от бэкенда и моментально выводить их на экран.
- **Приложения с разделением обязанностей между фронтендом и бэкендом.**
  - Этот подход известен как клиентское рендеринг, где бэкенд предоставляет только первичные данные. Браузер получает эти данные от сервера и запускает специальное клиентское приложение для обработки и отображения информации на экране.

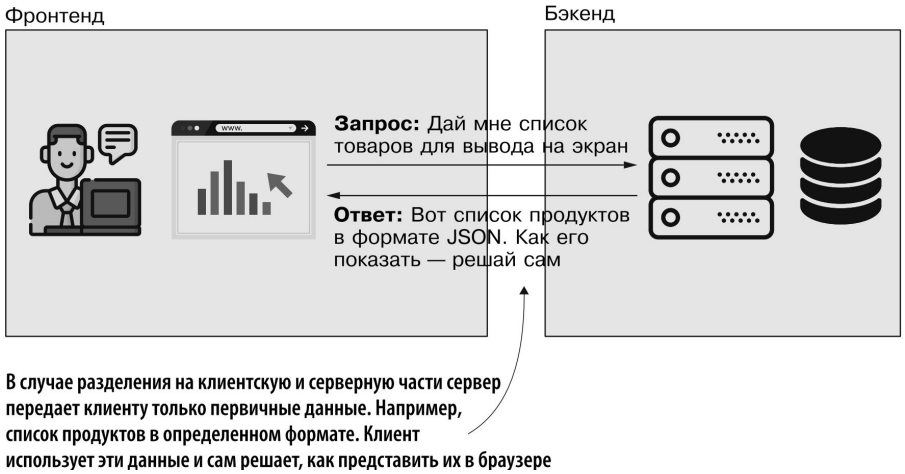
3. Способы реализации веб-приложений на основе Spring(2/3)

- Первый вариант известен как серверный рендеринг, где вся логика приложения выполняется на стороне бэкенда. Бэкенд получает запросы от браузера, обрабатывает их и возвращает информацию для отображения на экране в форматах, которые браузер может интерпретировать, такие как HTML, CSS и JavaScript.



2 Способы реализации веб-приложений на основе Spring(3/3)

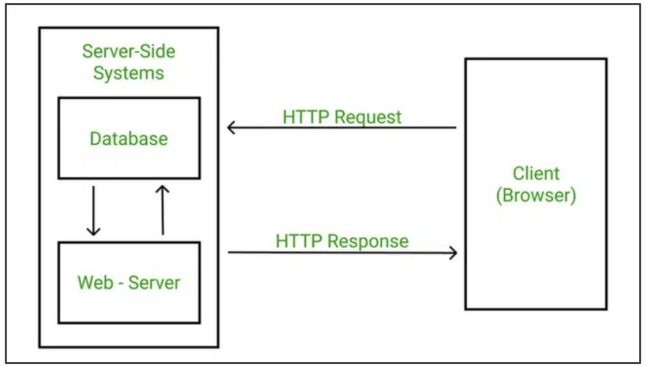
- Второй подход известен как клиент-серверное разделение обязанностей. В нём бэкенд передает браузеру только первичные данные, а клиентское приложение, загружаемое с сервера, интерпретирует эти данные и решает, как их отобразить. В результате серверная часть становится более отвязанной от представления данных на экране.



3. Использование контейнера сервлетов в веб разработке(1/3)

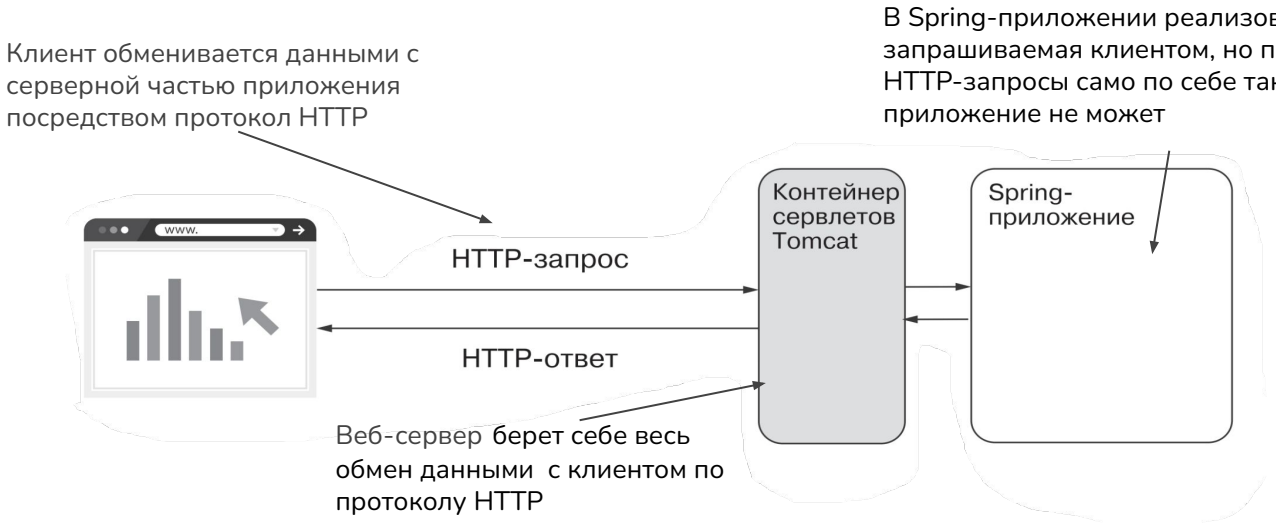
Важность основ HTTP(Hypertext Transfer Protocol) в веб-разработке

- Понимание основ HTTP необходимо для успешной коммуникации между клиентом и сервером в веб-приложениях.
- В приложении представлена необходимая информация о HTTP для эффективной работы с веб-разработкой на основе Spring.
- Лучше использовать существующие компоненты для поддержки HTTP в приложении, чем писать свой низкоуровневый функционал.



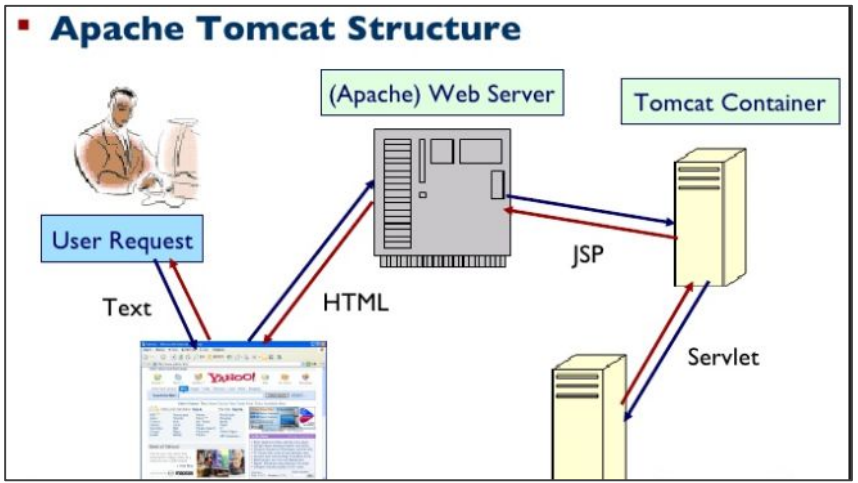
3. Использование контейнера сервлетов в веб разработке(2/3)

- Для Java-приложения необходим контейнер сервлетов, который обеспечивает обработку HTTP-запросов и ответов.
- Контейнер сервлетов, такой как Tomcat, переводит запросы HTTP для Java-приложения. Использование контейнера сервлетов избавляет от необходимости реализации коммуникационного уровня в приложении.



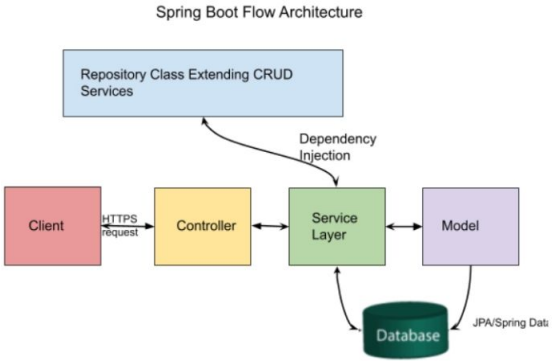
3. Использование контейнера сервлетов в веб разработке(3/3)

- Контейнер сервлетов (такой как Tomcat) умеет ***“умеет говорит на языке HTTP”***
- Он переводит HTTP-запрос для Spring-приложения, а в ответ приложения - в HTTP-ответ.
- **Сервлет** - это просто Java-объект.



3. Магия Spring-Boot

- Ранее для создания веб-приложения Spring нужно было вручную конфигурировать контейнер сервлетов, создавать экземпляр сервлета и настраивать его для обработки запросов Tomcat. Это было трудоемко и неудобно.
- **Автоматическая конфигурация:** значительно упрощает создание Spring-приложений, позволяя сосредоточиться на бизнес-логике.
- **Удобство для микросервисов:** идеально подходит для сервисно-ориентированных архитектур и микросервисов, избавляя от необходимости писать большое количество конфигурационного кода.

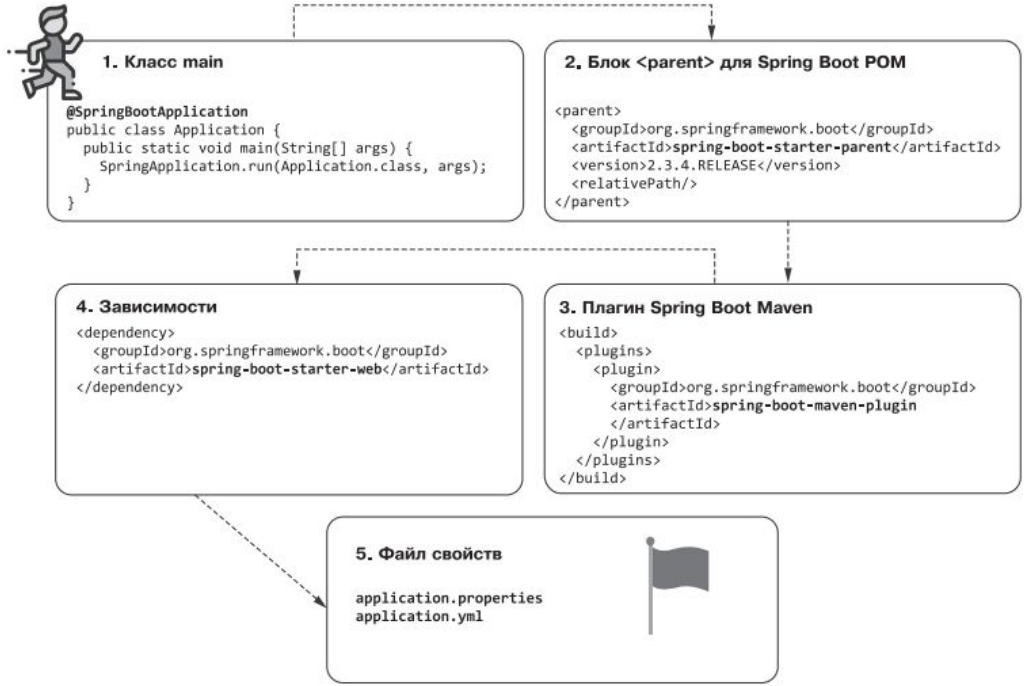


4. Создание проекта Spring Boot с помощью сервиса инициализации проекта(1/6)

После создании проекта Spring Initializr вписывает в конфигурацию проекта Maven

- класс **main** Spring-приложения;
- блок <**parent**> для Spring Boot POM;
- зависимости
- плагин Spring Boot Maven;
- файл свойств.

4. Создание проекта Spring Boot с помощью сервиса инициализации проекта(2/6)



4. Создание проекта Spring Boot с помощью сервиса инициализации проекта(3/6)

Блок <parent> для Spring Boot Maven

- Управление версиями зависимостей: Совместимость версий: родительский узел сообщает о совместимых версиях зависимостей.
- Автоматический выбор: вместо ручного указания версий, Spring Boot автоматически подбирает нужные версии для предотвращения несовместимостей.

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.4.RELEASE</version>
    <relativePath/>
</parent>
```

4. Создание проекта Spring Boot с помощью сервиса инициализации проекта(4/6)

- Плагин **Spring Boot Maven**, конфигурация которого была настроена start.spring.io при создании проекта и находится в pom.xml.
- Ниже представлено объявление плагина, обычно размещаемое в конце файла, между тегами <build><plugins> и </build></plugins>.

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

4. Создание проекта Spring Boot с помощью сервиса инициализации проекта(5/6)

- В **pom.xml** вы найдете зависимость, добавленную при создании проекта на **http://start.spring.io** — Spring Web. Она подключается следующим образом:

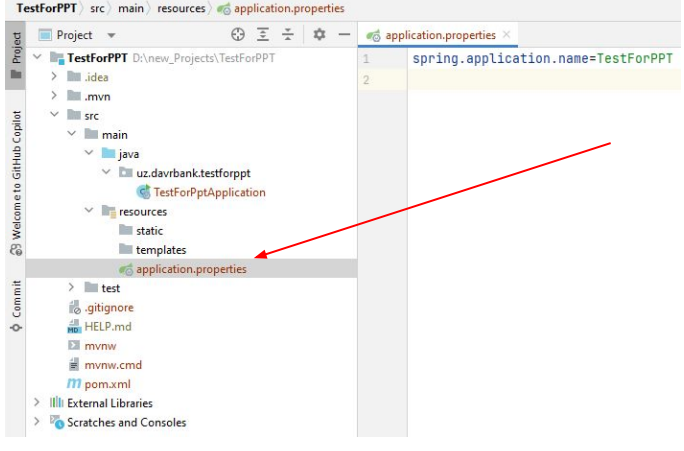
- Версию указывать не нужно, так как Spring Boot автоматически выбирает нужную версию. Это происходит благодаря родительскому узлу Spring Boot в pom.xml.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

4. Создание проекта Spring Boot с помощью сервиса инициализации проекта(6/6)

- И последний важный элемент, который Spring Initializr добавляет в проект, — это файл **application.properties**. Данный файл находится в папке ресурсов проекта Maven.

- Вначале он пуст, и в нашем примере таким и останется. Впоследствии мы поговорим об использовании этого файла для настройки параметров, необходимых в процессе выполнения приложения.



4. Упрощенное управление зависимостями с помощью диспетчеров зависимостей(1/2)

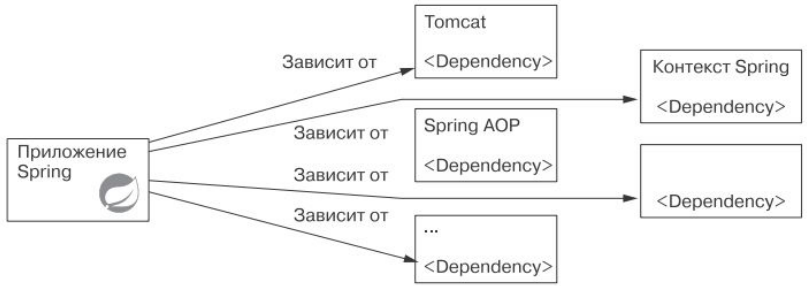
- Диспетчеры зависимостей (или "**стартеры**") являются одной из ключевых особенностей Spring Boot, позволяя значительно ускорить и упростить настройку проекта. Эти стартеры включают в себя наборы зависимостей, которые решают конкретные задачи, и помогают избежать необходимости вручную добавлять каждую зависимость отдельно.
- В файле **pom.xml** (если вы используете Maven) диспетчер зависимостей выглядит как обычная зависимость. Обычно имя диспетчера начинается с **spring-boot-starter**-, после чего следует описание функциональности, которую он добавляет в приложение.
- Этот пример показывает использование диспетчера зависимостей **spring-boot-starter-web**, который добавляет в проект все необходимые зависимости для разработки веб-приложений на базе Spring MVC.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

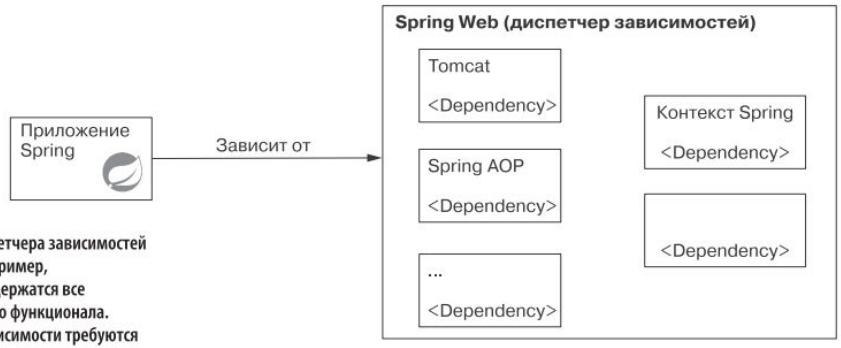
4. Упрощенное управление зависимостями с помощью диспетчеров зависимостей(2/2)

Без диспетчеров зависимостей

Все зависимости объявляются в приложении напрямую. Для реализации определенного функционала разработчик должен знать, какие зависимости ему необходимы и какие версии этих зависимостей являются совместимыми



С диспетчерами зависимостей



В приложении объявляется зависимость от диспетчера зависимостей в соответствии с требуемым функционалом (например, веб-функциями). В диспетчере зависимостей содержатся все зависимости, необходимые для реализации этого функционала. Разработчику не нужно знать, какие именно зависимости требуются и какие их версии являются совместимыми

7. Автоматическая конфигурация по соглашению на основе зависимостей

- Принцип «соглашения важнее конфигурации» в Spring Boot минимизирует ручную настройку за счет разумных значений по умолчанию.
- Автоматическая конфигурация — ключевая особенность Spring Boot.
- При запуске приложения автоматически поднимается сервер Tomcat на порту 8080, что позволяет быстро запустить и протестировать приложение.

```
Tomcat started on port(s): 8080 (http) with context path ''
Started Main in 1.684 seconds (JVM running for 2.306)
```

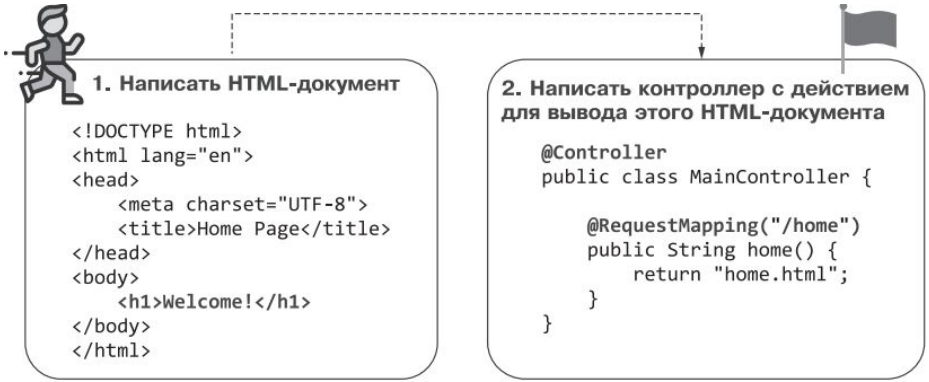
Spring Boot настроил Tomcat и по умолчанию запускает его через порт 8080

8. Реализация приложения с помощью Spring MVC(1/5)

Чтобы создать первую страницу веб-приложения на Spring Boot, выполните следующие шаги:

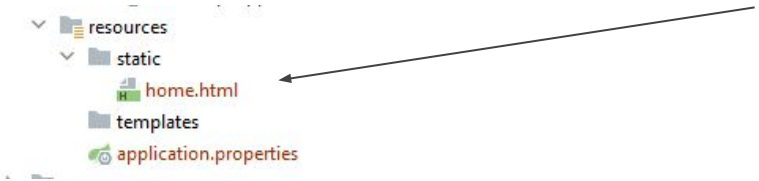
1. Напишите HTML-документ, который будет отображаться в браузере.
2. Создайте контроллер для этой страницы с соответствующим действием.

Это добавит статическое веб-содержимое в ваше приложение и превратит его в полноценное веб-приложение.



8. Реализация приложения с помощью Spring MVC(2/5)

- В проекте создайте статическую веб-страницу, поместив HTML-документ с коротким заголовком в папку **resources/static** проекта Maven. Эта папка используется по умолчанию для отображения страниц в Spring Boot.




```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Home Page</title>
</head>
<body>
    <h1>Welcome!</h1>
</body>
</html>
```

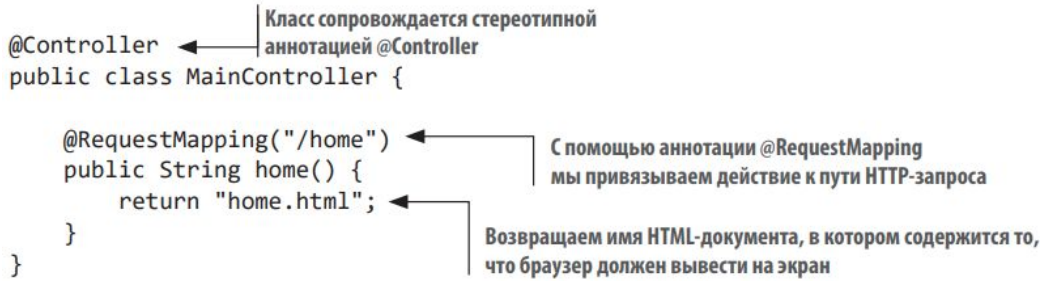
В обычном HTML-документе здесь размещается текст заголовка

8. Реализация приложения с помощью Spring MVC(3/5)

Для создания контроллера, связывающего HTTP-запрос со статической страницей, выполните следующие шаги:

1. Напишите контроллер и отметьте его аннотацией **@Controller**, чтобы Spring создал и управлял его бином.
2. В контроллере создайте метод, аннотированный **@RequestMapping("/home")**, который возвращает строку с именем HTML-документа, например "**home.html**".

Контроллер будет возвращать содержимое `home.html` в ответ на запросы по пути /home.

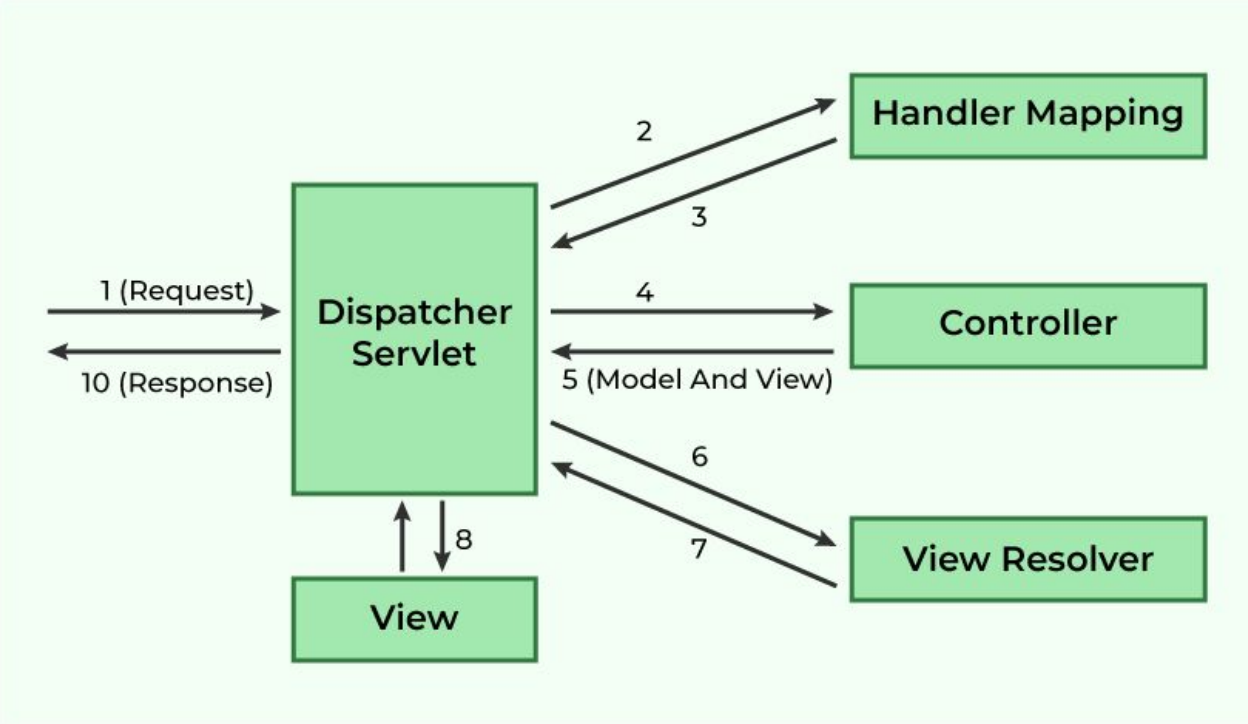


8. Реализация приложения с помощью Spring MVC(3/5)

8. Реализация приложения с помощью Spring MVC(4/5)

1. Клиент отправляет HTTP-запрос.
2. Tomcat получает запрос и передает его сервлету-диспетчеру (**dispatcher servlet**) Spring MVC.
3. Сервлет-диспетчер управляет запросами, определяя контроллер, который должен обработать запрос.
4. Сервлет-диспетчер использует компонент ***handler mapping*** для нахождения соответствующего действия контроллера через аннотацию @RequestMapping.
5. Если действие найдено, оно вызывается, и контроллер возвращает имя представления (view). Если нет, возвращается статус 404.
6. Сервлет-диспетчер делегирует поиск содержимого представления компоненту view resolver.
7. Сервлет-диспетчер отправляет сгенерированное представление клиенту в виде HTTP-ответа.

8. Реализация приложения с помощью Spring MVC(4/5)



8. Реализация приложения с помощью Spring MVC(5/5)

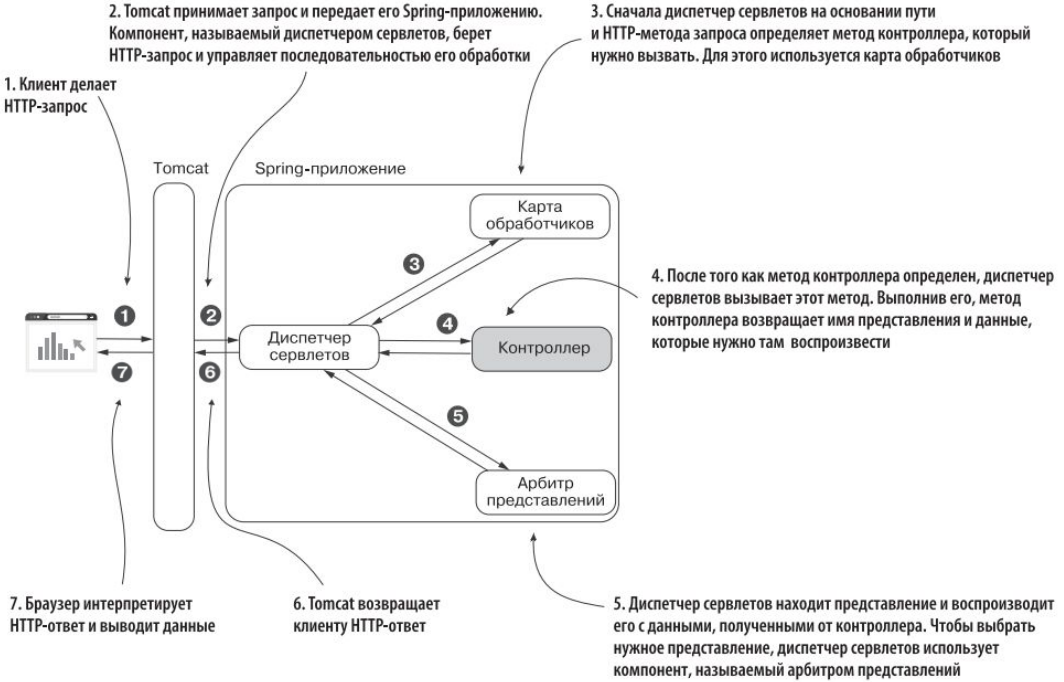


Рис. 7.18. Архитектура Spring MVC. На диаграмме показаны основные компоненты Spring MVC. Именно эти компоненты и способ их взаимодействия определяют поведение веб-приложения. Единственный из них, который нам нужно реализовать, — это контроллер (покрашен в серый цвет). Остальные компоненты создает Spring Boot

8. Заключение(1/2)

- Веб-приложение— это приложение, взаимодействующее с пользователем через веб-браузер, имеющее клиентскую (фронтенд) и серверную (бэкенд) части
- Фронтенд отправляет запросы к серверной части, а бэкенд обрабатывает эти запросы и возвращает ответы.
- Spring Boot — это проект экосистемы Spring, который упрощает создание веб-приложений, предоставляя настройки по умолчанию и упрощая конфигурацию зависимостей. Основные преимущества Spring Boot:
    - 1. Автоматическая конфигурация: минимизирует необходимость ручной настройки.
    - 2. Диспетчер зависимостей: обеспечивает совместимость версий зависимостей.
    - 3. Контейнер сервлетов: автоматически настраивает, например, Tomcat, для обработки HTTP-запросов и ответов.

8. Заключение(2/2)

- Spring MVC: структура классов для обработки HTTP-запросов.
- Контроллеры в Spring MVC описываются с помощью аннотаций:
- @Controller: отмечает класс как контроллер.
- @RequestMapping: связывает методы контроллера с определенными HTTP-запросами.
- Spring Boot и Spring MVC вместе значительно упрощают создание и конфигурирование веб-приложений.

REFERENCE

1: Spring Start Here

Resources


Spring
Start Here
LEARN WHAT YOU NEED
AND LEARN IT WELL
LAURENȚIU SPILCĂ
FOREWORD BY VICTOR RENTEA
MANNING

Thank you!

Presented by

Asadbek Quronboyev

(asadbek9805@gmail.com)