Java Pro 7.3

TEL-RAN
by Starta Institute

Введение в Spring (часть 3)

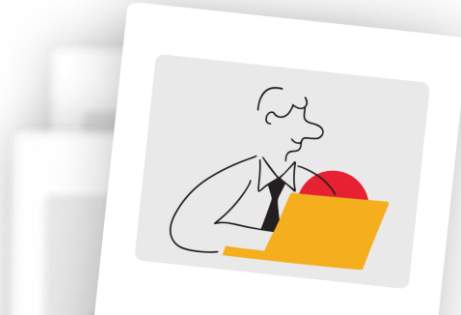
ПРЕПОДАВАТЕЛЬ

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

Что нужно делать, чтобы создать бин Вашего класса *CarUtility* в Spring-приложении?

```
class CarUtility {
    // ...
}
```

Повторение

TEL-RAN
by Starta Institute

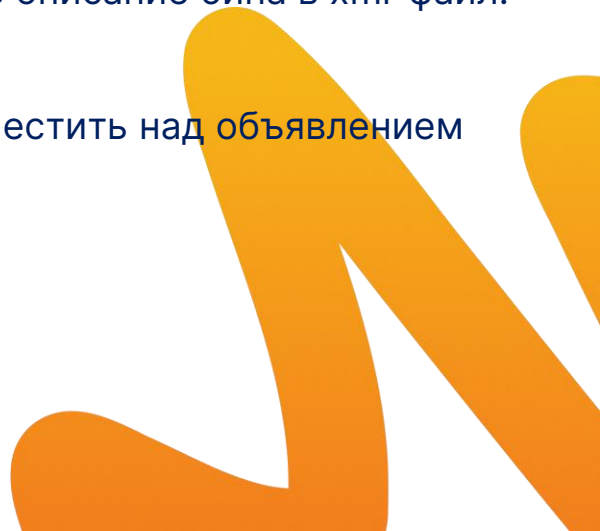
Что нужно делать, чтобы создать бин Вашего класса *CarUtility* в Spring-приложении?

```
@Component
class CarUtility {
    // ...
}
```

Если конфигурация Spring выполнена на основе xml, то добавить описание бина в xml-файл.

Либо можно указать в конфигурации тег

<context:component-scan base-package=«package.name"/> и поместить над объявлением класса аннотацию *@Component*.



Повторение

TEL-RAN
by Starta Institute

Улучшите код

```
@Component
public class CustomerService {
    private CustomerRepository customerRepository;
    private Logger logger;

    public CustomerService(CustomerRepository customerRepository, Logger logger) {
        this.customerRepository = customerRepository;
        this.logger = logger;
    }
    @Autowired
    public void setCustomerRepository(CustomerRepository customerRepository) {
        this.customerRepository = customerRepository;
    }
    @Autowired
    public void setLogger(Logger logger) {
        this.logger = logger;
    }
}
```

Повторение

TEL-RAN
by Starta Institute

Улучшите код

```
@Component
public class CustomerService {
    private CustomerRepository customerRepository;
    private Logger logger;

@Autowired
    public CustomerService(CustomerRepository customerRepository, Logger logger) {
        this.customerRepository = customerRepository;
        this.logger = logger;
    }
}
```

Вместо внедрения бинов в сеттеры гораздо короче и безопасней внедрять их в конструктор, если, конечно, отсутствие сеттеров не противоречит логике работы класса.

Повторение

Исправьте ошибку в коде

TEL-RAN
by Starta Institute

```
@Component
class Service {
    @Autowired
    public Service(Repository rep){}
   }

interface Repository {}

@Component
class DBRepository implements Repository {}

@Component
class XmlRepository implements Repository {}
```

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

```
@Component
class Service {
    @Autowired
    public Service(@Qualifier("dBRepository") Repository rep){}
}

interface Repository {}

@Component
class DBRepository implements Repository {}

@Component
class XmlRepository implements Repository {}
```

Когда в контексте существует несколько бинов, следующих одному интерфейсу, нужно указывать, какой бин требуется внедрить с помощью *@Autowired*.

Повторение

TEL-RAN
by Starta Institute

Исправьте ошибку в коде

```
@Component
public class PaymentOrder {
    // Создаёт шаблон платёжного поручения с уникальным номером
}

@Component
public class OrderProcessingService {

    public PaymentOrder fillInOrder() {
        PaymentOrder order = context.getBean(PaymentOrder.class);
        // Логика заполнения платёжного поручения
        return order;
    }
}
```

Повторение

TEL-RAN
by Starta Institute

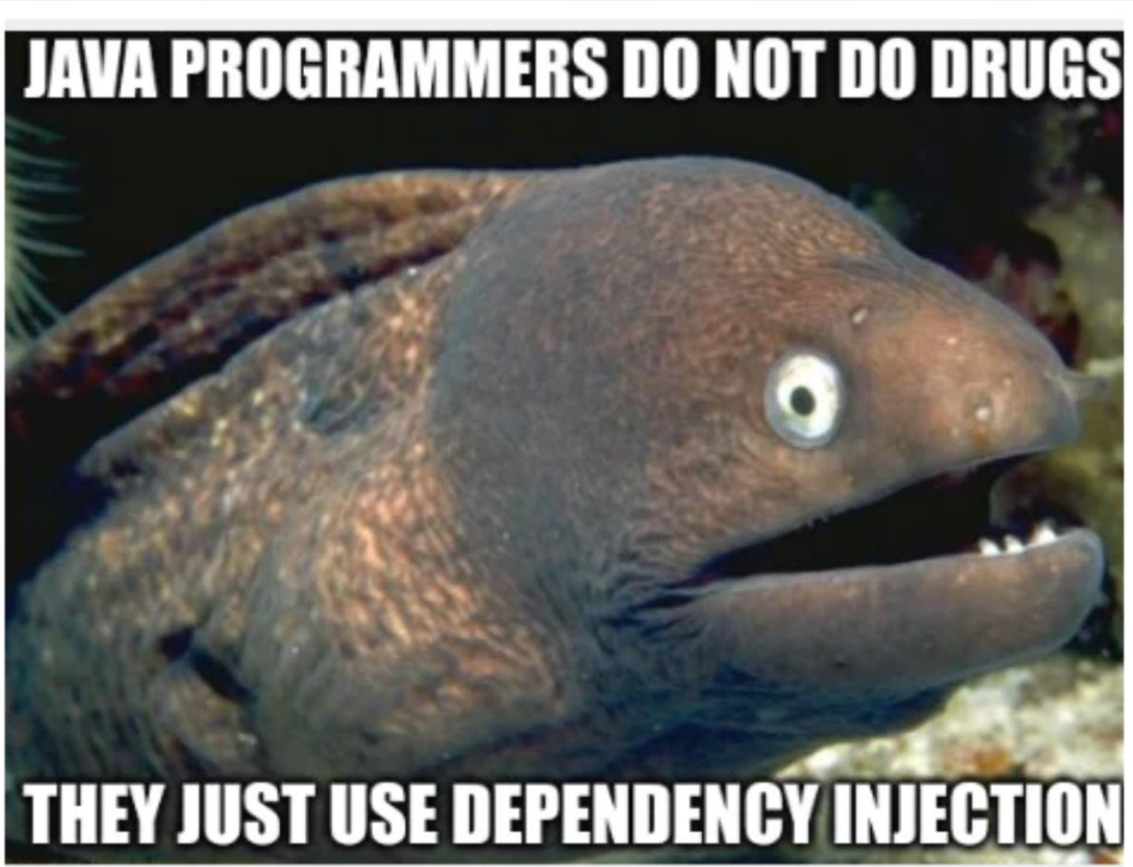
Исправьте ошибку в коде

```
@Component
@Scope("prototype")
public class PaymentOrder {
    // Создаёт шаблон платёжного поручения с уникальным номером
}

@Component
public class OrderProcessingService {

    public PaymentOrder fillInOrder() {
        PaymentOrder order = context.getBean(PaymentOrder.class);
        // Логика заполнения платёжного поручения
        return order;
    }
}
```

Для того, чтобы новое платёжное поручение каждый раз, нужно, чтобы бин PaymentOrder имел область видимости prototype.

Повторение

В чём прикол мема?



TEL-RAN
by Starta Institute

2 ОСНОВНОЙ БЛОК

Введение

TEL-RAN
by Starta Institute

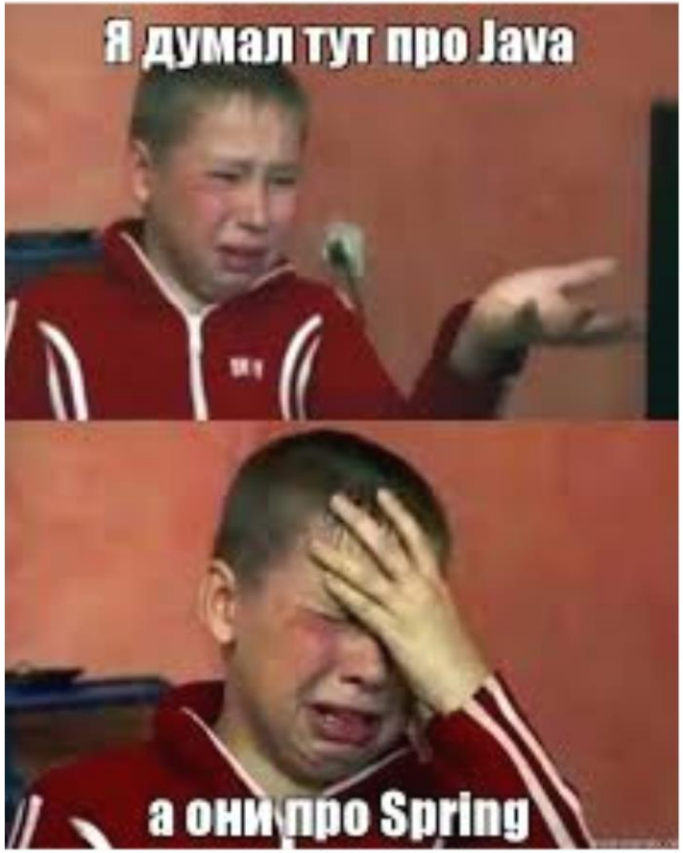
- Придаём значение
- Вмешательство с @
- Великий исход из xml
- Чужие бины
- Первый среди равных
- На кого можно положиться
- Лентяй

Проблема

TEL-RAN
by Starta Institute

Ранее мы узнали, что приложения Java имеют возможность получать настройки из файла с помощью стандартного механизма.

Позволяет ли использование Spring автоматизировать этот процесс?



Придаём значение

Внедрение значений настроек

TEL-RAN
by Starta Institute

При создании бина можно указать конкретные значения, передаваемые в поля объекта.

Кроме того, эти значения можно взять из файла настроек.

1 Необходимо создать в ресурсах файл с типом “.properties”. Например, *application.properties*:

```
db.url="www.google.com"
db.username="admin"
db.password="keep1Secret!"
```

Придаём значение

Внедрение значений настроек

TEL-RAN by Starta Institute

2 В xml-файле конфигурации нужно указать (обратите внимание на расшыренный тег <beans):

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
          http://www.springframework.org/schema/context
          http://www.springframework.org/schema/context/spring-context.xsd">
    <!-- Указание файла настроек, значения из которого будут подставляться в плейсхолдеры -->
    <context:property-placeholder location="classpath:application.properties"/>
    <!-- Описание бина с плейсхолдерами -->
    <bean id="databaseConfig" class="org.example.props.DatabaseConfig">
        <constructor-arg name="url" value="${db.url}"/>
        <constructor-arg name="username" value="${db.username}"/>
        <constructor-arg name="password" value="${db.password}"/>
    </bean>
</beans>
```

Придаём значение
@Value

TEL-RAN
by Starta Institute

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context.xsd">
    <!-- Поиск компонентов -->
    <context:component-scan base-package="org.example.pojo"/>
    <!-- Указание файла настроек, значения из которого будут подставляться в плейсхолдеры -->
    <context:property-placeholder location="classpath:application.properties"/>
</beans>
```

```
@Component
public record DatabaseConfig(
@Value("${db.url}") String url,
@Value("${db.username}") String username,
@Value("${db.password}") String password) { }
```

Если класс для хранения настроек помечен *@Component*, то для внедрения значений из файла настроек используется аннотация **@Value**.

Задание

TEL-RAN
by Starta Institute

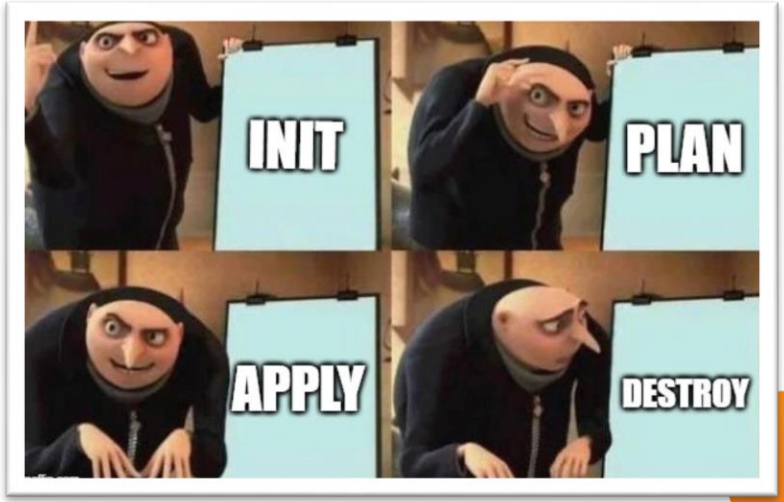
Создайте Spring-приложение, которое получает настройки *host*, *port* и *token* из файла *application.properties*, используя *@Value*.

Проблема

TEL-RAN
by Starta Institute

С переносом из xml-конфигурации в аннотации внедрения зависимостей, внедрения значений из файла настроек понятно, но ранее в xml-конфигурации мы ещё управляли жизненным циклом бина.

Как управлять жизненным циклом бина с помощью аннотаций?



Вмешательство с @PostConsruct и @PreDestroy

Для вмешательства в жизненный цикл бина с помощью аннотаций нужно поместить в него метод, выполняемый после инициализации, и пометить его **@PostConstruct**, и/или метод, выполняемый перед уничтожением бина, и пометить его **@PreDestroy**.

```
@Component
public class FirstComponent {
    private static final String PREFIX = "FirstComponent: ";

    public FirstComponent() {
        System.out.println(PREFIX + "Создание бина");
    }

    @PostConstruct
    public void start() {
        System.out.println(PREFIX + "После создания бина");
    }
    @PreDestroy
    public void finish() {
        System.out.println(PREFIX + "Перед удалением бина");
    }

    @Override
    public String toString() {
        return "I'm the FirstComponent made using @Component";
    }
}
```

Вмешательство с @

@PostConsruct и @PreDestroy

@PostConsruct и *@PreDestroy* являются частью *JEE*, а не Spring, поэтому нужно добавить в систему сборки соответствующую зависимость. Например, для maven:

```
<dependency>
    <groupId>jakarta.annotation</groupId>
    <artifactId>jakarta.annotation-api</artifactId>
    <version>3.0.0-M1</version>
</dependency>
```

Вмешательство с @

Пример жизненного цикла бина

TEL-RAN
by Starta Institute

SpringAnnotationLifecucleExample.zip

Задание

TEL-RAN
by Starta Institute

В ранее созданном приложении залогируйте настройки после их получения из файла. Во время работы программы токен должен поменяться (сгенерируйте с помощью UUID). Перед уничтожением бина замените токен в файле конфигурации на значение, хранящееся в бине.

Проблема

TEL-RAN
by Starta Institute

Ранее мы настраивали Spring с помощью

- xml-файла
- xml-файла и аннотаций

Тенденция подсказывает, что аннотации позволяют нам сильно сократить объём кода в xml-файле.

Может, стоит полностью отказаться от него? Тем более, что в наше время он уже непопулярен из-за многословности.

Великий исход из xml

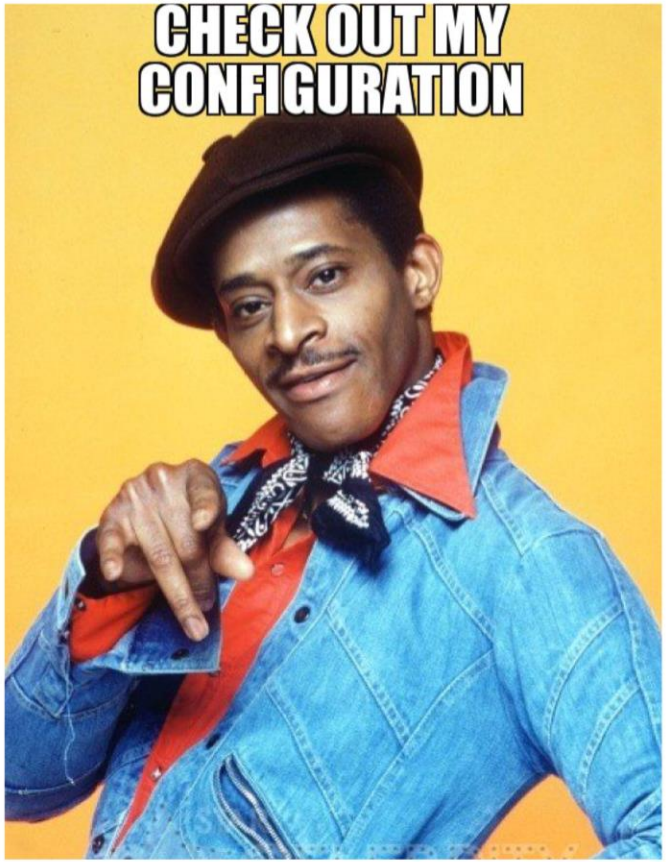
@Configuration

TEL-RAN
by Starta Institute

Для настройки Spring без xml потребуется использовать конфигурацию с помощью Java-кода и аннотаций.

Для этого необходимо создать один или несколько классов конфигурации. Они помечаются аннотацией **@Configuration**.

```
@Configuration
public class MyAppConfig {
}
```



Великий исход из xml
@Configuration

TEL-RAN
by Starta Institute

Каждый тег xml превращается в аннотацию конфигурационного класса:

```
<context:component-scan base-package="org.example.pojo"/>
```

```
@Configuration
@ComponentScan("org.example.pojo")
public class MyAppConfig { }
```

```
<context:property-placeholder location="classpath:application.properties"/>
```

```
@Configuration
@ComponentScan("org.example.pojo")
@PropertySource("classpath:application.properties")
public class MyAppConfig { }
```

Великий исход из xml

AnnotationConfigApplicationContext

Вместо ранее использованного класса *ClassPathXmlApplicationContext* нужно использовать класс **AnnotationConfigApplicationContext**.

Конструктор	Описание
AnnotationConfigApplicationContext()	Создаёт пустой контекст, который нужно заполнить бинами с помощью метода *register(java.lang.Class<?>...)* и обновить вручную методом *refresh()*
AnnotationConfigApplicationContext(Class<?>... componentClasses)	Создаёт контекст, заполненный бинами указанных классов.
AnnotationConfigApplicationContext(String... basePackages)	Создаёт контекст, заполненный бинами классов, найденными в указанных пакетах и помеченных *@Component*
AnnotationConfigApplicationContext(DefaultListableBeanFactory beanFactory)	Создаёт контекст с переданным экземпляром *DefaultListableBeanFactory*

Великий исход из xml

AnnotationConfigApplicationContext

Если вместо классов-компонентов в конструктор *AnnotationConfigApplicationContext* был передан класс, помеченный @Configuration, то в контекст попадут все бины указанной конфигурации

```
@Configuration
@ComponentScan("org.example.pojo")
public class MyAppConfig {
```

```
public static void main(String[] args) {
    AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(MyAppConfig.class);
    FirstComponent fc = context.getBean("firstComponent", FirstComponent.class);
    System.out.println(fc);
    context.close();
}
```

Программно можно выполнить сканирование бинов с помощью метода *scan()*:

```
context.scan("org.example.pojo");
```

Задание

TEL-RAN
by Starta Institute

Переведите ранее созданное приложение на конфигурацию без использования xml.

Создайте класс Connector, который использует бин настроек приложения (host, port и token). В классе создайте метод, эмулирующий подключение к удалённому серверу.

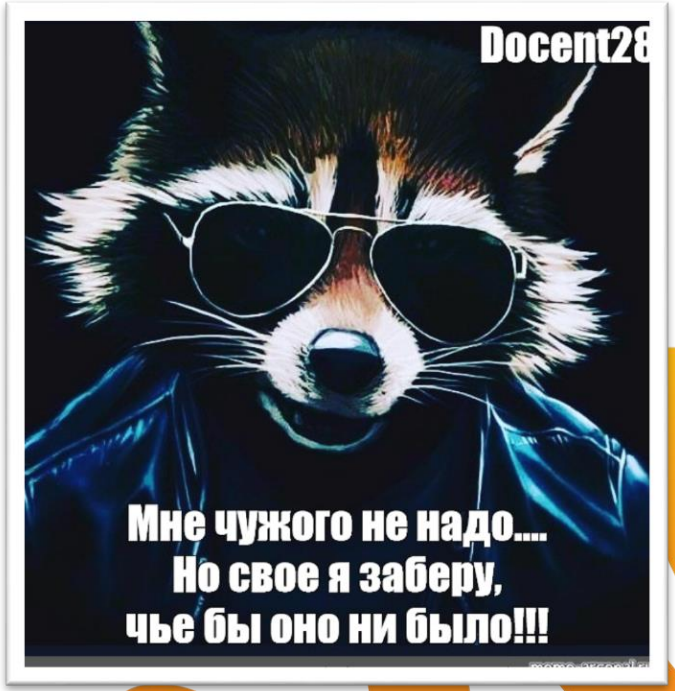
Получите бин Connector из контекста и запустите метод подключения.

Проблема

TEL-RAN
by Starta Institute

Мы уже умеем создавать бины собственных классов без использования xml-файла. Но чаще всего мы используем чужие классы в программе – будь то классы стандартной библиотеки Java или классы подключаемых библиотек. Например, объект класса *ObjectMapper* библиотеки *Jackson* – достаточно затратный с точки зрения ресурсов объект. Его бы не мешало поместить в контекст как синглтон, чтобы пользоваться только одним экземпляром и иметь доступ к нему через контекст из любой точки приложения.

Как создать бин чужого класса, который не помечен @Component?



Чужие бины
@Bean

TEL-RAN
by Starta Institute

Для создания бинов из любых классов (чужих или своих) можно использовать классы, помеченные *@Configuration*, а методы в них, возвращающие экземпляры интересующих нас классов, помечаются аннотацией **@Bean**.

```
@Configuration
public class MyAppConfig {
    @Bean
    public Random random() {
        return new Random();
    }
}
```

При необходимости в настройках аннотации *@Bean* можно указать имя бина, можно ли его использовать для автоматического внедрения (*@Autowired*), а также *init-* и *destroy*-методы.

```
@Bean(
        name = "first",
        autowireCandidate = false,
        initMethod = "start",
        destroyMethod = "finish"
)
public FirstBean firstBean() {
    return new FirstBean();
}
```

Чужие бины
@Bean

TEL-RAN
by Starta Institute

Внедрение бинов, полученных из методов с @Bean, в компоненты можно делать автоматизированно с помощью *@Autowired*.

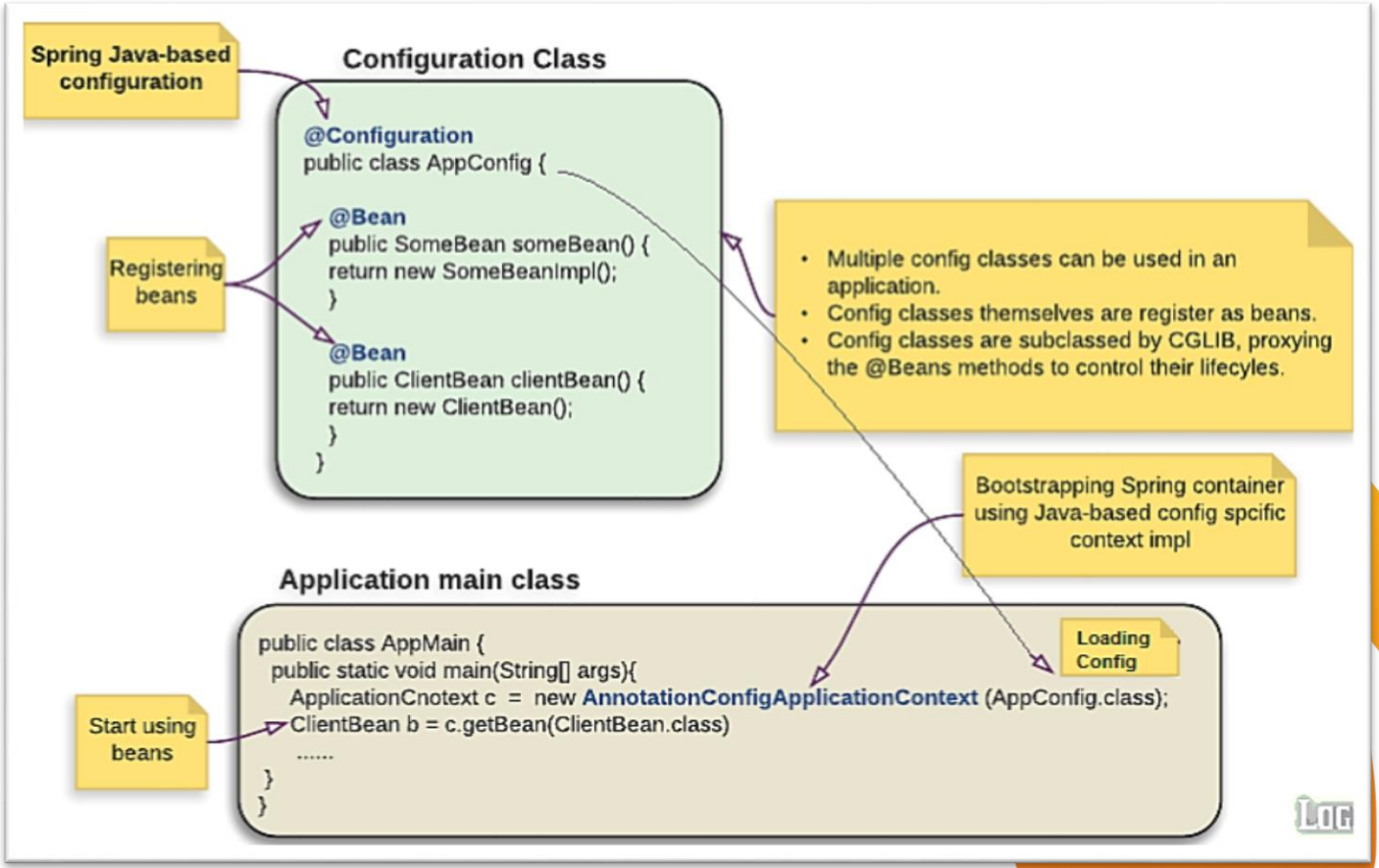
Тем не менее, можно внедрять бины и вручную в конфигурационном файле, вызывая соответствующие методы, порождающие бины.

```
public class SecondBean {
    private final FirstBean first;

    public SecondBean(FirstBean first) {
        this.first = first;
    }
}
```

```
@Configuration
@ComponentScan("org.example.pojo")
public class MyAppConfig {
    @Bean(
            name = "first",
            autowireCandidate = false,
            initMethod = "start",
            destroyMethod = "finish"
    )
    public FirstBean firstBean() {
        return new FirstBean();
    }

    @Bean
    public SecondBean secondBean() {
        return new SecondBean(firstBean());
    }
}
```

Чужие бины

Контекст и бины



Чужие бины

Пример использования @Bean

TEL-RAN
by Starta Institute

SpringAnnotationConfigurationExample.zip

Задание

TEL-RAN
by Starta Institute

Создайте бин Снег в классе конфигурации. Внедрите бин Снег в бин класса Зима, имплементирующий интерфейс Сезон.

Проблема

TEL-RAN
by Starta Institute

Возвращаясь к ситуации, когда несколько классов являются наследниками одного предка или имплементируют один и тот же интерфейс может возникнуть необходимость уточнить, какой бин внедрять с помощью *@Autowired*, но такой подход требует в каждом классе указывать *@Qualifier*.

Можем ли мы задать бин по умолчанию, чтобы выбрать первого среди равных и не указывать @Qualifier для этого бина?

Первый среди равных

@Primary

Если существует несколько бинов, следующих одному интерфейсу, то приоритетный для автовнедрения бин можно задать с помощью аннотации **@Primary**.

```
@Bean
public ObjectMapper objectMapper() {
    JavaTimeModule module = new JavaTimeModule();
    return new ObjectMapper()
            .setSerializationInclusion(JsonInclude.Include.NON_NULL)
            .registerModule(module);
}

@Bean
@Primary
public ObjectMapper objectMapperWithDateTimeModule() {
    JavaTimeModule module = new JavaTimeModule();
    ObjectMapper mapper = new ObjectMapper()
            .setSerializationInclusion(JsonInclude.Include.NON_NULL)
            .registerModule(module);
    DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm a z");
    mapper.setDateFormat(df);
    return mapper;
}
```

Первый среди равных
@Primary

TEL-RAN
by Starta Institute

При этом возможность внедрять нужный бин с помощью *@Qualifier* остаётся.

```
@Autowired
public UsualObjectMapperHolder(@Qualifier("objectMapper") ObjectMapper mapper) {
    this.mapper = mapper;
}
```

Первый среди равных

Пример использования @Primary

TEL-RAN
by Starta Institute

SpringAnnotationPrimaryExample.zip

Задание

TEL-RAN
by Starta Institute

Дополните предыдущее задание. Создайте также бины классов Лето, Весна, Осень, имплементирующие интерфейс Сезон. Создайте бин Текущий сезон, который хранит текущий сезон года (когда бы ни запускалась программа, сезон должен быть правильный). Создайте бин Туристический сезон – каждый новый туристический сезон должен начинаться с Зимы (используйте @Primary). Создайте бин класса Год и внедрите в него все 4 сезона года.

Проблема

TEL-RAN
by Starta Institute

Хотя автоматическое внедрение зависимостей через *@Autowired* позволяет *Spring* автоматически управлять зависимостями, порядок инициализации не всегда гарантирован. Это особенно важно в случаях, когда порядок инициализации имеет значение для корректной работы приложения.

Как придать инициализации бинов нужный порядок?

На кого можно положиться

@DependsOn

TEL-RAN
by Starta Institute

Аннотация **@DependsOn** используется для определения порядка инициализации бинов, когда бин зависит от других бинов и требует, чтобы эти зависимости были полностью инициализированы до его собственной инициализации.

@DependsOn может задавать как зависимость от времени инициализации, так и, в случае только бинов-одиночек, соответствующую зависимость от времени уничтожения. Зависимые бины, которые определяют *@DependsOn*-отношения с данным бином, уничтожаются в первую очередь, вплоть до уничтожения самого данного бина.

@DependsOn используется только совместно с component-scanning. При xml-конфигурации бина метаданные *@DependsOn* игнорируются.

На кого можно положиться

@DependsOn

В примере бин *fileProcessor* зависит от бинов *fileReader* и *fileWriter.* Если не использовать *@DependsOn* и порядок инициализации будет непредсказуемым, может возникнуть ситуация, когда *fileProcessor* инициализируется раньше, чем бины, от которых он зависит, что приведет к ошибке.

@DependsOn требует от *Spring* сперва проинициализировать *fileReader* и *fileWriter*, обеспечивая корректный порядок инициализации бинов.

```
@Configuration
@ComponentScan({"org.example.file",
                "org.example.shared"})
public class MyAppConfig {
    @Autowired
    File file;

    @Bean("fileProcessor")
    @DependsOn({"fileReader", "fileWriter"})
    public FileProcessor fileProcessor() {
        return new FileProcessor(file);
    }

    @Bean("fileReader")
    public FileReader fileReader() {
        return new FileReader(file);
    }

    @Bean("fileWriter")
    public FileWriter fileWriter() {
        return new FileWriter(file);
    }
}
```

Первый среди равных

Пример использования @DependsOn

SpringAnnotationDependsOnExample.zip

Задание

TEL-RAN
by Starta Institute

Создайте бин DataProcessor, который зависит от бина DatabaseConnection. В свою очередь DatabaseConnection не может существовать, пока база данных не будет проинициализирована бином DatabaseInitializer. Обеспечьте корректный порядок инициализации бинов.

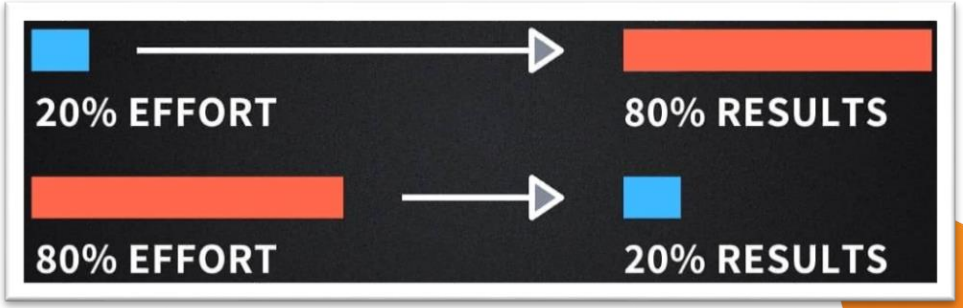
Проблема

TEL-RAN
by Starta Institute

Приложение с широкой функциональностью может содержать сотни бинов. При этом действует правило 80/20 по отношению к функционалу приложения: 20% функционала приложения используется 80% времени.

Мы знаем, что *Spring* создаёт бины на этапе инициализации контекста. Но какой смысл хранить все бины, занимая таким образом оперативную память, если большинство бинов не используется?

Как обеспечить загрузку бинов в контекст только при необходимости использования бина?



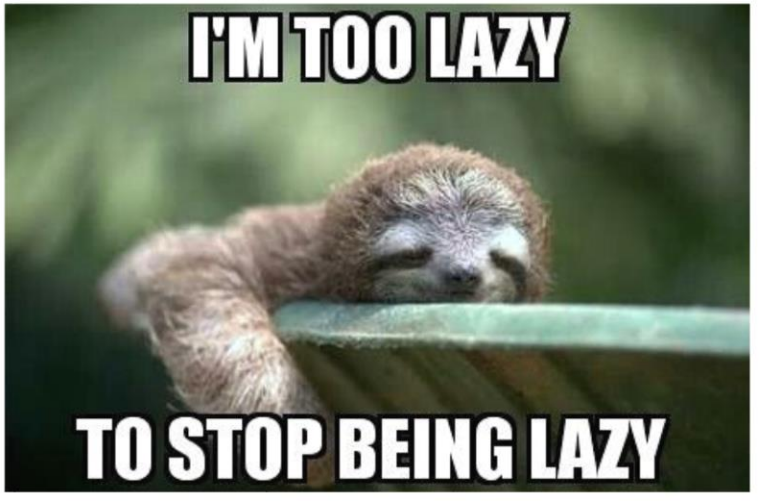
Лентяй

@Lazy

TEL-RAN
by Starta Institute

@Lazy – аннотация, которая указывает, что бин должен иметь «ленивую» инициализацию, т.е. он будет создан в контексте только после того, как какой-то из классов запросит его из контекста. Это увеличивает время получения бина из контекста, но экономит ресурсы приложения. *@Lazy* может применяться совместно с

- *@Configuration*: все методы внутри класса конфигурации, помеченный *@Bean*, будут возвращать бины с ленивой инициализацией;
- *@Bean:* помеченный метод будет возвращать бины с ленивой инициализацией;
- *@Component:* помеченный компонент будет бином с ленивой инициализацией;
- *@Autowired:* внедрение бинов будет происходить в режиме ленивой инициализации.



Первый среди равных

Пример использования @Lazy

SpringAnnotationLazyExample.zip

Задание

TEL-RAN
by Starta Institute

Допустим, у Вас есть сервис, который выполняет долгую (30 сек) операцию получения сертификатов доступа к внутренней сети предприятия при старте приложения из корпоративного сервера. Вам необходимо избежать замедления старта приложения и отложить инициализацию этого сервиса до момента его первого вызова. Сравните скорость запуска приложения с использованием @Lazy и без неё.

TEL-RAN
by Starta Institute

3 Домашнее задание

Домашнее задание

TEL-RAN
by Starta Institute

Решите задания, используя Spring и изученные аннотации.

1.1 Напишите простое приложение для управления задачами (To-Do List). Создайте бины Задач, Список задач и т.д. Размер списка задач, их приоритеты, заголовок и описание по умолчанию должны быть взяты из файла настроек.

1.2 Дополните приложение. Приложение должно отправлять уведомления о срочных задачах. Создайте интерфейс NotificationService, который будет иметь несколько реализаций для отправки уведомлений по электронной почте, SMS и push. Используйте аннотации @Component, @Qualifier, @Primary и другие для настройки внедрения зависимостей.

1.3 Приложение должно отправлять данные на сервер. Создайте бин класса Connector, который эмулирует подключение к серверу. Подключение занимает длительное время, поэтому не должно замедлять запуск программы.

ЗАКЛЮЧЕНИЕ

TEL-RAN
by Starta Institute