AXIOM JDK

Фреймворк "Axiom Spring" Функциональные характеристики

Axiom JDK I Апрель 2025

Copyright © 2019-2025 Все права защищены АО "АКСИОМ" (АКСИОМ)

Программное обеспечение АКСИОМ содержит программное обеспечение с открытым исходным кодом. Дополнительная информация о коде сторонних разработчиков доступна на сайте https://axiomjdk.ru/third_party_licenses. Для дополнительной информации о том, как получить копию исходного кода, можно обратиться по адресу info@axiomjdk.ru.

ДАННАЯ ИНФОРМАЦИЯ МОЖЕТ ИЗМЕНЯТЬСЯ БЕЗ ПРЕДВАРИТЕЛЬНОГО УВЕДОМЛЕНИЯ. АКСИОМ ПРЕДОСТАВЛЯЕТ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ "КАК ЕСТЬ" БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, АКСИОМ ПРЯМО ОТКАЗЫВАЕТСЯ ОТ ВСЕХ ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ, ВКЛЮЧАЯ, НО НЕ ОГРАНИЧИВАЯСЬ ПОДРАЗУМЕВАЕМЫМИ ГАРАНТИЯМИ ТОВАРНОЙ ПРИГОДНОСТИ И ПРИГОДНОСТИ ДЛЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ.

АКСИОМ НИ ПРИ КАКИХ ОБСТОЯТЕЛЬСТВАХ НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА ЛЮБЫЕ КОСВЕННЫЕ, СЛУЧАЙНЫЕ, СПЕЦИАЛЬНЫЕ, ШТРАФНЫЕ ИЛИ КОСВЕННЫЕ УБЫТКИ, ИЛИ УБЫТКИ ОТ ПОТЕРИ ПРИБЫЛИ, ДОХОДА, ДАННЫХ ИЛИ ИСПОЛЬЗОВАНИЯ ДАННЫХ, ПОНЕСЕННЫЕ ВАМИ ИЛИ ЛЮБОЙ ТРЕТЬЕЙ СТОРОНОЙ, БУДЬ ТО В РЕЗУЛЬТАТЕ ДЕЙСТВИЯ ДОГОВОРА ИЛИ ДЕЛИКТА, ДАЖЕ ЕСЛИ АКСИОМ БЫЛО ПРЕДУПРЕЖДЕНО О ВОЗМОЖНОСТИ ТАКИХ УБЫТКОВ.

Использование любого программного продукта АКСИОМ регулируется соответствующим лицензионным соглашением, которое никоим образом не изменяется условиями данного уведомления. Программные продукты и фирменные наименования: Axiom JDK, Axiom JDK Pro, Axiom Runtime Container Pro, Axiom Linux, Libercat, Libercat Certified и АКСИОМ принадлежат АКСИОМ и их использование допускается только с разрешения правообладателя.

Товарный знак Linux® используется в соответствии с сублицензией от Linux Foundation, эксклюзивного лицензиата Линуса Торвальдса, владельца знака на всемирной основе. Java и OpenJDK являются товарными знаками или зарегистрированными товарными знаками компании Oracle и/или ее аффилированных лиц. Другие торговые марки являются собственностью их соответствующих владельцев и используются только в целях идентификации.

Содержание

1. Описание функциональных характеристик

Spring Framework — это мощная и гибкая платформа для создания корпоративных приложений на Java. Она предоставляет множество функциональных возможностей, которые делают разработку проще, надежнее и удобнее.

Инверсия управления (IoC) и внедрение зависимостей (DI)

Spring использует IoC (Inversion of Control) для управления зависимостями в приложении. При этом подходе контейнер Spring управляет жизненным циклом и связями объектов.

IoC реализуется через механизмы:

- Внедрение зависимостей (Dependency Injection).
- Использование аннотаций (`@Component`, `@Service`, `@Repository`).
- XML-конфигурации (устарело, но поддерживается).
- Конфигурации на основе Java с использованием классов (`@Configuration` и `@Bean`).

Внедрение зависимостей (Dependency Injection) упрощает тестирование и модульность, позволяя использовать различные реализации интерфейсов.

Пример DI через аннотации:

```
@Service
public class UserService {
    private final UserRepository userRepository;

    @Autowired
    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }
}
```

Модульность

Spring Framework разбит на модули, которые можно использовать по мере необходимости. Основные модули:

- **Spring Core**: Основной модуль, включающий контейнер IoC.
- **Spring AOP**: Поддержка аспектно-ориентированного программирования.
- **Spring MVC**: Фреймворк для создания веб-приложений.
- **Spring Data**: Упрощает взаимодействие с базами данных.
- **Spring Security**: Для обеспечения безопасности приложений.
- **Spring Boot**: Упрощенная разработка и конфигурация приложений.
- **Spring Batch**: Для обработки больших объемов данных.
- **Spring Integration**: Для реализации интеграционных решений.

Аспектно-ориентированное программирование (AOP)

Spring предоставляет механизм AOP (Aspect-Oriented Programming) для выполнения кросс-контекстных задач (например, логирование, обработка исключений, транзакции). Spring AOP позволяет внедрять дополнительную логику (аспекты) без изменения основного кода.

AOP позволяет отделить бизнес-логику от вспомогательных задач, повышая читаемость и модульность кода.

Основные компоненты AOP:

- **Pointcut**: Определение точек выполнения кода.
- **Advice**: Логика, которая выполняется до, после или вокруг основного метода.
- **Aspect**: Совокупность Pointcut и Advice.

Пример использования AOP:

```
@Aspect
@Component
public class LoggingAspect {
    @Before("execution(* com.example.service.*.*(..))")
    public void logBefore(JoinPoint joinPoint) {
```

```
        System.out.println("Вызов метода: " +
joinPoint.getSignature().getName());
    }
}
```

Работа с базами данных (Spring Data)

Spring Data предлагает унифицированный способ взаимодействия с различными базами данных, включая реляционные (JPA, JDBC) и NoSQL (MongoDB, Redis).

Spring поддерживает декларативные методы работы с репозиториями, например:

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    List<User> findByLastName(String lastName);
}
```

Spring MVC и REST

Spring MVC предоставляет инструменты для разработки веб-приложений, включая поддержку контроллеров, маршрутизации и шаблонов представлений.

Spring позволяет делать простую интеграцию с REST API:

- Аннотации для создания REST-контроллеров (`@RestController`, `@GetMapping`, `@PostMapping`).
- Поддержка сериализации данных в JSON или XML.

Пример REST-контроллера:

```
@RestController
@RequestMapping("/users")
public class UserController {
    @GetMapping("/{id}")
    public User getUser(@PathVariable Long id) {
        return userService.findById(id);
    }
}
```

Spring Boot

Spring Boot — это надстройка над Spring Framework, упрощающая конфигурацию и запуск приложений.

Spring Boot поддерживает следующие функции и компоненты:

- Встроенные веб-серверы (Axiom Libercat, Tomcat, Jetty, Undertow).
- Автоконфигурация (`@EnableAutoConfiguration` модуль Spring Boot Starter).
- Конфигурации через файл `application.properties` или `application.yml`.
- Быстрый запуск через `java -jar`.

Безопасность (Spring Security)

Spring Security — мощный инструмент для обеспечения безопасности приложений:

- Аутентификация и авторизация.
- Интеграция с OAuth2, JWT, SAML.
- Защита от CSRF, XSS и других атак.

Тестирование

Spring предоставляет инструменты для удобного тестирования приложений:

- Поддержка юнит-тестов с помощью JUnit и TestNG.
- Интеграционные тесты с загрузкой контекста Spring (`@SpringBootTest`).
- Mock-объекты и эмуляция слоев приложения с использованием `@MockBean`.

Поддержка транзакций

Spring поддерживает управление транзакциями в реляционных и NoSQL базах данных.

Декларативное управление транзакциями с использованием `@Transactional`, например:

```
@Service
@Transactional
```

```
public class UserService {
    public void createUser(User user) {
        userRepository.save(user);
    }
}
```

Интеграция и взаимодействие с другими технологиями

Spring поддерживает интеграцию со следующими технологиями:

- JMS (Java Message Service).
- Kafka, RabbitMQ.
- Spring WebFlux (реактивное программирование).
- Интеграция с облаками (AWS, GCP, Azure).

Управление конфигурацией

Spring поддерживает внешние конфигурации для гибкой настройки приложений:

- Файлы `application.properties` или `application.yml`.
- Хранилища конфигураций, такие как Spring Cloud Config Server.

Поддержка реактивного программирования

Spring WebFlux позволяет создавать реактивные, асинхронные приложения с поддержкой обработки больших объемов данных.

Пример WebFlux-контроллера:

```
@RestController
public class ReactiveController {
    @GetMapping("/flux")
    public Flux<String> getFlux() {
        return Flux.just("A", "B", "C");
    }
}
```

Мониторинг и управление

Spring предоставляет инструменты для мониторинга приложений:

- Spring Actuator:
  - Метрики (/actuator/metrics).
  - Управление состоянием приложения.
- Интеграция с системами мониторинга (Prometheus, Grafana).

AXIOM JDK

Фреймворк "Axiom Spring"
Функциональные характеристики