Java Pro 7.3

TEL-RAN
by Starta Institute

Введение в Spring

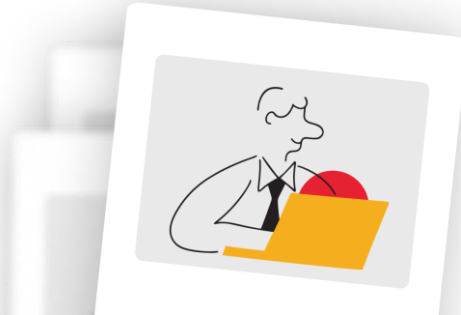
ПРЕПОДАВАТЕЛЬ

TEL-RAN
by Starta Institute

Юрий Костяной

Java/Kotlin backend-разработчик

- 3+ года опыта в коммерческой разработке
- 2+ года опыта в преподавании
- Проекты по интеграции сторонних платформ, CRM
- Проблемно-ориентированный подход в преподавании



ВАЖНО:

- Камера должна быть включена на протяжении всего занятия.
- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

TEL-RAN
by Starta Institute

ПЛАН ЗАНЯТИЯ

1. Повторение
2. Основной блок
3. Вопросы по основному блоку
4. Домашняя работа

TEL-RAN
by Starta Institute

1 ПОВТОРЕНИЕ ИЗУЧЕННОГО

Повторение

TEL-RAN
by Starta Institute

Какую аннотацию нужно добавить?

```java
public class ExampleClass {
    /**
     * Method is {@code Deprecated }.
     * Use {@link ExampleClass#newMethod()} instead.
     */
    public void oldMethod() {
        System.out.println("This is an old method.");
    }

    public void newMethod() {
        System.out.println("This is a  new method.");
    }

    public static void main(String[] args) {
        ExampleClass obj = new ExampleClass();
        obj.oldMethod();
    }
}
```

Повторение

TEL-RAN
by Starta Institute

Какую аннотацию нужно добавить?

@Deprecated показывает, что метод устарел. Параметр forRemoval поясняет, будет ли элемент удалён в следующих версиях Java.

```java
public class ExampleClass {
    /**
     * Method is {@code Deprecated }.
     * Use {@link ExampleClass#newMethod()} instead.
     */
    @Deprecated(forRemoval = true)
    public void oldMethod() {
        System.out.println("This is an old method.");
    }

    public void newMethod() {
        System.out.println("This is a  new method.");
    }

    public static void main(String[] args) {
        ExampleClass obj = new ExampleClass();
        obj.oldMethod();
    }
}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибки в коде

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.CLASS)
public @interface InvokeFirst {
}
```

```
public class SmartHouse {
    @InvokeFirst
    public void switchOnLight() {
        System.out.println("light is switched on");
    }
}
```

```
public static void main(String[] args) {
Class<SmartHouse> clazz = SmartHouse.class;
    Method[] methods = clazz.getMethods();
    System.out.println(Arrays.stream(methods)
            .filter(m -> m.isAnnotationPresent(InvokeFirst.class))
            .map(Method::getName).toList());
}
```

Повторение

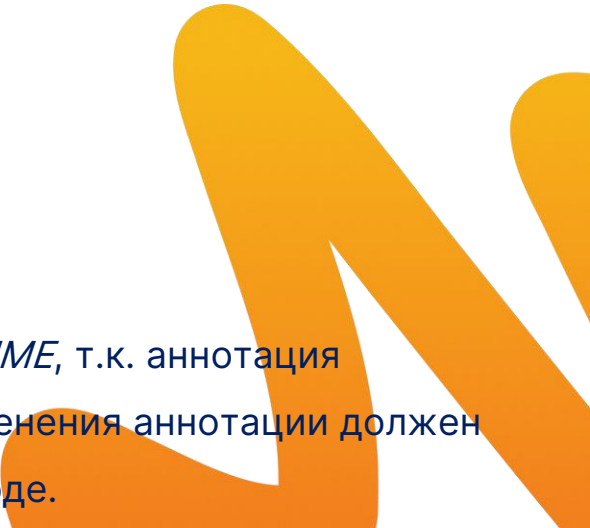
TEL-RAN
by Starta Institute

Исправьте ошибки в коде

```
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface InvokeFirst {
}
```

```
public class SmartHouse {
    @InvokeFirst
    public void switchOnLight() {
        System.out.println("light is switched on");
    }
}
```

```
public static void main(String[] args) {
Class<SmartHouse> clazz = SmartHouse.class;
    Method[] methods = clazz.getMethods();
    System.out.println(Arrays.stream(methods)
            .filter(m -> m.isAnnotationPresent(InvokeFirst.class))
            .map(Method::getName).toList());
}
```



1 Жизненный цикл аннотации должен быть *RetentionPolicy.RUNTIME*, т.к. аннотация использована во время выполнения программы. 2 Область применения аннотации должен быть *ElementType.METHOD*, т.к. аннотация использована на методе.

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде. Что будет выведено в консоль?

```
public class SingletonExample {
    private static SingletonExample instance;
    public SingletonExample() {}

    public static SingletonExample getInstance() {
        return instance = instance != null ? instance : new SingletonExample();
    }

    public static void main(String[] args) {
        SingletonExample firstInstance = SingletonExample.getInstance();
        SingletonExample secondInstance = SingletonExample.getInstance();
        System.out.println(firstInstance == secondInstance);
    }
}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде. Что будет выведено в консоль?

```
public class SingletonExample {
    private static SingletonExample instance;
    private SingletonExample() {}

    public static SingletonExample getInstance() {
        return instance = instance != null ? instance : new SingletonExample();
    }

    public static void main(String[] args) {
        SingletonExample firstInstance = SingletonExample.getInstance();
        SingletonExample secondInstance = SingletonExample.getInstance();
        System.out.println(firstInstance == secondInstance);
    }
}
```

Конструктор singleton-класса должен быть приватным. В консоль будет выведено *true*.

Повторение

Исправьте ошибку в коде

```
@AllArgConstructor
class Shape {
    private String type;

    @Override
    protected Object clone()
                throws CloneNotSupportedException {
        return super.clone();
    }
}

public class PrototypeExample {
    public static void main(String[] args) {
        Shape circle = new Shape("Circle");

        try {
            Shape clonedCircle = (Shape) circle.clone();
            System.out.println(circle == clonedCircle);
        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
        }
    }
}
```

Повторение

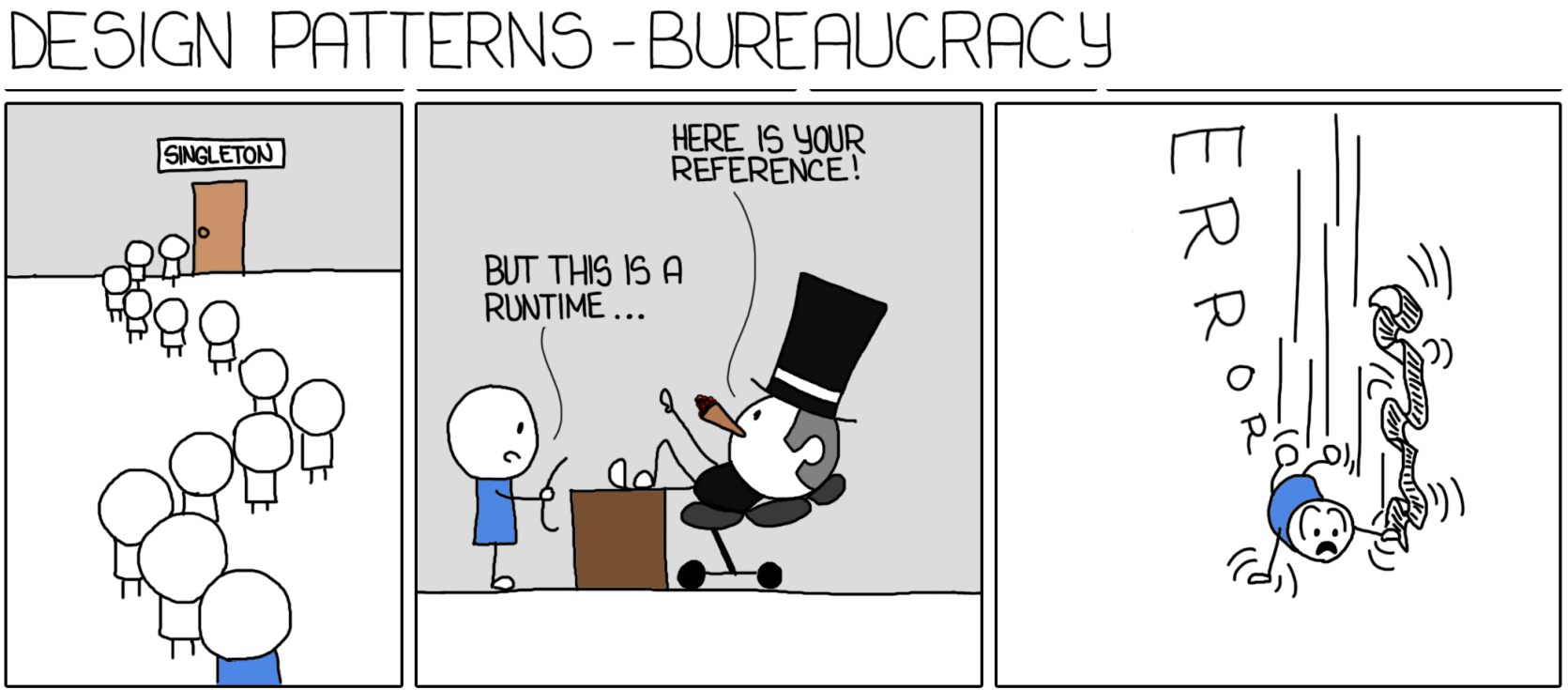
Исправьте ошибку в коде

```
@AllArgConstructor
class Shape implements Cloneable {
    private String type;

    @Override
    protected Object clone()
                throws CloneNotSupportedException {
        return super.clone();
    }
}

public class PrototypeExample {
    public static void main(String[] args) {
        Shape circle = new Shape("Circle");

        try {
            Shape clonedCircle = (Shape) circle.clone();
            System.out.println(circle == clonedCircle);
        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
        }
    }
}
```

Класс должен наследовать интерфейс *Cloneable*, чтобы вызов метода *clone()* не выбрасывал *CloneNotSupportedException*.

Повторение

В чём прикол мема?



TEL-RAN
by Starta Institute

2 ОСНОВНОЙ БЛОК

Введение

TEL-RAN
by Starta Institute

- Корпоративная Java
- Рамочная работа
- Весна идёт – весне дорогу!
- Зависимость и уколы
- На бобах

Проблема

TEL-RAN
by Starta Institute

Первые приложения на Java появились в далёком 1995 году. Со временем Java стала популярна как язык создания веб-приложений и веб-сервисов. Но всё, что мы изучали до этого, относится к обычному программированию и имеет мало отношения к веб-разработке.

Какие практики Java были созданы для разработки веб-приложений?

Корпоративная Java

Java Enterprise Edition

TEL-RAN
by Starta Institute

То, что мы изучали ранее, было **Java Standard Edition (Java SE)**, т.е. базовые спецификации языка и стандартной библиотеки Java.

Java Enterprise Edition (ранее **Java EE**, **JEE**, с 2018 года **Jakarta EE)** представляет платформу (набор взаимосвязанных спецификаций) для создания крупных и средних корпоративных приложений на языке Java. Прежде всего это сфера веб-приложений и веб-сервисов.

Корпоративная Java

Java Enterprise Edition

TEL-RAN
by Starta Institute

Java EE состоит из набора *API* и среды выполнения. Некоторые из *API*:

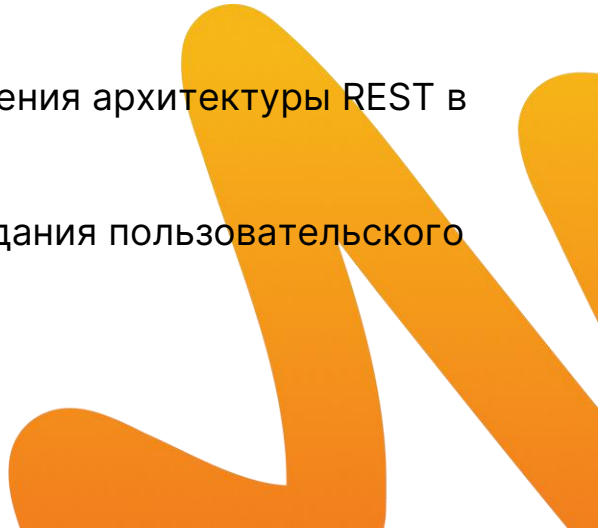
- **Java Servlets**. Сервлеты представляют специальные модули, которые обрабатывают запросы от пользователей и отправляют результат обработки.
- **JavaServer Pages (JSP)**. Также модули на стороне сервера, которые обрабатывают запросы. Удобны для генерации большого контента HTML. По сути представляют собой страницы с кодом HTML/JavaScript/CSS с вкраплениями кода на Java.
- **Enterprise JavaBeans (EJB)** представляют классы, которые хранят бизнес-логику.
- **Contexts and Dependency Injection (CDI)** предоставляет механизм для внедрения и управления зависимостями в другие объекты.
- **JSON Processing (JSON-P)** позволяет работать со строками *JSON* в Java
- **JSON Binding (JSON-B)** предоставляет функционал для сериализации и десериализации JSON в объекты Java.

Корпоративная Java

Java Enterprise Edition

TEL-RAN
by Starta Institute

- **WebSocket** позволяет интегрировать технологию *WebSocket* в приложения на Java для облегчённого обмена данными между серверным и клиентским приложением.
- **Java Message Service (JMS)** - *API* для пересылки сообщений между двумя и более клиентами.
- **Security API** - *API* для стандартизации и упрощения задач обеспечения безопасности в приложениях на Java.
- **Java API for RESTful Web Services (JAX-RS)** - *API* для применения архитектуры REST в приложениях.
- **JavaServer Faces (JSF)** предоставляет возможности для создания пользовательского интерфейса из готовых модулей на стороне сервера.



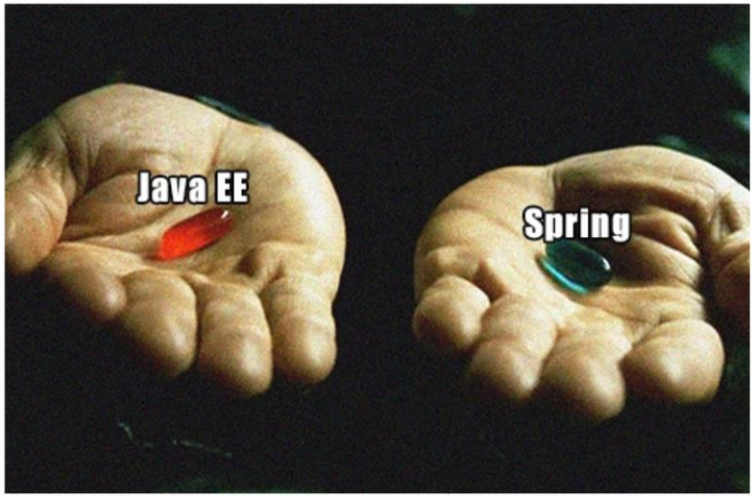
Корпоративная Java

JEE VS Spring

TEL-RAN
by Starta Institute

JEE – это официально принятый набор спецификаций, т.е. де-юре надо использовать все эти API и стандарты для разработки enterprise-приложений. Проектов, с JEE, на самом деле, много. Такие проекты ориентированы на многолетнее существование и развитие. Использование JEE даёт некоторую гарантию только необходимых изменений технологий, при этом JEE обеспечивает обратную совместимость, что даёт возможность развивать проект поэтапно.

Тем не менее, большинство проектов сейчас используют не чистый *JEE*, а *Spring Framework*, который следует спецификациям JEE, но не на 100%, и считается стандартом де-факто.



Корпоративная Java

JEE VS Spring

TEL-RAN
by Starta Institute

JEE как среда разработки сложнее, чем *Spring,* что в конечном итоге увеличивает окончательную стоимость проекта (сложно и дорого, но гибко).

Spring обеспечивает слабую связанность приложений, быстрее впитывает новые фишки и технологии. Например, конфигурации можно было выполнять на основе *XML*, затем появилась возможность использовать *Groovy* и аннотаций. Позволяет использовать простые старые объекты Java (*POJO* – plain old Java object), т.е. разработчикам не нужен корпоративный контейнер, такой как сервер приложений.

Обеспечивает разработчикам Java высокий уровень модульности. Предоставляет реализацию библиотек Java Eеи упрощает их использование. Лицензия с открытым исходным кодом. Spring Boot сильно упрощает первичную настройку приложения. Упрощает тестирование.

Задание

TEL-RAN
by Starta Institute

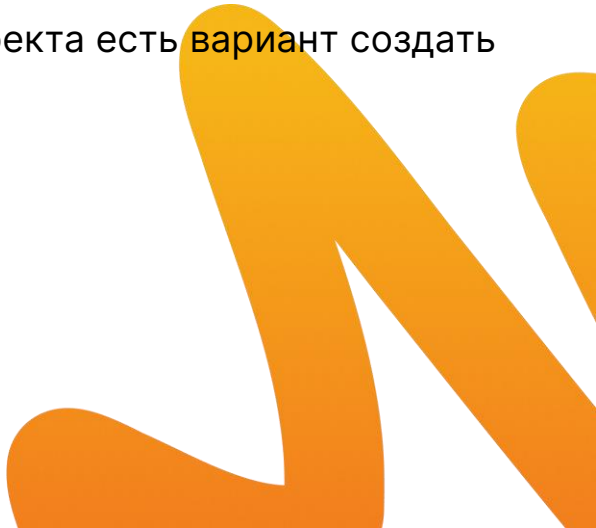
1 Скачать и усановить IntelliJ IDEA **Ultimate**

https://www.jetbrains.com/ru-ru/idea/download/?section=windows

Если уже установлена, то новую качать не нужно.

2 Создать JBAccount (зарегистрироваться при активации trial-версии). При создании аккаунта укажите, что Вы – студент.

3 После активации запустите и убедитесь, что при создании проекта есть вариант создать проект Java Enterprise и/или Spring.



Проблема

Представьте, что Вы – не разработчик ПО, а строитель и Вам нужно построить дом.

Первый вариант – создать проект дома с нуля. Без опыта проектирования, скорее всего, Вы построите дом, который не подойдет для проживания.

Второй вариант – использовать готовый типовой проект. В нем будут продуманы способы заливки фундамента, расположение коммуникаций, теплоизоляция стен. Но при этом нельзя изменить планировку, затрагивать несущие конструкции.

Второй способ ускоряет строительство дома и даёт гарантии, что дом будет надёжным.

Дом на колесах

Рамочная работа

Фреймворк

TEL-RAN
by Starta Institute

Фреймворк (framework) (англ. «каркас», «структура», «заготовка») – программная платформа, определяющая структуру приложения, облегчающее разработку и объединение разных компонентов большого программного проекта.

Фреймворки служат той же цели, что типовой проект в строительстве: разработчик использует готовый шаблон и наполняет его своим кодом. Фреймворки отвечают за стабильность программы: работу с базами данных и файловой системой, обработку ошибок, защиту паролем и т.д. Все низкоуровневые задачи решает фреймворк, а разработчик сосредотачивается в основном на написании бизнес-логики. При этом разработчик обязан следовать набору правил, которые предъявляет фреймворк.

Рамочная работа

Фреймворк

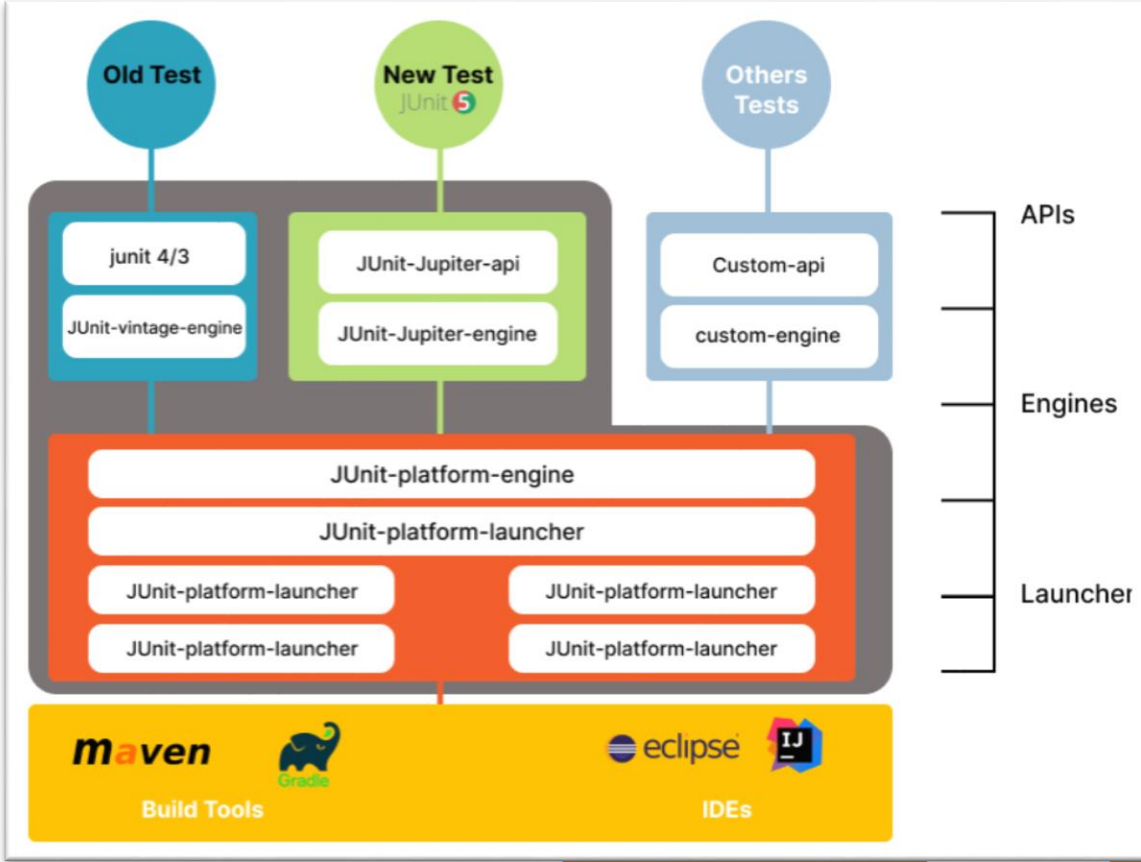
TEL-RAN
by Starta Institute

По сути, фремворк – это чужое приложение (набор взаимосвязанных классов), которое принимает Ваши классы на вход, подставляет их в точки расширения и затем начинает работу. Ваши классы следуют требованиям фреймворка (наследуют интерфейсы или помечаются аннотациями, содержат необходимы члены и т.д.).

Фремворк – это средство автоматизации и только от разработчика зависит, как это средство будет применяться в проекте.

Рамочная работа
Фреймворк

Мы уже сталкивались с фреймворком *JUnit 5*. Когда мы писали тесты, то это был набор классов с методами, но ни один класс не содержал метод *main()*. Запуск тестов представляет собой запуск *main()* в фреймворке *JUnit* и передачу фреймворку написанных классов тестов в качестве точек расширения.



Рамочная работа
Фреймворк

TEL-RAN
by Starta Institute

За запуск тестов могут отвечать разные классы фреймворка, поэтому в фрейворке предусмотрены специальные аннотации *@RunWith* и *@ExtendWith*, в настройках которых можно указать, какой класс должен запускать тесты.

```
@RunWith(JUnitPlatform.class)
public class GreetingsUnitTest {
    // ...
}
```

Запуск тестов в обычном проекте

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = { SpringTestConfiguration.class })
public class GreetingsSpringUnitTest {
    // ...
}
```

Запуск тестов в проекте со *Spring*

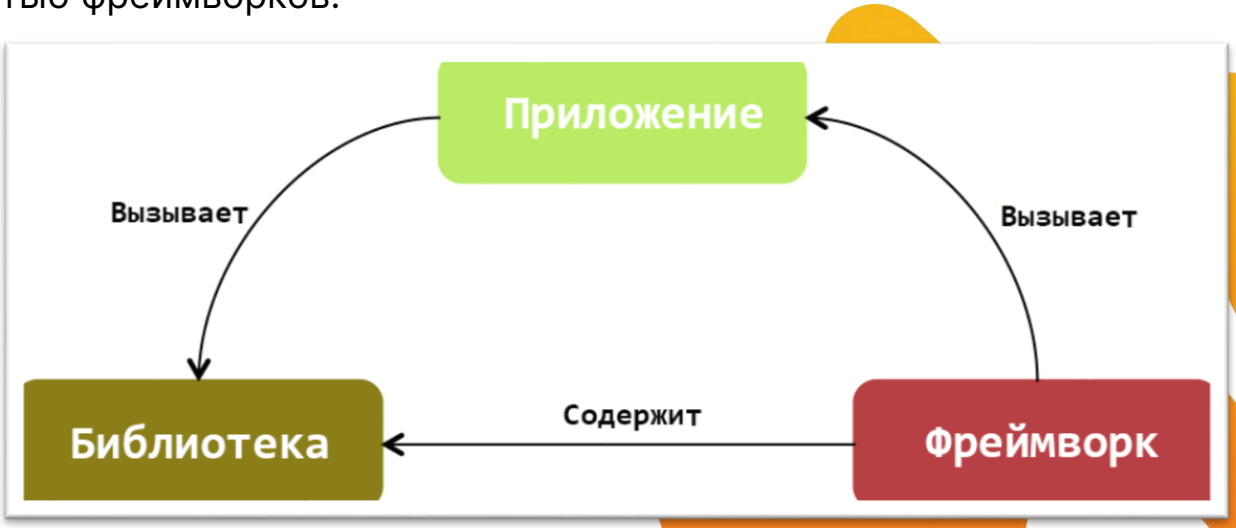
Рамочная работа

Фреймворк и библиотека

TEL-RAN
by Starta Institute

Библиотека – набор готовых (скомпилированных) классов, которые помогают решить определённую задачу или набор похожих задач. У библиотек и фреймворков одна цель – освободить программиста от постоянного решения однотипных задач. Но библиотека не влияет на структуру Вашего приложения. Библиотеки могут использоваться параллельно с фреймворками или быть частью фреймворков.

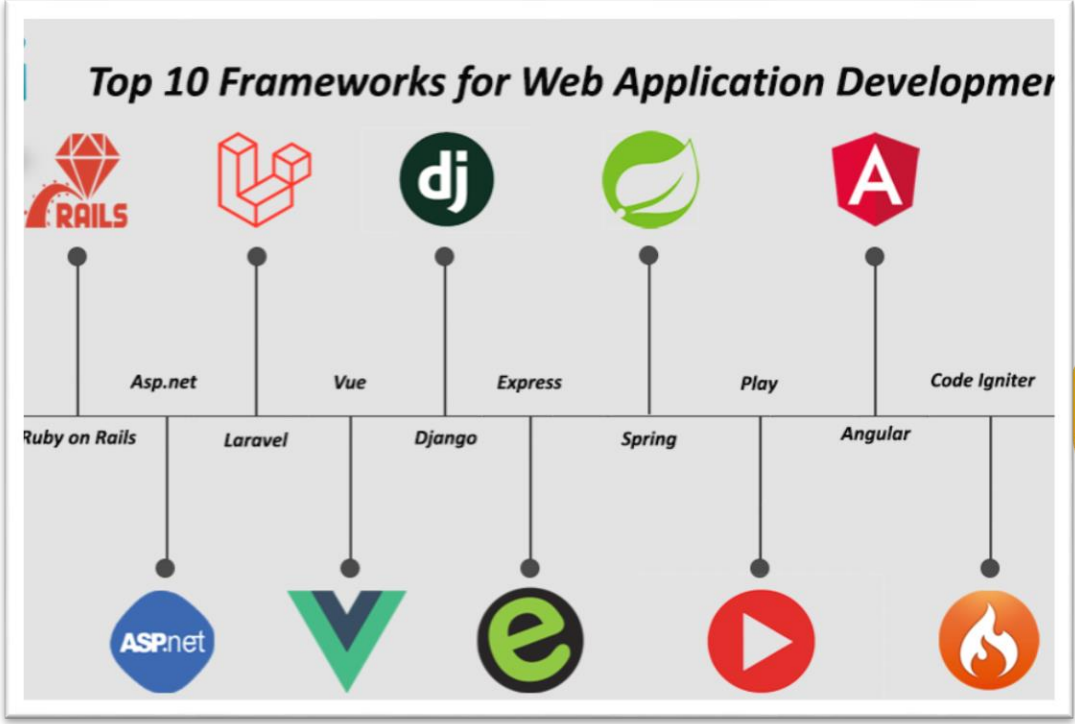
Например, библиотека *Jackson* используется в составе *Spring* «из коробки», хотя и может быть заменена на другую.



Рамочная работа

Фреймворки для web-разрвботки

TEL-RAN
by Starta Institute



Указанные фреймворки относятся к разным языкам программирования и включают как frontend, так и backend-фреймворки.

Рамочная работа

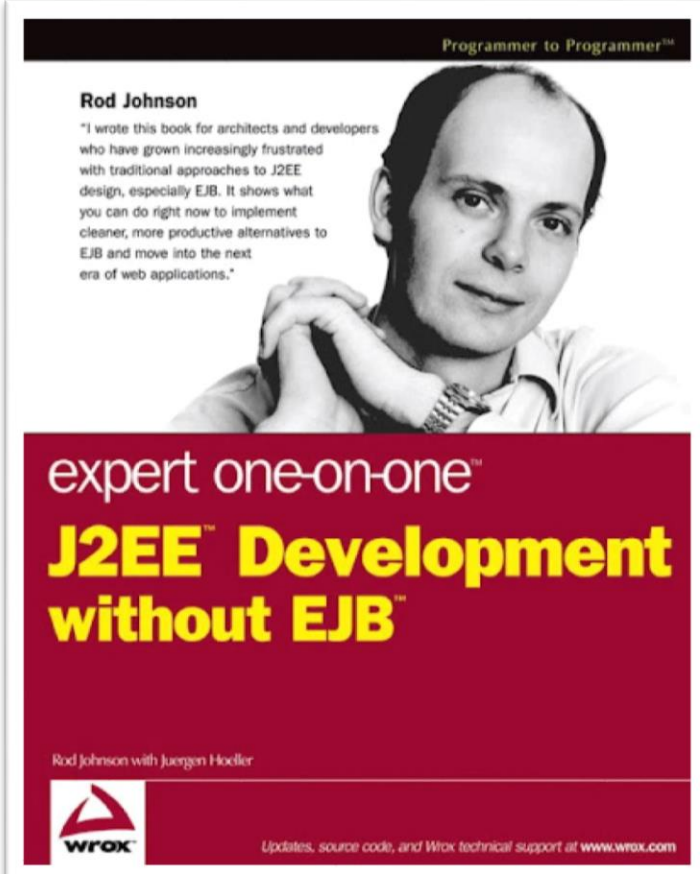
Фреймворки для Junior Java Developer

Проблема

TEL-RAN
by Starta Institute

В октябре 2002 года Род Джонсон написал книгу «*Expert One-onOne J2EE Design and Development*».

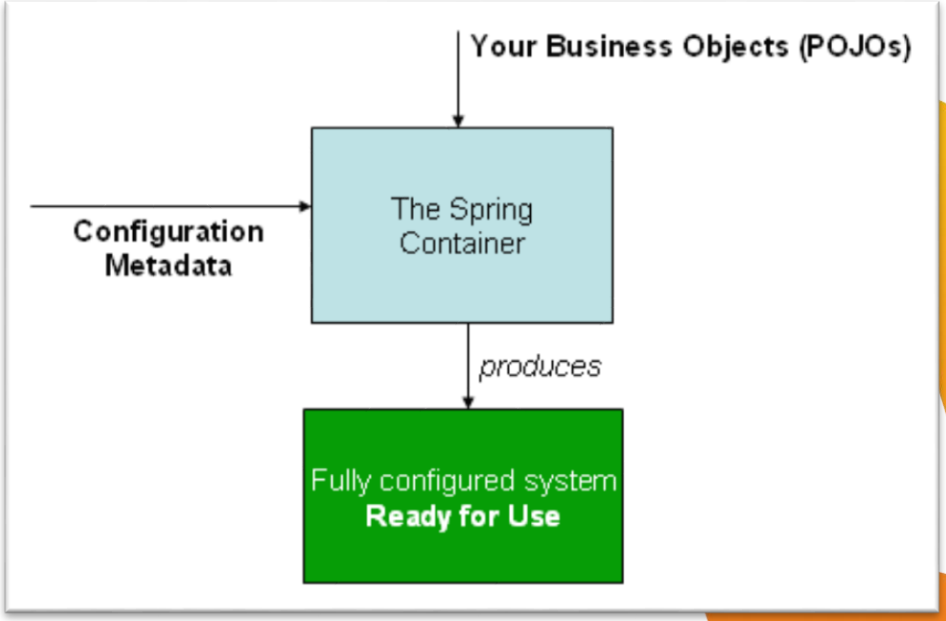
В этой книге автор указал на ряд недостатков *Java EE* и её компонентной среды *EJB (Enterprise Java Bean*). Вместо этого он предложил более простое решение, основанное на обычных Java-классах (*POJO*) и внедрении зависимостей (*DI - Dependency Injection*).



Проблема

TEL-RAN
by Starta Institute

Приведенный им дизайн разработки *J2EE* мгновенно стал хитом. Издатель книги предоставил веб-страницу для исходного кода, а также онлайн-форум для обсуждения. В феврале 2003 года разработчики Юрген Хеллер и Янн Карофф убедили Рода Джонсона создать проект *«Spring»*.

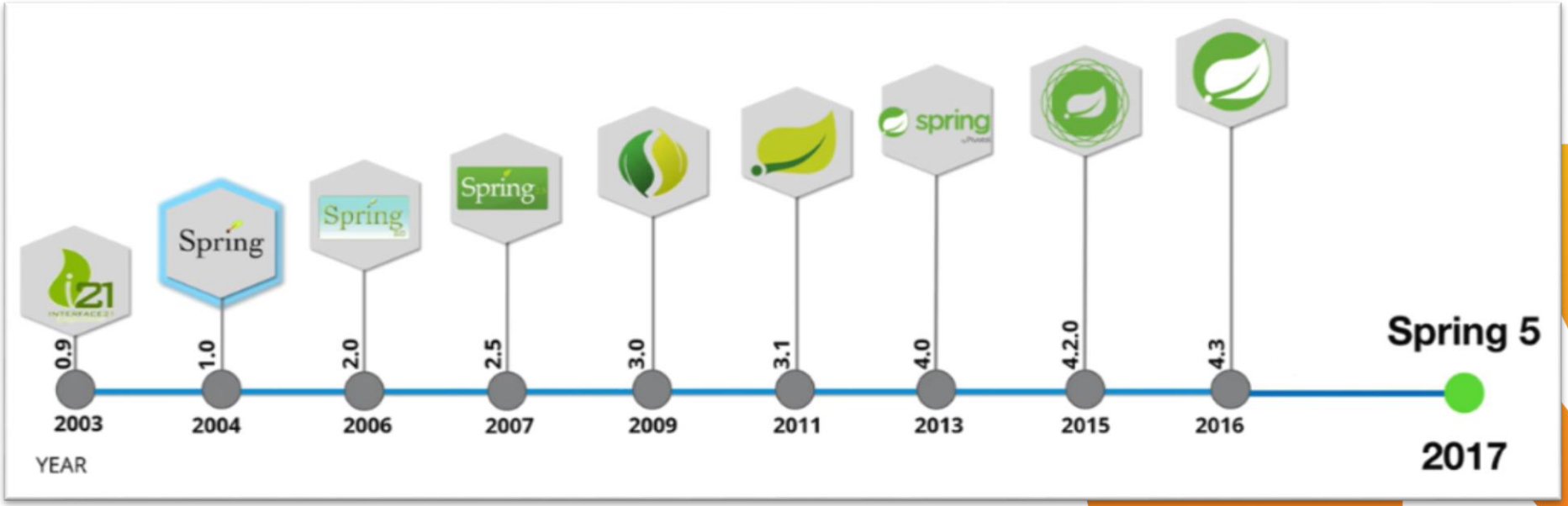


Весна идёт – весне дорогу!

Spring Framework

TEL-RAN
by Starta Institute

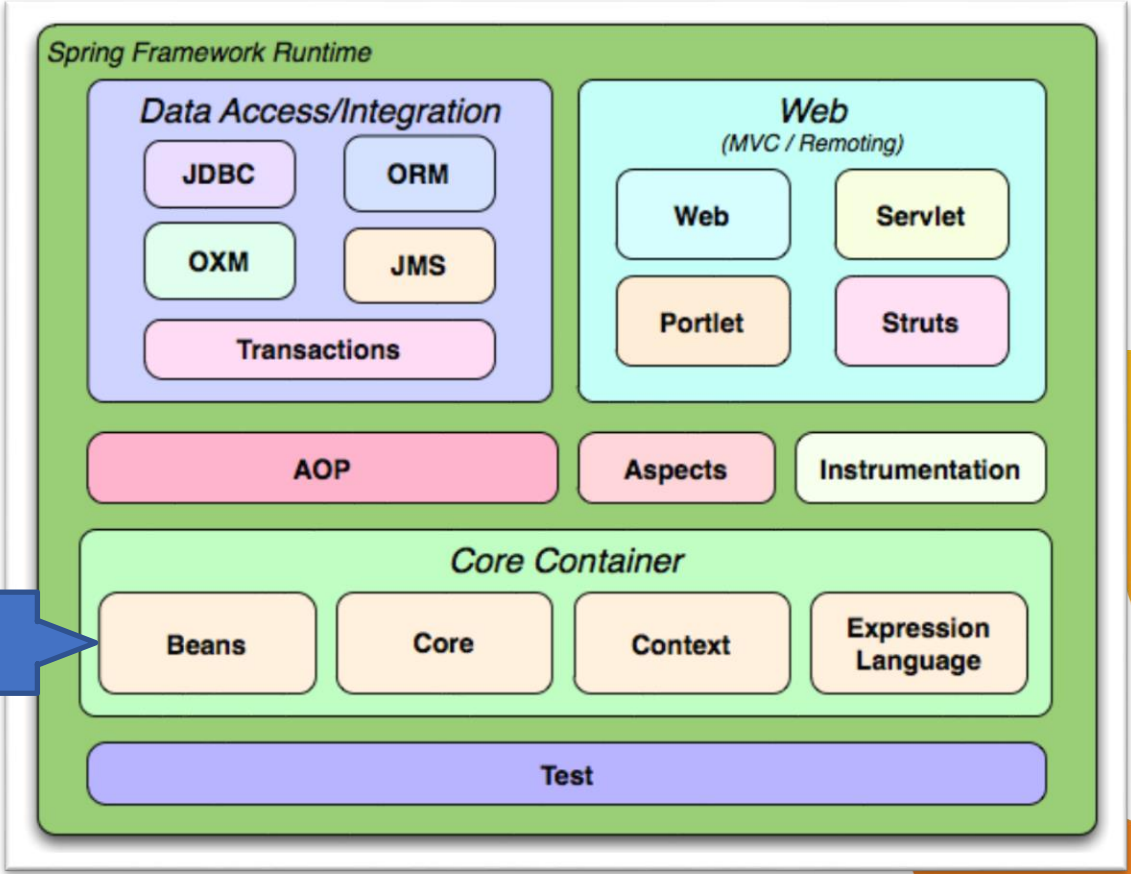
С 2003 года Spring Framework претерпел множество изменений. Мы будем рассматривать актуальную версию Spring 6.x.



Весна идёт – весне дорогу!

Структура Spring Framework

TEL-RAN
by Starta Institute

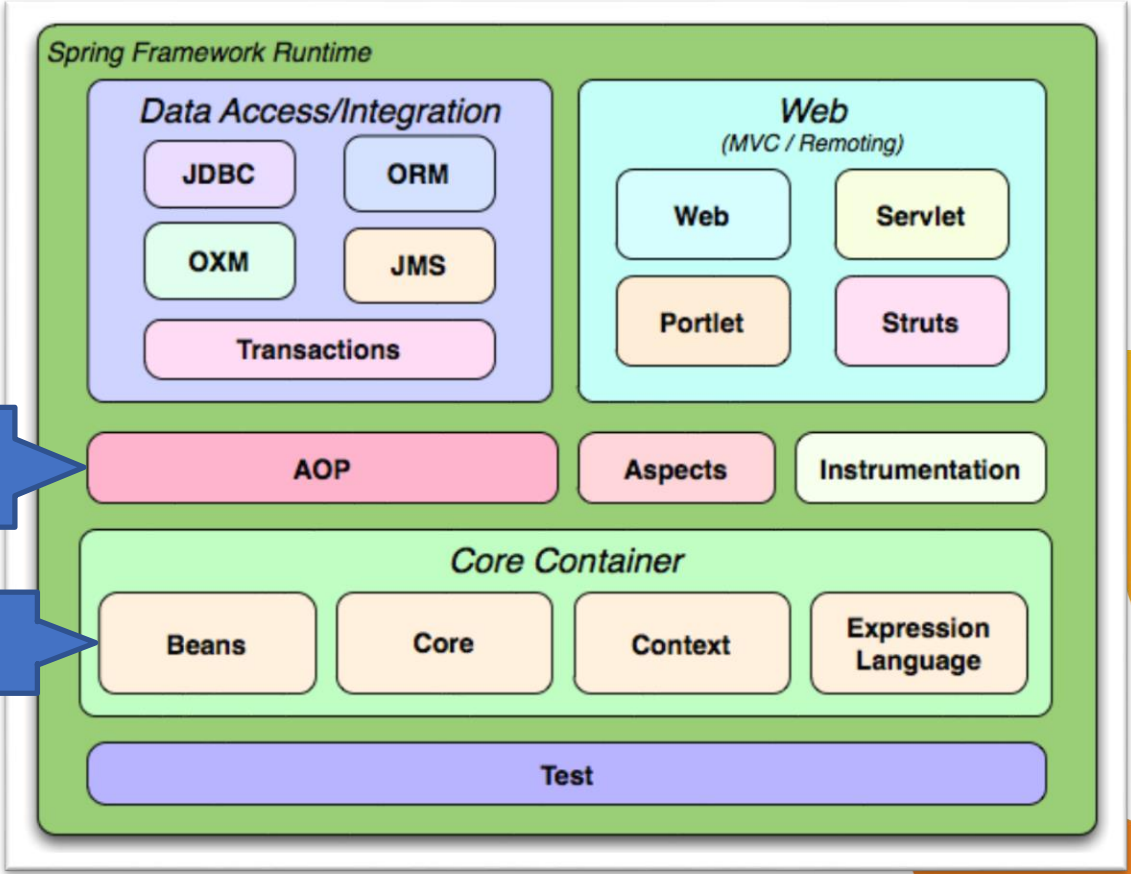


Основной модуль фреймворка

Весна идёт – весне дорогу!

Структура Spring Framework

TEL-RAN
by Starta Institute



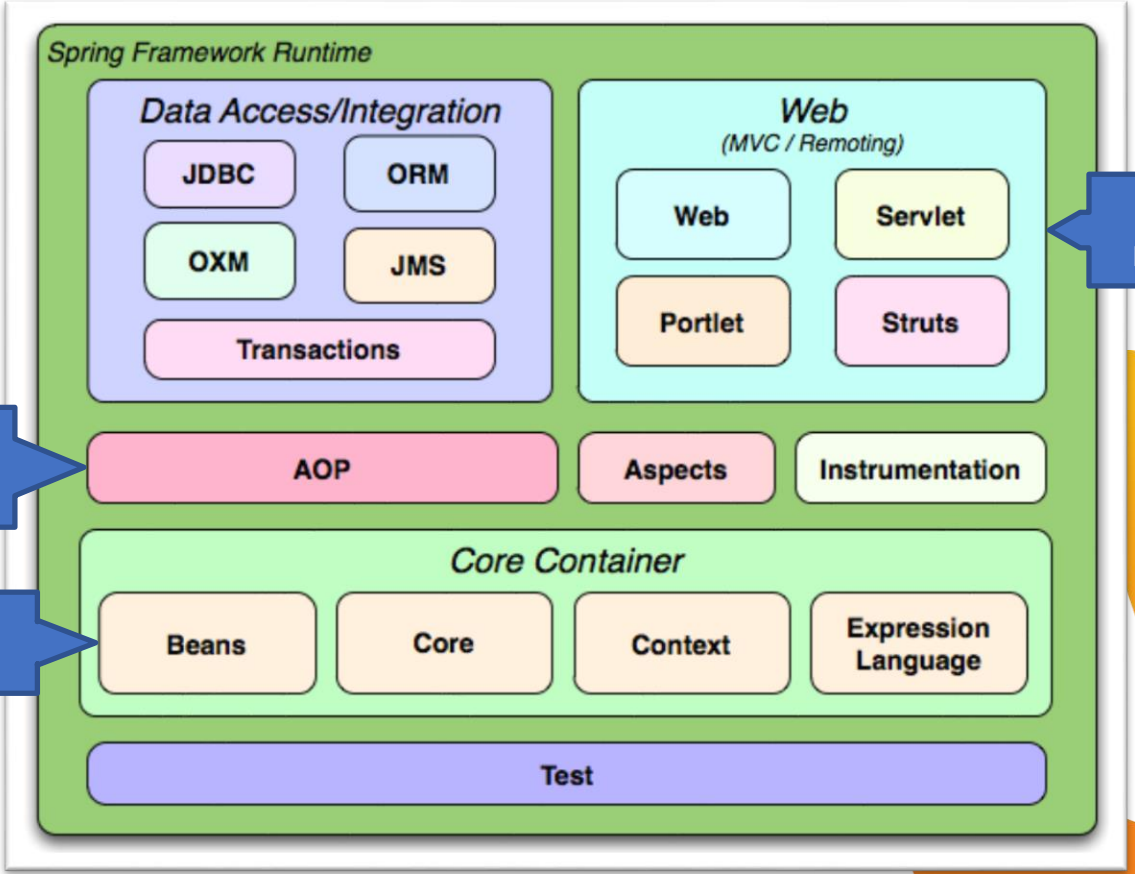
Модуль аспектно-ориентированного программирования

Основной модуль фреймворка

Весна идёт – весне дорогу!

Структура Spring Framework

TEL-RAN
by Starta Institute



Модуль работы веб-приложений

Модуль аспектно-ориентированного программирования

Основной модуль фреймворка

Весна идёт – весне дорогу!

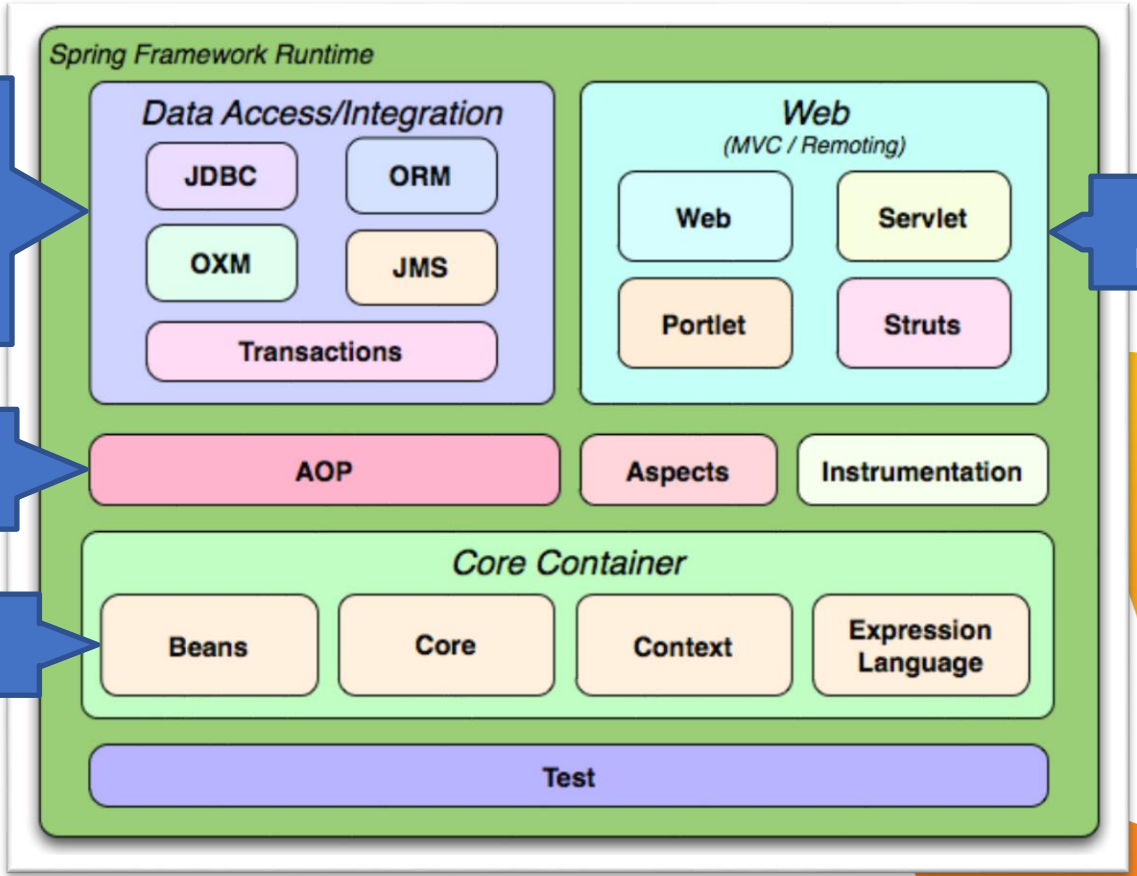
Структура Spring Framework

TEL-RAN
by Starta Institute

Модуль работы с базами данных и сервисами сообщений (транзакционные действия)

Модуль аспектно-ориентированного программирования

Основной модуль фреймворка



Модуль работы веб-приложений

Весна идёт – весне дорогу!

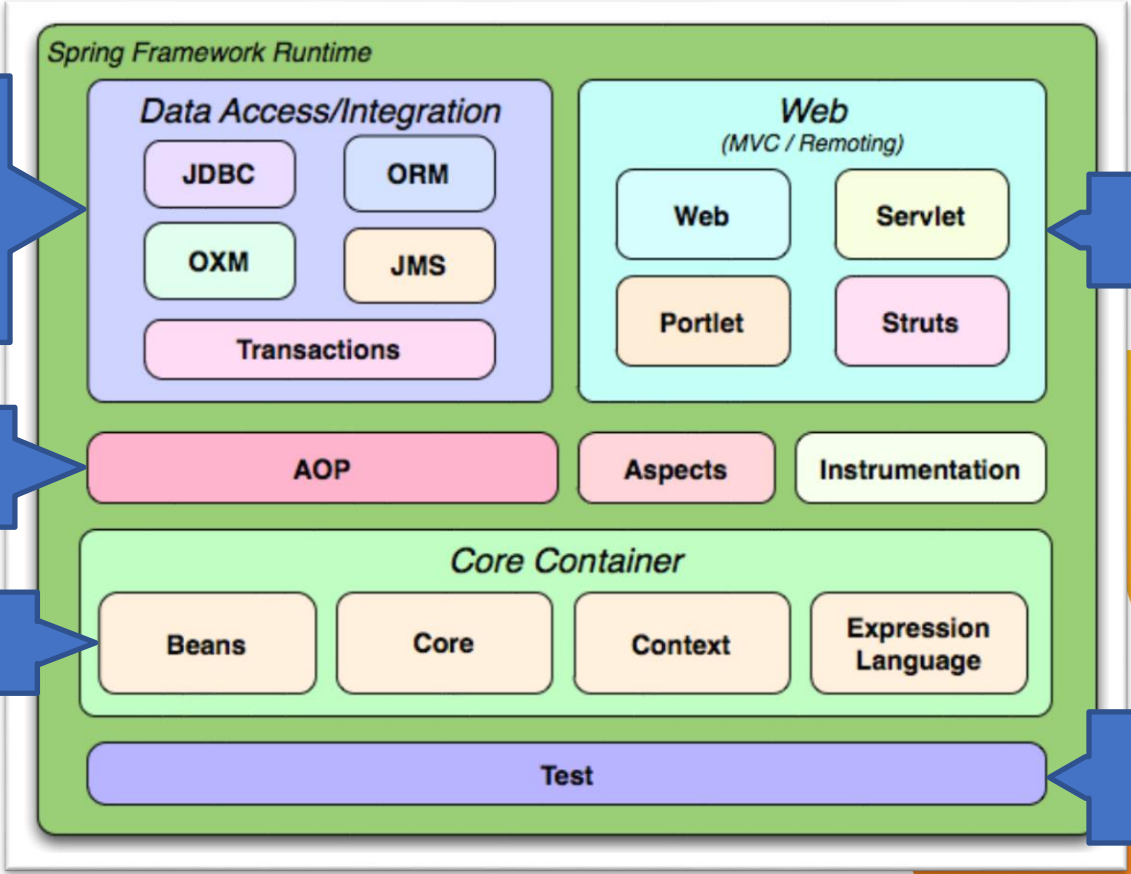
Структура Spring Framework

TEL-RAN
by Starta Institute

Модуль работы с базами данных и сервисами сообщений (транзакционные действия)

Модуль аспектно-ориентированного программирования

Основной модуль фреймворка



Модуль работы веб-приложений

Модуль тестирования приложений

Весна идёт – весне дорогу!

Другие компоненты Spring

TEL-RAN
by Starta Institute

Spring Boot – упрощает создание приложений на основе Spring, сокращая до минимума первичную настройку приложения и автоконфигурирует элементы приложения на *Spring*.
Пример запуска Spring boot за 5 минут

Spring Data - значительно упрощает использование технологий доступа к данным, реляционных и нереляционных баз данных (убирает повторяющийся код и упрощает взаимодействие с данными).

Spring Cloud - используется в микросервисной архитектуре, упрощая взаимодействие микросервисов между собой и автоматизируя развертывание приложений на облачных платформах типа *AWS*, *Azure* и т.д.

Spring Security - предоставляет мощный и настраиваемый инструмент проверки подлинности (аутентификации) и контроля доступа (авторизации) в приложение.

Весна идёт – весне дорогу!

Другие компоненты Spring

TEL-RAN
by Starta Institute

Spring GraphQL - данный модуль обеспечивает поддержку приложений Spring, построенных на *GraphQL Java*. **GraphQL** – это язык запросов для API, позволяющий клиентам запрашивать ограниченное множество данных, в которых они нуждаются, что в свою очередь позволяет собирать данные в ограниченном количестве запросов.

Spring Session – предоставляет API и реализации для управления информацией о сеансе пользователя (данные сеанса пользователя сохраняются в постоянном хранилище вроде *Redis*, *MongoDb*, *HazelCast* и т. д).

Spring Integration – модуль предназначен для упрощенного обмена сообщениями в приложениях на основе Spring и поддержки интеграции с внешними системами через декларативные адаптеры. Эти адаптеры обеспечивают более высокий уровень абстракции по сравнению с поддержкой Spring для удаленного взаимодействия, обмена сообщениями и планирования.

Весна идёт – весне дорогу!

Другие компоненты Spring

TEL-RAN
by Starta Institute

Spring REST – предоставляет богатый набор инструментов, упрощающий разработку *REST* API: инструменты для маршрутизации запросов, для преобразования *JSON/XML* в объекты требуемых типов и т.д.

Spring Web Flow – основан на *Spring MVC* и позволяет реализовать «потоки» веб-приложения. Такие потоки инкапсулируют последовательность шагов, которые направляют пользователя через выполнение некоторой бизнес-задачи.Они охватывают несколько HTTP-запросов, имеют состояние, работают с транзакционными данными, могут использоваться повторно и могут быть динамическим и долговременным по своей природе.

Spring WebServices – призван облегчить разработку сервисов *SOAP* на основе контрактов, позволяя создавать гибкие веб-сервисы, используя один из многих способов манипулирования полезными нагрузками XML.

Весна идёт – весне дорогу!

Другие компоненты Spring

TEL-RAN
by Starta Institute

Spring HATEOAS – предоставляет некоторые API-интерфейсы для упрощения создания *REST* контроллеров, которые следуют принципу *HATEOAS* при работе со Spring и особенно *Spring MVC*. **HATEOAS** – Hypermedia As The Engine Of Application State – Гипермедиа как двигатель состояния приложения.

Spring Batch – данный модуль предоставляет функционал для пакетных обработок данных (когда данные обрабатываются большими кусками – пакетами), жизненно важных для повседневной работы корпоративных систем. Предоставляет функции многократного использования, которые необходимы для обработки больших объемов записей, включая ведение журнала / трассировку, управление транзакциями, статистику обработки заданий, перезапуск заданий, пропуск и управление ресурсами.

Spring AMQP – применяет основные концепции Spring к разработке решений обмена сообщениями на основе *AMQP* – Advanced Message Queueing Protocol – Расширенный протокол очереди сообщений. Содержит реализацию для *RabbitMQ*.

Весна идёт – весне дорогу!

Другие компоненты Spring

TEL-RAN
by Starta Institute

Spring for Apache Kafka – применяет основные концепции Spring к разработке решений для обмена сообщениями на основе *Kafka*.

Spring CredHub – предоставляет поддержку на стороне клиента для хранения, получения и удаления учетных данных с сервера *CredHub*, предоставляющего API для безопасного хранения, создания, извлечения и удаления учетных данных различных типов.

Spring FLO – библиотека *JavaScript*, которая предлагает простой встраиваемый визуальный конструктор *HTML5* для конвейеров и простых графиков для мониторинга потоковой и пакетной передачи данных.

Spring LDAP – библиотека упрощает операции по протоколу *LDAP.*

Spring Roo – модуль предоставляет собой инструмент RAD, который может создавать и управлять вашим приложением на основе Spring. Его цель – повысить производительность разработчиков Java. Он не может написать бизнес-логику вашего приложения, но может обрабатывать конфигурационные и инфраструктурные вещи.

Весна идёт – весне дорогу!

Другие компоненты Spring

TEL-RAN
by Starta Institute

Spring Shell - позволяет легко создавать полнофункциональное приложение оболочки (также известное как командная строка), полагаясь на jar-файлы *Spring Shell* и добавляя свои собственные команды (которые поступают как методы в *Spring beans*). Создание приложения командной строки может быть полезно, например, для взаимодействия с REST API вашего проекта или для работы с локальным содержимым файла.

Spring Statemachine – позволяет разработчикам приложений использовать концепции конечного автомата с приложениями Spring.

Spring Vault – предоставляет знакомые абстракции Spring и поддержку на стороне клиента для доступа, хранения и отзыва секретов. Он предлагает как низкоуровневые, так и высокоуровневые абстракции для взаимодействия с *Vault*, освобождая пользователя от проблем с инфраструктурой.

Весна идёт – весне дорогу!

Другие компоненты Spring

TEL-RAN
by Starta Institute

Весна идёт – весне дорогу!

Компоненты Spring для Junior-разработчика

TEL-RAN
by Starta Institute

Обязательно:

- Spring Core
- Spring Boot
- Spring Web (MVC и REST)
- Spring Data (Spring ORM, Spring JDBC, Spring JPA)
- Spring Test

Желательно:

- Spring Security
- Spring AOP
- Spring

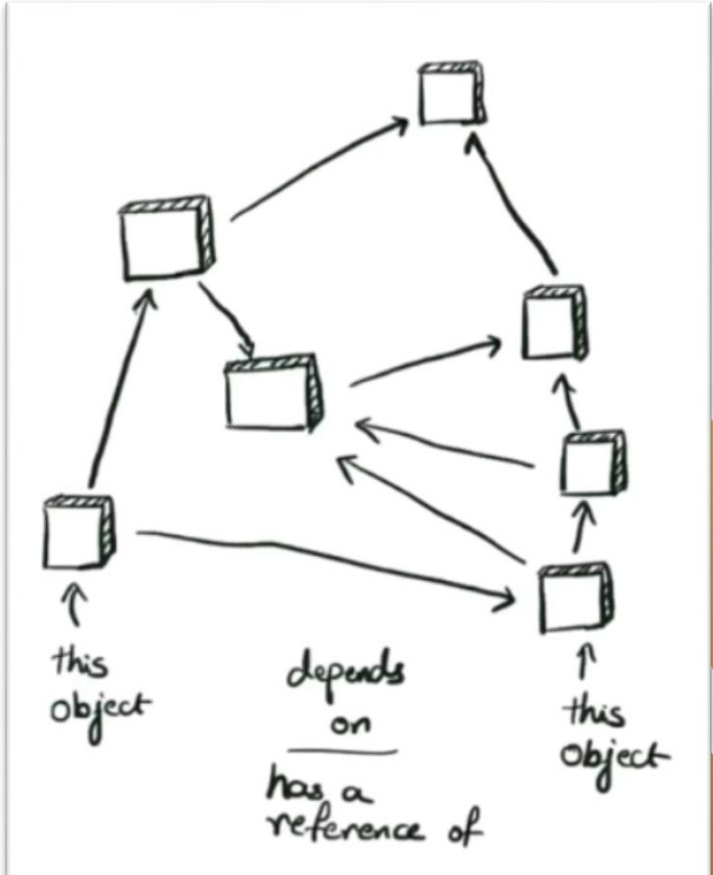
Проблема

TEL-RAN
by Starta Institute

Проект состоит из классов и интерфейсов. Классы взаимосвязаны друг с другом (*наследование*, *композиция*, *агрегация*). Все связи классов называют **зависимостями**. Если класс А зависит от класса Б это означает, что объект А не может выполнить свою работу, пока не будет создан объект Б.

Для уменьшения связности классов SOLID рекомендует использовать промежуточные интерфейсы. Этот подход приводит к тому, что классы зависят от интерфейсов, а не от конкретных реализаций. Но такой подход приводит к росту количества сущностей в приложении.

Есть ли способ автоматизировать уменьшение связности и не следить за созданием связанных объектов?



Весна идёт – весне дорогу!

Spring Core

TEL-RAN
by Starta Institute

Spring Core (Core container) выполняет базовый функционал *Spring Framework*, реализующий понятия *Inversion of control (IoC)* и *dependency injection (DI)*.

Весна идёт – весне дорогу!

Inversion of control

TEL-RAN
by Starta Institute

Inversion of control (IoC) – это абстрактный принцип, набор рекомендаций для написания слабо связанного кода. Суть принципа в том, что каждый компонент системы должен быть как можно более изолированным от других, не полагаясь в своей работе на детали конкретной реализации других компонентов.

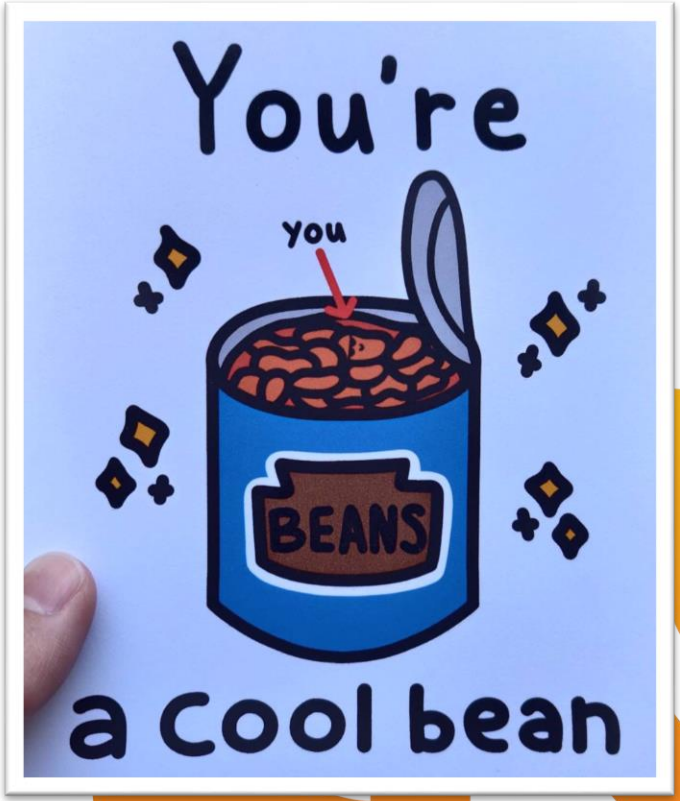
Весна идёт – весне дорогу!

Бины

TEL-RAN
by Starta Institute

Spring Core самостоятельно создаёт интерфейсы между зависимыми классами, либо предоставляет такие интерфейсы для имплементации, если речь идёт о классах, подставляемых в точки расширения фреймворка. *Spring Core* следит за тем, какие объекты соответствуют созданным интерфейсам и помечает их как кандидатов для использования.

Найденные объекты-кандидаты называются **бинами (bean)**. Все бины создаются *Spring Core* централизованно и помещаются в «хранилище бинов», которое называется **контекст (context)**. Программист не контролирует жизненный цикл бина (не создаёт, не уничтожает). Бины имеют уникальное имя в контексте.

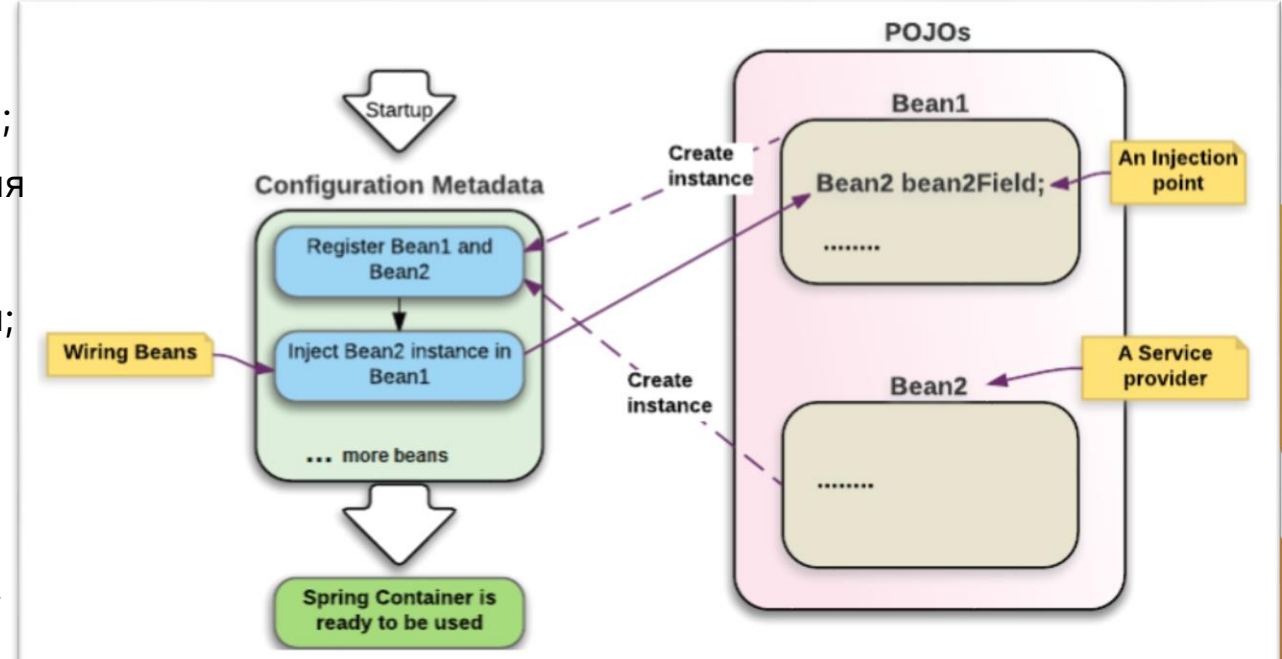


Весна идёт – весне дорогу!

Dependency injection

Dependency injection (DI) – шаблон проектирования, при котором мы указываем в интерфейсе нашего компонента (через конструктор или сеттер), от чего тот зависит. *DI* помогает реализовать принцип *IoC*, обеспечивающий:

- отделение выполнения задачи от ее реализации;
- упрощение переключения между реализациями;
- модульность программы;
- простое тестирование программы благодаря, изоляции компонент или имитации зависимостей.

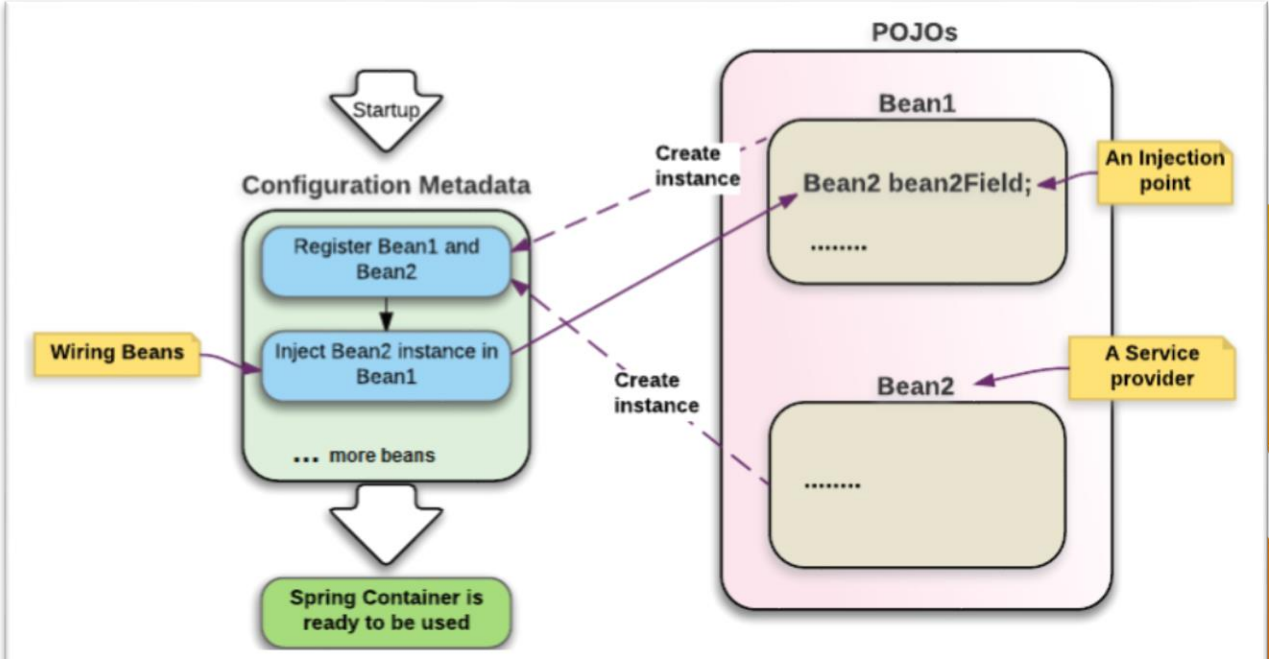


Весна идёт – весне дорогу!

Dependency Lookup

TEL-RAN
by Starta Institute

Другой паттерн – **Dependency Lookup (DL)** – применяется реже: в компонент передают только контекст (*ApplicationContext*), и мы сами посредством *ApplicationContext.getBean(...)* получаем нужные зависимости. При этом мы не указываем в интерфейсе, что конкретно будем получать.



Весна идёт – весне дорогу!

DI-container

DI-container (он же **IoC-контейнер**) – это классы, реализующие IoC в паре с DI в *Spring Core*. По сути, это программный контейнер, предоставляющий настраиваемый *контекст приложения* (пакет *org.springframework.context*), в котором создаются, инициализируются, кэшируются и управляются подключаемые объекты, известные как *bean-компоненты* или просто *бины* (пакет *org.springframework.beans*). Обычно бины реализуют бизнес-логику.

Весна идёт – весне дорогу!

DI-container

TEL-RAN
by Starta Institute

В Spring есть два различных вида контейнеров:

1. Spring *BeanFactory* Container;
2. Spring *ApplicationContext* Container;

Интерфейс **BeanFactory** предоставляет механизм конфигурирования бинов (обеспечивает базовую функциональность – создание, уничтожение, связи зависимостей).
ApplicationContext является интерфейсом-наследником *BeanFactory* и добавляет простой способ интеграции с *Spring AOP*, т.е. позволяет внедрять зависимости с помощью аннотаций.
Платформа Spring предоставляет несколько реализаций интерфейса *ApplicationContext:*
ClassPathXmlApplicationContext позволяет настроить приложение с помощью xml-файла.
Расширение **WebApplicationContext** настраивает контекст для работы веб-приложения.

Весна идёт – весне дорогу!

Доступность бинов

TEL-RAN
by Starta Institute

Spring позволяет внедрить подходящий бин автоматически. Но если подходящего бина в контексте нет, то контекст не начнёт свою работу (жарг. не поднимется) и выбросит исключение *NoSuchBeanDefinitionException*.

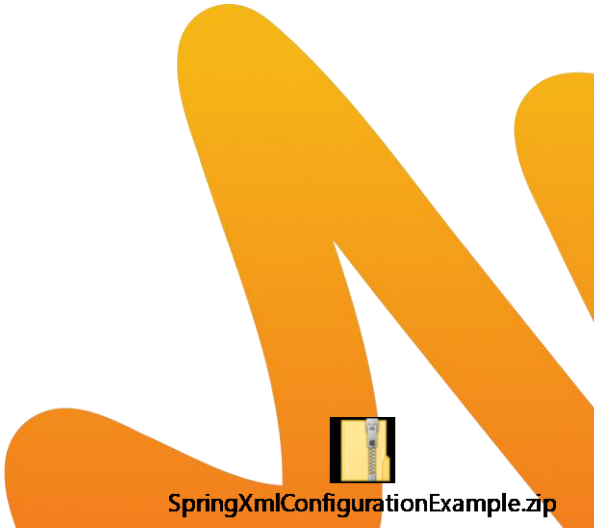
Если *Spring core* обнаружит два подходящих бина, то будет брошено исключение *NoUniqueBeanDefinitionException*. В этом случае необходимо настроить условие создания того или иного бина в конфигурации (напр., *@ConditionalOnMissingBean*, *@ConditionalOnProperty*) или указать имя бина в точках инъекции с помощью *@Qualifier*.

Задание

TEL-RAN
by Starta Institute

Создайте Spring-приложение с xml-конфигурацией. Создайте один бин и получите его из контекста. Что будет, если создать второй бин с другим именем из того же класса?

Создайте второй бин, который зависит от первого. Сделайте инъекцию через конструктор. Создайте третий бин, который зависит от первых двух. Сделайте инъекцию через сеттеры.



SpringXmlConfigurationExample.zip

Весна идёт – весне дорогу!

Область видимости бинов

По умолчанию видимость (*scope*) всех бинов указывается как **Singleton**, т.е. по первому запросу создаётся один бин и далее переиспользуется во всех точках внедрения.

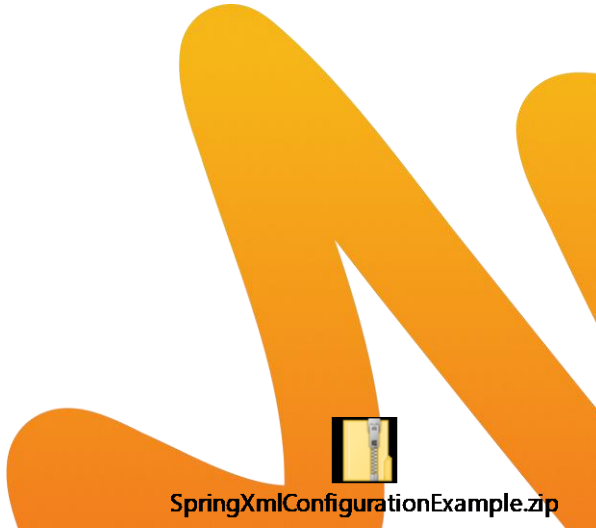
При необходимости можно поменять *scope* бина на **Prototype**. Тогда при каждом запросе из контекста будет выдан новый экземпляр бина. В этом случае контекст перестаёт контролировать жизненный цикл бина. Уничтожение бина становится задачей разработчика.

В *Spring MVC* также есть видимости **Request**, **Session** и **GlobalSession**, но о них поговорим позже.

Задание

TEL-RAN
by Starta Institute

Создайте бин с областью видимости Singleton. Достаньте из контекста два бина и сравните их по ссылкам. Повторите операцию для бинов с областью видимости Prototype.



SpringXmlConfigurationExample.zip

TEL-RAN
by Starta Institute

3 Домашнее задание

Домашнее задание

TEL-RAN
by Starta Institute

1.1 Создайте класс Customer с полями id, name, dateOfLastNotification, phone, email, геттрами и сеттерами. В приложении создайте два бина типа Customer (используйте scope Prototype). Получите бины из контекста и установите первому параметры {1L, “Bob”, LocalDate.now(), “+19138445656”, null }, второму {2L, “Sarah”, 2024-16-01 , “+19158455617”, ”sarah-sweet@candy.com”}.

1.2 Создайте класс NotificationService, который выполняет оповещение клиентов с помощью метода sendSpam(Customer customer): если с последней отправки прошло больше 30 дней, то он отправляет спам по имеющимся контактам (метод делегирует эту работу классам от которых зависит - SmsNotificationService и EmailNotificationService). Получите NotificationService из контекста и передайте ему бины клиентов. SmsNotificationService и EmailNotificationService должны внедряться в NotificationService.

1.3 Создайте класс SmsNotificationServiceImpl, следующий интерфейсу SmsNotificationService, и класс EmailNotificationServiceImpl, следующий интерфейсу EmailNotificationService.

ЗАКЛЮЧЕНИЕ

TEL-RAN
by Starta Institute