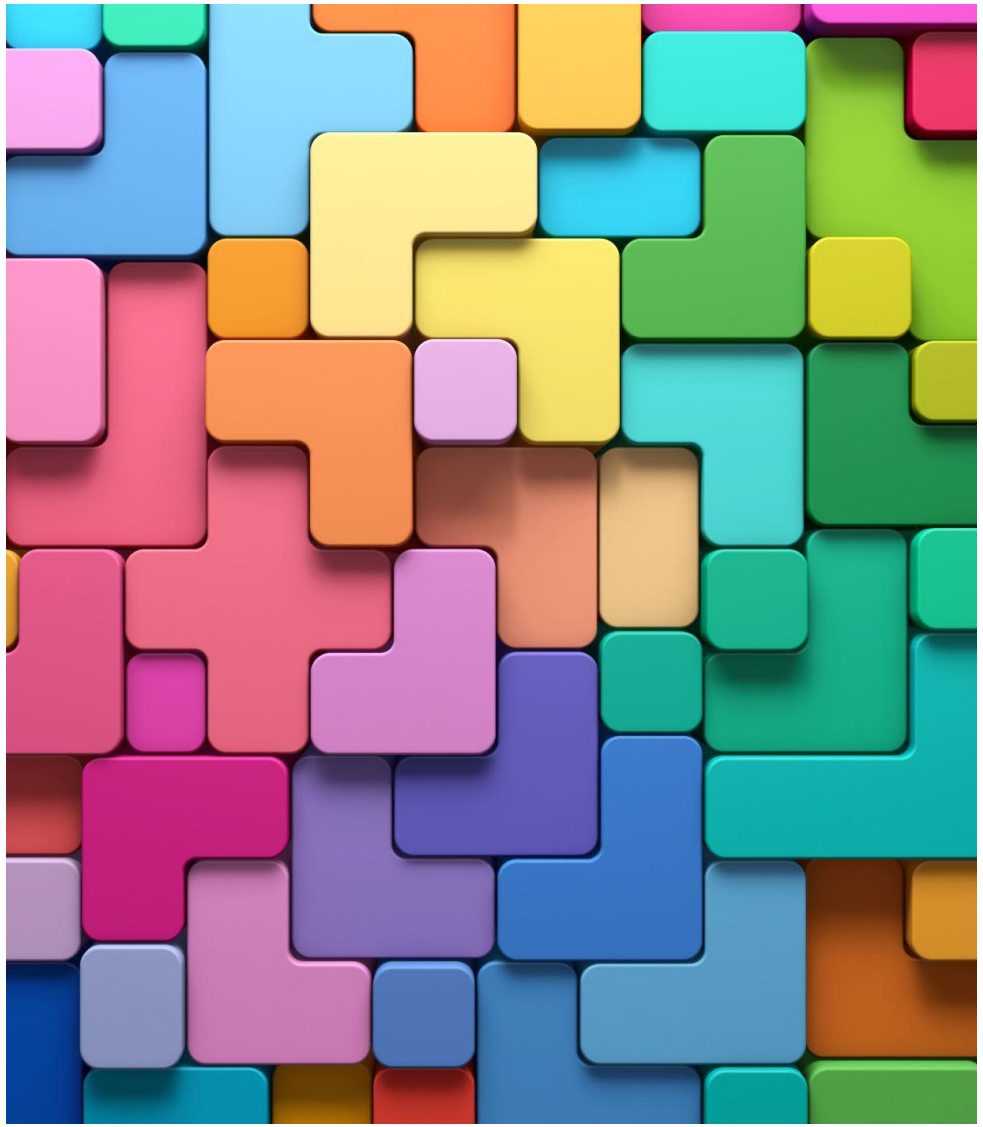
Java Pro

Spring Framework



Spring Framework introduction

- Spring Framework
- IoC (Inversion of Control)
- DI (Dependency Injection)
- Beans
- POJO
- ApplicationContext
- Аннотации @Configuration, @Bean

Spring Framework

Spring Framework — это популярный фреймворк на Java, который предоставляет инфраструктуру для создания приложений. Он помогает разработчикам создавать надежные, масштабируемые и поддерживаемые приложения, снимая с них многие задачи по управлению сложностью, такие как конфигурирование зависимостей, управление жизненным циклом компонентов, транзакциями, безопасностью и т.д.

Основная цель Spring — упрощение разработки сложных приложений, предоставляя **легковесный контейнер**, который можно настроить через простые POJOs

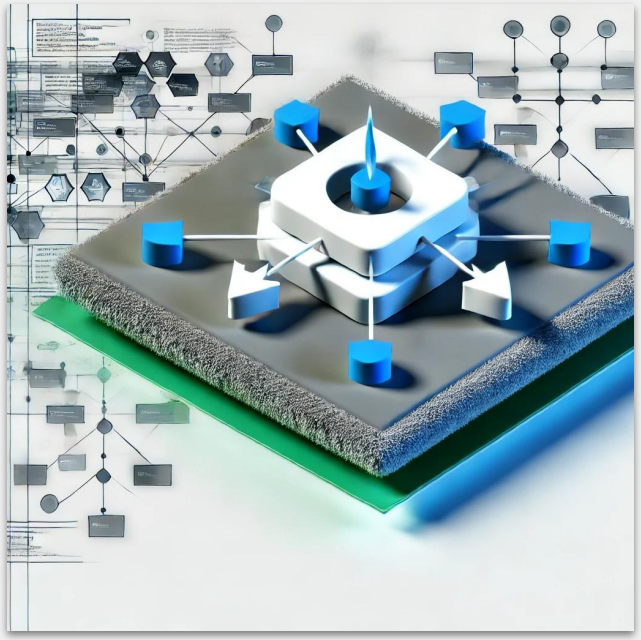
Преимущества Spring Framework

- **Мультиуровневая поддержка:** Spring предоставляет инструменты и поддержку для каждого уровня приложения, обеспечивая гибкость архитектуры и возможность адаптации под любые потребности.
- **Использование POJO:** Работа с простыми объектами Java (POJO) упрощает программирование, снижает связанность компонентов и делает их легче для тестирования.
- **Легкость интеграции и тестирования:** Spring облегчает интеграцию различных модулей и предоставляет мощные инструменты для тестирования, упрощая процесс разработки.
- **Декларативное программирование:** Поддержка декларативного программирования позволяет реализовывать функциональность через аннотации и конфигурации, минимизируя необходимость в шаблонном коде.
- **Управление ресурсами:** Spring автоматизирует управление ресурсами, такими как фабричные классы и синглтоны, делая код более чистым и устойчивым.
- **Гибкая конфигурация:** Поддержка различных способов конфигурации (XML, аннотации, Java-код) позволяет выбирать наиболее удобный подход для настройки приложения.
- **Поддержка middleware:** Spring упрощает взаимодействие с middleware, предоставляя инструменты для создания сложных, масштабируемых приложений.

Основная идея Spring: Инверсия управления (IoC)

Одна из ключевых идей, лежащих в основе Spring, — это **инверсия управления** (Inversion of Control, IoC).

В обычных приложениях разработчик сам создает и управляет объектами (например, инициализирует зависимости объектов). В Spring этот процесс передается фреймворку, который создает и управляет объектами за вас.



Инверсия управления (IoC)

Инверсия управления (IoC) и Внедрение зависимостей (DI)

- ➔ **Инверсия управления** позволяет передать управление созданием и конфигурацией объектов фреймворку. Это избавляет разработчиков от необходимости вручную управлять зависимостями объектов.
- ➔ **Внедрение зависимостей** позволяет легко управлять зависимостями между компонентами, что упрощает конфигурацию приложения и делает код более модульным и тестируемым.

Легковесный контейнер Spring

Когда говорят о "легковесном контейнере" Spring, имеют в виду специальный механизм Spring, который управляет созданием и конфигурированием объектов. Этот контейнер называется Spring IoC Container.

Как это работает?

1. **Определение объектов:** Вы создаете классы (объекты Java), которые описывают логику вашего приложения. Эти классы называются **POJOs (Plain Old Java Objects)** — это обычные объекты Java, не зависящие от каких-либо специфических API фреймворков.
2. **Конфигурация** через аннотации или XML: Вместо того, чтобы вручную создавать объекты этих классов и настраивать их, вы описываете зависимости между ними с помощью конфигурационных файлов или аннотаций. Например, с помощью аннотации @Autowired можно указать, что одна зависимость должна быть автоматически инъектирована в другой объект.
3. **Создание и управление объектами контейнером:** Spring Container (IoC контейнер) читает конфигурацию и создает объекты, управляет их жизненным циклом, связывает их зависимости. Например, если класс *ServiceA* зависит от *RepositoryB*, Spring Container автоматически создаст и передаст нужную зависимость в ServiceA.
4. **Использование объектов:** Разработчик может использовать созданные и настроенные объекты в своем приложении без необходимости вручную создавать их или управлять их зависимостями. Spring берет на себя эту задачу.

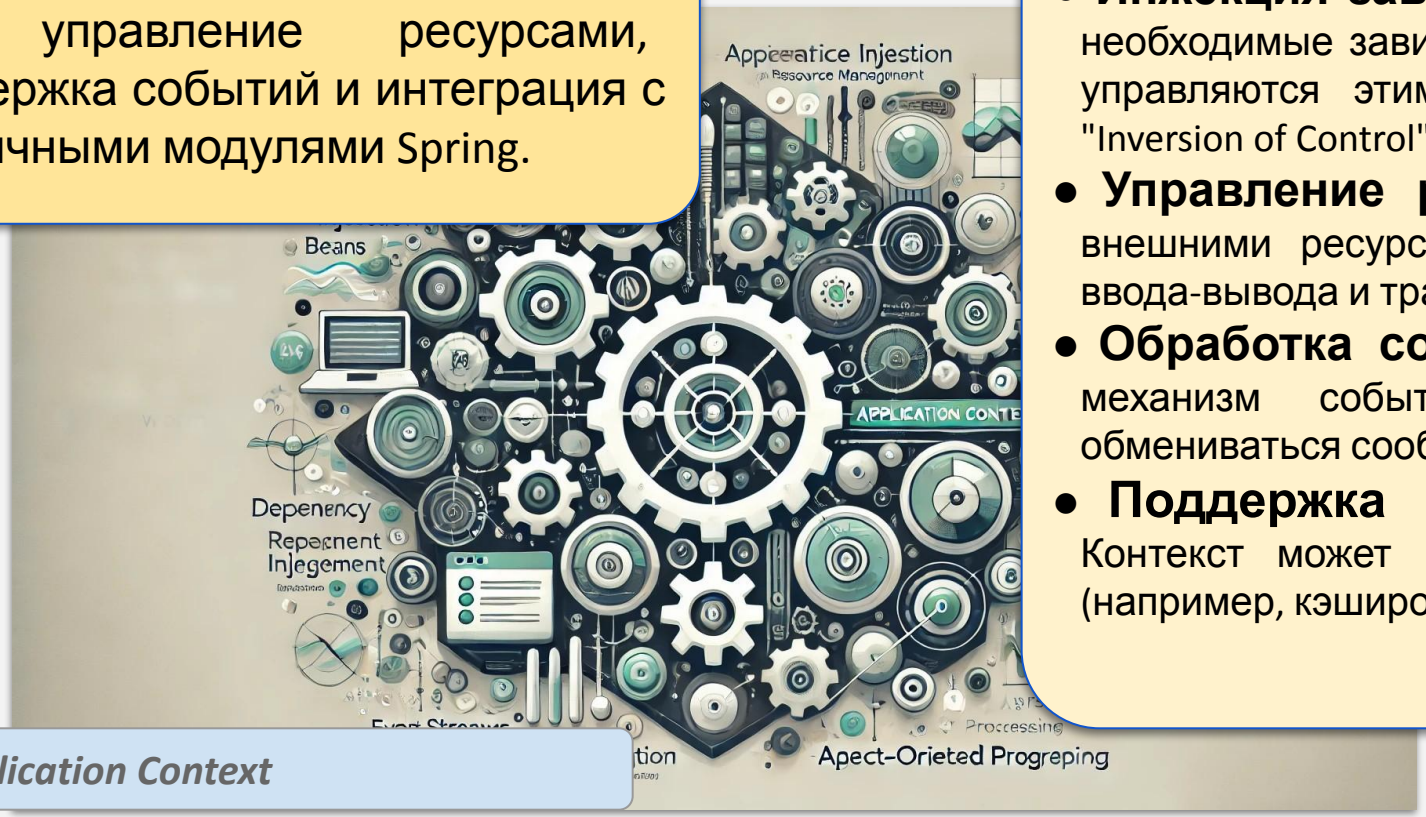
Бин (Bean) в Spring — это объект, который управляется IoC Container (контейнером инверсии управления).

Когда вы определяете компонент приложения как бин, Spring:

- **автоматически создаёт его экземпляр**,
- **управляет его жизненным циклом,**
- **внедряет зависимости,** а затем
- **предоставляет его другим частям вашего приложения**

Application Context

Application Context является одной из ключевых реализаций IoC Container в Spring. Он не только создаёт и управляет объектами, но также обеспечивает множество дополнительных функций, таких как управление ресурсами, поддержка событий и интеграция с различными модулями Spring.



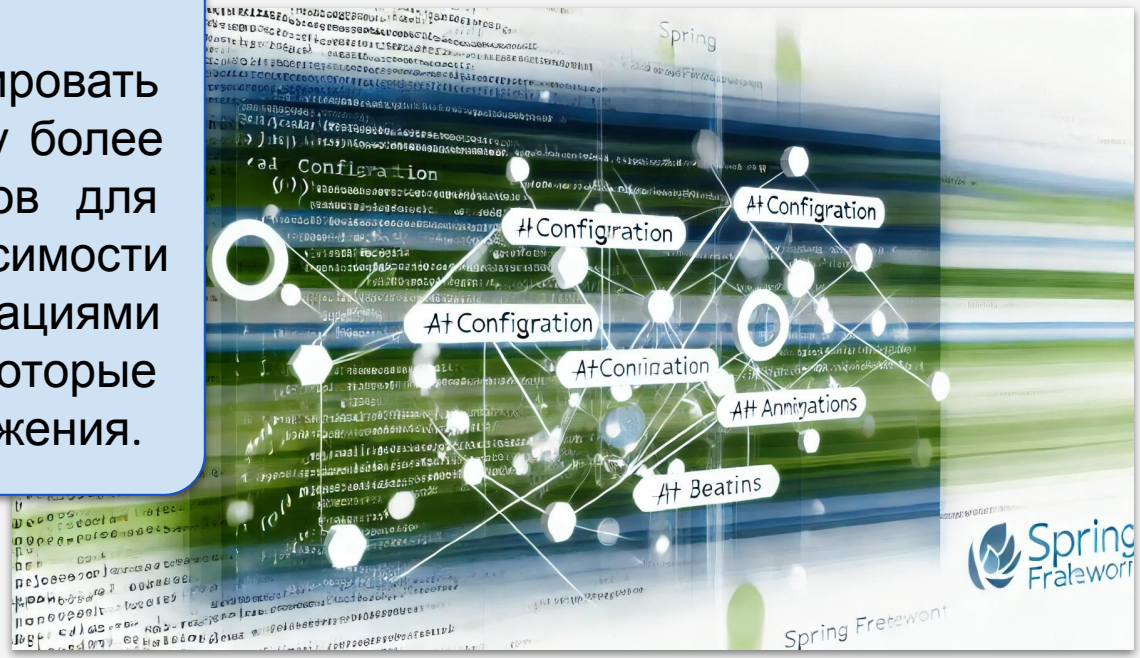
Application Context

Основные функции Application Context

- **Управление компонентами (бинами):** Application Context создаёт и управляет объектами (бинами) в приложении, обеспечивая их правильную инициализацию, зависимость и жизненный цикл.
- **Инжекция зависимостей:** Он автоматически внедряет необходимые зависимости (другие бины) в объекты, которые управляются этим контейнером. Это реализует принцип "Inversion of Control" (IoC).
- **Управление ресурсами:** Контекст может управлять внешними ресурсами, такими как файлы свойств, потоки ввода-вывода и транзакции.
- **Обработка событий:** Application Context поддерживает механизм событий, который позволяет компонентам обмениваться сообщениями.
- **Поддержка AOP** (Aspect-Oriented Programming): Контекст может применяться для управления аспектами (например, кэширование, транзакции) в приложении.

Конфигурация приложения

Spring Framework предоставляет возможность конфигурировать приложения с помощью Java-кода, что делает настройку более удобной и гибкой. Вместо использования XML-файлов для конфигурации, вы можете описывать компоненты и зависимости прямо в коде, используя аннотации. Ключевыми аннотациями для этого подхода являются **@Configuration** и **@Bean**, которые позволяют определять и управлять бинами вашего приложения.



@Configuration

Обозначает **класс** как источник конфигурации Spring, где определяются бины и зависимости.

@Bean

Аннотация, которая ставится **над методом** внутри класса, помеченного @Configuration. Она указывает, что возвращаемый этим методом объект будет создан и помещен в контекст Spring как бин.