УДК 004.4

ИСПОЛЬЗОВАНИЕ SPRING FRAMEWORK ПРИ СОЗДАНИИ JAVA ВЕБ-ПРИЛОЖЕНИЯ

© 2024 В. В. Гордиенко[1], Ф. П. Лебедчук[2]

[1] кандидат технических наук, доцент кафедры информационной безопасности
e-mail: vika.gordienko.1973@mail.ru,
[2] студент 5 курса
e-mail: melancholiacksu@mail.ru

Курский государственный университет

В статье рассматриваются возможности Spring Framework в разработке веб-приложений на языке программирования Java. Анализируются основные компоненты и модули Spring с акцентом на Spring Boot и внедрение зависимостей (Dependency Injection, DI), их роль в упрощении разработки, а также преимущества, которые они предоставляют с помощью готовых инструментов и шаблонов. Авторы последовательно раскрывают тему: от введения в концепции Spring Framework через особенности Spring Boot до демонстрации внедрения зависимостей на конкретных примерах.

Ключевые слова: Java, Spring Framework, Spring Boot, Dependencies, Maven.

USING THE SPRING FRAMEWORK WHEN CREATING JAVA WEB APPLICATIONS

© 2024 V. V. Gordienko[1], F. P. Lebedchuk[2]

[1] Candidate of Engineering Sciences, Associate Professor,
Department of Information Security, KSU
e-mail: vika.gordienko.1973@mail.ru,
[2] 5th year student
e-mail: melancholiacksu@mail.ru

Kursk State University

The article discusses the possibilities of the Spring Framework in the development of web applications in the Java programming language. The main components and modules of Spring are analyzed, with an emphasis on Spring Boot and Dependency Injection (DI), their role in simplifying development, as well as the advantages they provide using ready-made tools and templates. The authors consistently reveal the topic: from an introduction to the concepts of the Spring Framework, through the features of Spring Boot, to demonstrating the implementation of dependencies using specific examples.

Keywords: Java, Spring Framework, Spring Boot, Dependencies, Maven.

Современные веб-приложения требуют высокой гибкости, масштабируемости и быстроты разработки. В условиях цифровой трансформации знание новейших инструментов программирования становится важным навыком для каждого профессионала в области IT. Spring Framework широко применяется как в учебных проектах, так и в крупных корпоративных системах. Благодаря своей гибкости и наличию модуля Spring Boot этот фреймворк идеально подходит для обучения: он позволяет сосредоточиться на освоении основных принципов программирования, минимизируя время, затрачиваемое на сложные настройки.

Spring делает программирование на Java более быстрым, простым и безопасным для всех. Стремление Spring к скорости, простоте и производительности сделало его самым популярным Java-фреймворком в мире [1].

В отличие от других платформ Spring фокусируется на нескольких областях функционирования приложений и предоставляет для них широкий спектр дополнительных функций.

Одной из основных особенностей Spring Framework является использование паттерна Dependency Injection (DI, внедрение зависимостей). DI помогает намного проще реализовывать необходимую приложениям функциональность, а также разрабатывать слабо связанные классы, делая их более универсальными.

Если Spring Framework фокусируется на предоставлении гибкости, то Spring Boot стремится сократить длину кода и упростить разработку web-приложения. Используя конфигурацию при помощи аннотаций и стандартного кода, Spring Boot сокращает время, затрачиваемое на разработку приложений. Данная возможность помогает создать автономные приложения с меньшими или почти нулевыми затратами на их конфигурацию.

Автоконфигурация – это особенность Spring Boot. При помощи аннотаций он автоматически настраивает специальные конфигурационные классы.

Теперь рассмотрим некоторые особенности и преимущества упомянутых фреймворков.

Преимущества Spring Framework:

1) Spring Framework может быть задействован на всех архитектурных слоях, применяемых при разработке web-приложений;
2) использует модель POJO при написании классов, а это очень легкая структура;
3) позволяет свободно связывать модули и легко их тестировать;
4) поддерживает декларативное программирование;
5) избавляет от самостоятельного создания фабричных и синглтон-классов;
6) поддерживает различные способы конфигурации;
7) предоставляет сервис уровня middleware.

Преимущества Spring Boot:

1) Spring Boot не требует развертывания war-файлов;
2) создает автономные приложения;
3) помогает напрямую встроить в приложение Tomcat, Jetty или Undertow;
4) не требует XML-конфигурации;
5) направлен на уменьшение объема исходного кода;
6) имеет дополнительную функциональность «из коробки»;
7) простота запуска;
8) простая настройка и управление.

Благодаря таким функциям, как автоконфигурация, Spring Boot избавляет от написания лишнего кода и помогает избежать ненужной настройки.

Spring Framework не только предлагает такие функции, как внедрение зависимостей или обработка транзакций, но также выступает в качестве основы для других фреймворков Spring. Лучшим примером для этого является Spring Boot. Spring Boot использует Spring Framework в качестве своей основы. Он упрощает зависимости Spring и запускает приложения прямо из командной строки. Он также не требует

наличия внешнего контейнера приложений. Spring Boot помогает контролировать компоненты приложения и настраивает их извне [2].

Создание Spring Boot приложения

Исторически у разработчиков приложений Java имелось несколько вариантов утилит сборки проектов. Некоторые со временем стали непопулярными, и не без причин, и теперь сообщество разработчиков сконцентрировалось на двух: Maven и Gradle. Spring Boot поддерживает обе с одинаковым успехом [3].

Maven – это инструмент для сборки, разработанный Apache Software Foundation и используемый в основном для сборки и управления проектами на Java. Он основан на концепции модели объекта проекта (POM) и использует XML-файл для определения сборки проекта и его зависимостей. Maven также предоставляет набор плагинов, которые можно использовать для выполнения различных задач, таких как компиляция кода, запуск тестов и упаковка приложений.

Gradle – это инструмент для сборки, разработанный компанией Gradle Inc. и используемый в основном для сборки и управления проектами на Java. В отличие от Maven, он не основан на конкретной модели объектов проекта, вместо этого он использует более гибкий и выразительный синтаксис для определения сборки проекта. Gradle также предоставляет набор плагинов, которые можно использовать для выполнения различных задач, таких как компиляция кода, запуск тестов и упаковка приложений.

И Maven, и Gradle – это мощные инструменты сборки, которые можно использовать для управления зависимостями, сборки и тестирования кода, а также для упаковки и развёртывания приложений. Maven основан на файле POM и хорошо подходит для небольших проектов, в то время как Gradle более выразителен и настраиваем, что делает его более подходящим для крупных и сложных проектов. Кроме того, более быстрое время сборки Gradle и поддержка инкрементных сборок делают его лучшим выбором для проектов Spring Boot. В конечном счёте выбор между Maven и Gradle будет зависеть от конкретных потребностей и требований конкретного проекта [4].

Отправная точка создания Spring Boot приложения – Spring Initializr. Он позволяет разработчикам быстро и удобно сгенерировать базовую структуру проекта, минимизируя рутинные действия, которые обычно требуются при создании нового проекта вручную. На данном этапе с помощью этого веб-инструмента можно добавить в проект зависимости.

Большинство из них называется зависимостями начального проекта (Starter Project dependencies). Начальная зависимость starter – тип особой дополнительной библиотеки, которая автоматически вставит набор базовых функций, составленный из других проектов экосистемы Spring, в ваше новое приложение Spring Boot [5].

Внедрение зависимостей (Dependency Injection, DI)

Для более глубокого понимания возможностей Spring Framework и его инструментов рассмотрим практические примеры, реализованные в рамках учебного проекта, а также продемонстрируем, как можно эффективно решать задачи, связанные с разработкой веб-приложений.

Внедрение в maven происходит путём добавления в раздел <dependencies> файла pom.xml, автоматически созданного при использовании Spring Initializr.

При создании своего проекта мы использовали следующие зависимости:

1. Spring Boot Starter Web – зависимость для разработки веб-приложений с поддержкой как REST API, так и традиционных серверных приложений,

использующих HTML-страницы. Она включает Spring MVC, встроенный веб-сервер Tomcat и инструменты для обработки данных.

Spring MVC («модель – представление – контроллер») был создан для разделения ответственности относительно данных, их доставки и визуализации в предположении, что представления будут визуализироваться в виде веб-страницы, отображаемой сервером. Для связи всего этого служит аннотация @Controller.

@Controller – стереотип/псевдоним для аннотации @Component, означающий, что при запуске приложения из этого класса образуется компонент Spring, создаваемый и управляемый контейнером инверсии управления (inversion of control, IoC) в приложении. Классы, снабженные аннотацией @Controller, включают объект Model для передачи слою представления данных, соответствующих модели, и отображения (совместно с ViewResolver) приложением конкретного представления с помощью специальной технологии [3].

Необходимо отметить, что в отличие от @RestController в @Controller в аннотации @RequestMapping недоступны версии Put, Patch и Delete, так как html-формы не могут выполнять данные запросы.

```
@Controller
public class BooksController {
   private final BookService bookService;

   public BooksController(BookService bookService){
      this.bookService = bookService;
   }

   @GetMapping("/books")
   public String showBooks(Model model) {
      List<Book> books;
         books = bookService.findAll();
      model.addAttribute("books", books);
      return "books/show";
   }
}
```

В данном примере представлен контроллер для работы с книгами в веб-приложении. BookService – сервис, созданный с помощью Spring Boot Data Jpa и обеспечивающий доступ к бизнес-логике, связанной с книгами. В get-методе мы получаем список всех книг и добавляем его в модель books, чтобы передать его в представление show.html.

2. Boot DevTools предназначена для ускорения процесса разработки и тестирования приложений на Spring Boot. Она предоставляет инструменты для автоматической перезагрузки приложения при изменении кода и улучшает взаимодействие разработчика с приложением.

Ключевые функции: автоматическая перезагрузка приложения при изменении файлов; шаблоны, такие как Thymeleaf, не кешируются, чтобы изменения были видны мгновенно.

3. Spring Boot Data Jpa используется для упрощения работы с базами данных в Java-приложениях. Это модуль Spring, который предоставляет абстракцию над JPA (Java Persistence API), значительно уменьшая количество шаблонного кода, необходимого для реализации операций с базой данных (CRUD).

CRUD – это акроним, обозначающий четыре основные операции, выполняемые с данными в базе данных:

– Create (создание) – добавление новых записей в базу данных;
– Read (чтение) – получение данных из базы;
– Update (обновление) – изменение существующих данных;
– Delete (удаление) – удаление данных из базы.

Данная зависимость существенно уменьшает количество кода, автоматически генерирует стандартные SQL-запросы, а также поддерживает создание кастомных методов и запросов.

Для создания кастомного метода для работы с сущностью book создадим репозиторий. После чего определим метод findByReader, который ищет книги по указанному пользователю.

```
@Repository
public interface BookRepo extends JpaRepository<Book, Long> {
   List<Book> findByReader(User reader);
}
```

Далее добавим метод и привяжем репозиторий к сервису:

```
@Service
public class BookService {
   private final BookRepo bookRepo;
   @Autowired
   public BookService(BookRepo bookRepo) {
      this.bookRepo = bookRepo;
   }
   public List<Book> findByReader(User reader) {
      return bookRepo.findByReader(reader);
   }
```

После этого мы сразу можем использовать метод в контроллере:

```
   @GetMapping("/{id}")
   public String showPerson(@PathVariable("id") long id, Model model) {
      User user = userService.findOneById(id);
      List<Book> books = bookService.findByReader(user);
      model.addAttribute("user", user);
      if (!books.isEmpty()) {
         model.addAttribute("books", books);
      }
      return "users/profile";
   }
```

Контроллер находит пользователя по ID, получает связанные с ним книги и передаёт данные в модель для отображения страницы users/profile.

4. Postgresql – драйвер для работы с соответствующей СУБД.

Для подключения БД к проекту необходимо скачать и установить PostgreSQL и платформу для администрирования и настройки СУБД pgAdmin с официального сайта, а также создать файл application.properties, в котором указываются данные БД.

```
spring.datasource.url=jdbc:postgresql://localhost/your_database
spring.datasource.username=your_username
spring.datasource.password=your_password
spring.jpa.generate-dll=true
```

5. Lombok – вспомогательная библиотека Java, которая позволяет значительно уменьшить количество шаблонного кода за счет использования аннотаций, таких как геттеры, сеттеры, конструкторы, toString, equals и др., во время компиляции.

```
@Entity
@Table(name = "book")
@Getter
@Setter
@NoArgsConstructor
@EqualsAndHashCode
public class Book {
   private Long id;
   private String name;
   private String author;
}
```

6. Spring Boot Starter Thymeleaf – зависимость для интеграции шаблонизатора, который упрощает создание динамических HTML-страниц.

В Thymeleaf используются естественные шаблоны – файлы, включающие элементы кода, которые тем не менее можно открыть и просмотреть в любом стандартном браузере. Возможность просматривать файлы шаблонов как HTML позволяет разработчикам и архитекторам создавать и развивать шаблоны Thymeleaf без запуска каких-либо серверных процессов. Все взаимодействия кода, для которых требуются соответствующие серверные элементы, помечаются как ориентированные на Thymeleaf, отсутствующие части просто не отображаются [3].

В примере показана форма для назначения книги конкретному пользователю. Thymeleaf позволяет обращаться непосредственно к полям и их значениям, а также динамически предоставлять списки объектов сущностей.

```
<form th:method="POST" th:action="@{/books/user/{id}(id=${book.getId()})}">
   <label for="user">Выберите читателя: </label>
      <select th:object="${user}" th:field="*{id}" id="user">
      <option th:each="user : ${users}" th:value="${user.getId()}"
      th:text="${user.getName()}"></option>
      </select>
   <input type="submit" class="button" value="Назначить книгу">
</form>
```

Существуют другие технологии представлений, такие как FreeMarker, Groovy Markup и Mustache. Однако при использовании шаблонов FreeMarker необходимо изменять html файлы на ftlh, что не позволяет шаблонизатору Thymeleaf корректно работать.

В заключение отметим, что Spring Framework благодаря своей модульной архитектуре и широкому набору инструментов значительно упрощает процесс разработки Java веб-приложений. Использование Spring Boot, внедрение зависимостей, легкая интеграция с базами данных и работа с шаблонизаторами дает возможность сосредоточиться на бизнес-задачах, минимизируя необходимость в рутинной настройке. Такой подход делает Spring Framework отличным выбором как для обучения, так и для профессиональной разработки, особенно в условиях современных стандартов качества, гибкости и скорости при создании веб-приложений.

Библиографический список

1. Почему Spring? [сайт]. – URL: https://spring.io/why-spring (дата обращения 02.12.2024)

2. Spring Framework и Spring Boot: понимание различий [Сайт]. – URL: https://topjava.ru/blog/spring-framework-vs-spring-boot-differences (дата обращения 02.12.2024)

3. *Хеклер, Марк.* Spring Boot по-быстрому / Марк Хеклер. – Санкт-Петербург: Питер, 2022. – 352 с.

4. Maven против Gradle: всестороннее сравнение для разработчиков Spring Boot [Сайт]. – URL: https://akalankaih19.medium.com/maven-vs-gradle-a-comprehensive-comparison-for-spring-boot-developers-a04136562e (дата обращения 02.12.2024)

5. *Лонг, Джош.* Java в облаке. Spring Boot, Spring Cloud, Cloud Foundry / Джош Лонг, Кеннет Бастани. – Санкт-Петербург: Питер, 2019. – 624 с.