



ОНЛАЙН-ОБРАЗОВАНИЕ

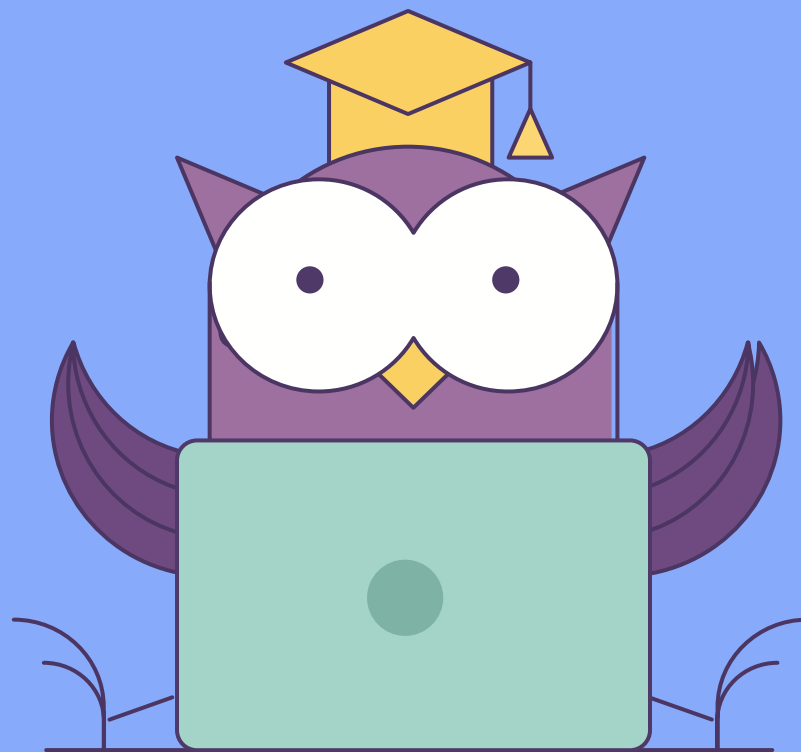


Введение в Spring Framework

Занятие № 01



Меня хорошо слышно && видно?



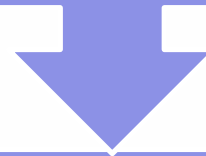
Напишите в чат, если есть проблемы!

Ставьте ☐ + если все хорошо
Ставьте ☐ - если есть проблемы

- Разобраться с организационными вопросами.
- ориентироваться в проектах Spring для дальнейшего изучения,
- применять принцип IoC при написании классов и тестов,
- создавать контекст Spring,
- определять Spring Beans в контексте,
- организовывать правильный DI

- Сегодня вводное занятие, просьба не переживать 😊
- Активно участвуем.
- Не стесняйтесь задавать вопрос в чат.
- Но off-topic обсуждаем в слаке #random.
- Я всё вижу, но на вопрос в чате могу ответить не сразу.
- Договорились?

Организационные вопросы



Spring Framework



Spring IoC



Практика



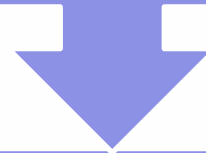
**Договорились?
Поехали!**



Организационные вопросы



Spring Framework



Spring IoC



Практика

00

Организационные вопросы

- Женат на Java, ещё с JDK 1.4;
- Написал гигабайты кода в очень больших и маленьких проектах;
- Провёл более 1860 часов курсов, тренингов и вебинаров;
- Учил и подготовил более 670 разработчиков;
- Почти побил мировой рекорд по отсутствию сна, но при этом отлично сдал проект.



- Java-разработчик в Технологическом центре **Deutsche Bank**.
- В отрасли с **2003** года. Программировать начал со времен ZX-Spectrum.
- **8 лет** в разработке банковских процессинговых систем пластиковых карт.
- Занимался разработкой "коробочных" систем для внешних заказчиков, разработкой систем для внутренних заказчиков.



Программист со страстью к обучению. Богатый опыт разработки в кровавых энтерпрайзах, гаражных стартапах и геймдеве..

Пишет на java, читает на Java и думает тоже на Java



Виталий Куценко

В 2004 году окончил Химический Факультет МГУ им М.В.Ломоносова, в 2005 - вечерний факультет МИФИ по специальности инженер-системотехник. Участвовал в проектах разных размеров и нагрузки, от небольших на несколько рабочих мест одного предприятия до обработки потоков данных крупнейших банков мира. Работал в таких компаниях как Diasoft, 1С, Luxoft, Росгосстрах, Сбербанк.



Дмитрий Коган

Был президентским стипендиатом (одним из двух лучших студентов) в университете, занимался нелинейными динамическими системами, поведением нейронов и детерминистическим хаосом, учил немецкий язык – и в возрасте 18 лет отправился на свою первую, но не последнюю полугодовую практику на немецкую фирму DASA (DaimlerChrysler Aerospace), ставшую позже фирмой EADS (European Aeronautic Defence and Space), а ещё позже – Airbus. Там ещё в 90-х в качестве web-разработчика пробовал писать апплеты на Java. Апплеты дико не взлюбил, и как показало будущее, которого у апплетов не было, был прав.

В 2000 году по окончании университета, разминувшись тогда с готовой карьерой преподавателя, остался работать в Германии. А в 2002 году вместе со всем IT-отделом перешёл в немецкое подразделение американского IT-гиганта CSC, который не так давно слился с HP Enterprise в новую фирму – DXC, купившую всем известный Luxoft.

На данный момент, является председателем совета предприятия.

Публицист. Спортивный журналист. Блогер.

Играет в “Что? Где? Когда?” (был чемпионом Германии), в компьютерные игры (на это, правда, совсем нет времени) и на скрипке (когда друзьям нужен кто-то, с кем можно сыграть дуэт).



Александр Оруджев

Инженер-программист. В отрасли с 2006 года.

Входит в состав команды, занимающейся разработкой ИС для информационного сопровождения бурения нефтяных и газовых скважин.

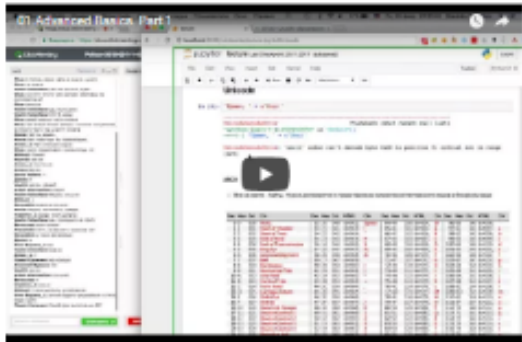


- У нас есть чат - **Telegram**.
- Кто ещё не зашёл в телегу?
- Это главный канал связи – если какие-то вопросы – задавайте прямо туда.
- Сразу предупрежу – мы из разных часовых поясов, поэтому можем ответить и ночью.

Как проходит работа – Чат с преподавателем

- Когда Вы будете сдавать домашнее задание – там будет ещё один чат.
- Вопросы по ДЗ – можете задавать туда, но для оперативности – лучше в телегу.

1 Advanced basics. Часть 1

- Описание**
Кодировки, Unicode в Python 2, coercion, Floating point numbers, IEEE 754, особенности реализации, основные ошибки и особенности использования, itertools, Iterable, Iterator, Iterator protocol, Sequence protocol, Generators, pipelines, routing, coroutines.
- Дата и время**
26 февраля, понедельник в 2020
- Опрос**
Пожалуйста, пройдите [опрос о занятии](#)
- Материалы**

- Домашнее задание**
 ДЗ-1: Log Analyzer
 (1) Написать программу на тему занятия: итераторы, генераторы, декораторы; (2) Делаем анализатор логов веб-сервера.
 Статус: не сдано [Читайте подробнее](#)

● **Домашнее задание**

ДЗ-1: Log Analyzer

(1) Несколько упражнений на тему занятия: итер

Статус: не сдано [Чат с преподавателем](#)

2 Advanced basics. Часть 2

- У Вас есть **личный кабинет** – там после вебинара будут записи.
- **Не пугайтесь**, если что-то не получилось. Темп быстрый.
- У Вас всегда будет **запись**.
- Запись буду выкладывать как можно раньше.

Файл: Инструкция «Личный кабинет»

Что там есть:

- Как найти запись вебинара
- Как сдавать домашние задания
- Как писать отзыв на занятие
- Да, это интерфейс, который требует инструкций 😊



Напишите в телегу

- Как Вас зовут;
- Ваш опыт в Java;
- Ваш опыт с Maven || Gradle;
- Ваша степень знакомства со Spring;
- Ваши ожидания от тренинга.



- **пн, 17.30 МСК, ср 17.30 МСК**
потом пн, 17.00 МСК, ср 17.00 МСК,
в среднем по 1,5-2 часа. (на самом деле 2 часа 😊)
ДЗ примерно через одно занятие
(поначалу будет чаще)

- Вебинары – это не только лекции, это и практика!
- Мы будем работать не только дома, но и на занятиях.
- Если Вы смотрите в записи – тоже практикуйтесь.

А сейчас приготовьте git, IntelliJ IDEA

Файл: Настройка окружения.pdf

Что там есть:

- Что мы работаем на Java 11
- Как настроить Git, IntelliJ IDEA и прочее
- Как мы делаем pull-request-ы для ДЗ в GitHub/GitLab/Bitbucket 😊



- Дедлайна нет! Ну кроме окончания курса 😊
- Но если опоздаете – будет проверена позже, после остальных
- Срок нашей проверки работы – 2 дня, но мы, обычно, проверяем быстрее, а иногда медленнее 😊
- Можно и делать задания раньше, чем мы их дадим, но будьте готовы переделать 😊

- **Введение**
- ключевые функциональности – IoC, AOP, Spring Boot
- **Работа с базами данных**
- Spring JDBC, Spring ORM, Транзакции, Spring Data
- **Разработку Web-приложений**
- Spring MVC
- **«Около» и «Дзен»**
- Spring Security, Spring Integration, Монолиты, Микросервисы, Docker, Облака и всё что с ними связано.
- **Проектная работа (и ещё 4 «занятия»)**

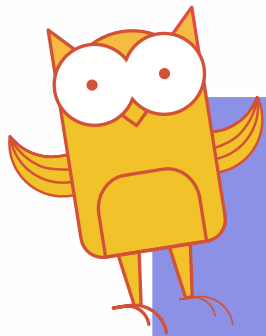
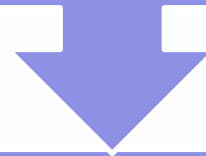




1. Введение в Spring Framework
2. Введение в Spring Framework part 2
3. Конфигурирование Spring-приложений
4. Unit-тестирование
5. «Чёрная магия» Spring Boot
6. AOP, Spring AOP
7. Продвинутая конфигурация Spring-приложений
8. Разбор домашних заданий, QnA

Ваши вопросы?

Организационные вопросы



Spring Framework



Spring IoC



Практика

01

Spring Framework

- **Spring** – индустриальный стандарт.
- Не побоюсь сказать, что 80% вакансий – так или иначе используют Spring
- Spring – «**lightweight**» фреймворк (объём кода, который необходимо написать программисту – минимален).
- Мы, как раз, и будем учиться эффективно его использовать

- Spring – это фреймворк уровня приложений, на нём можно построить целиком всё приложение.
- Есть «подфреймворки» Spring – **проекты Spring**, которые отвечают только за один слой приложения.
- Наш курс и построен на изучении проектов Spring

- Spring IoC + AOP = Context
- Spring JDBC
- Spring Data *
- Spring MVC, Spring WebFlux
- Spring Security
- Spring Integration
- Spring Cloud *
- Spring Boot
- И куча всего другого интересного 😊



Spring Framework 5



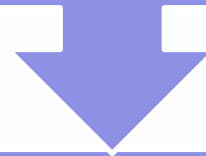
<http://spring.io>

- Документация
- Текущие версии библиотек
- Примеры и tutorials

- <http://www.baeldung.com> - лаконичные примеры (не всегда эффективные);
- <http://www.mkyong.com/> – максимально лаконичные примеры (иногда старый Spring);
- <https://stackoverflow.com/questions/tagged/spring> – и другие разделы посвящённые Spring;
- <http://mvnrepository.com/> – при работе с Maven зависимостями.

Ваши вопросы?

Организационные вопросы



Spring Framework



Spring IoC

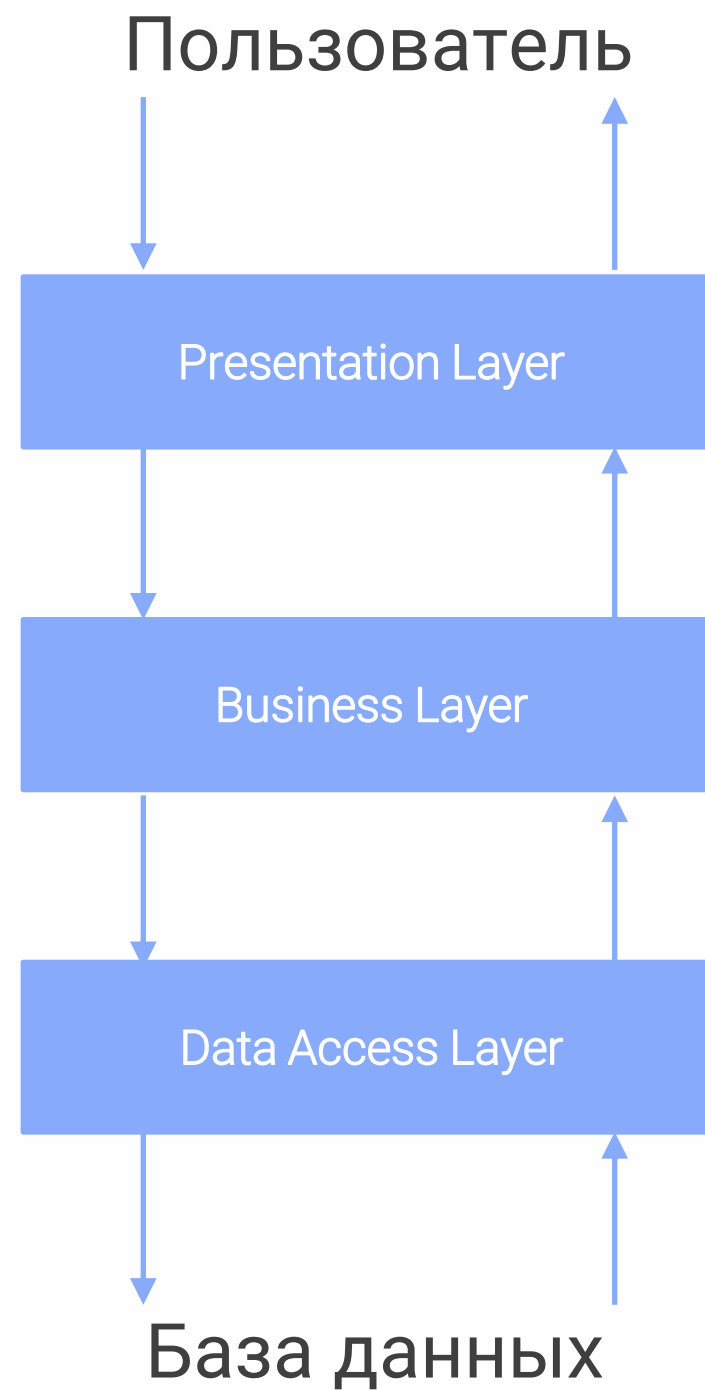


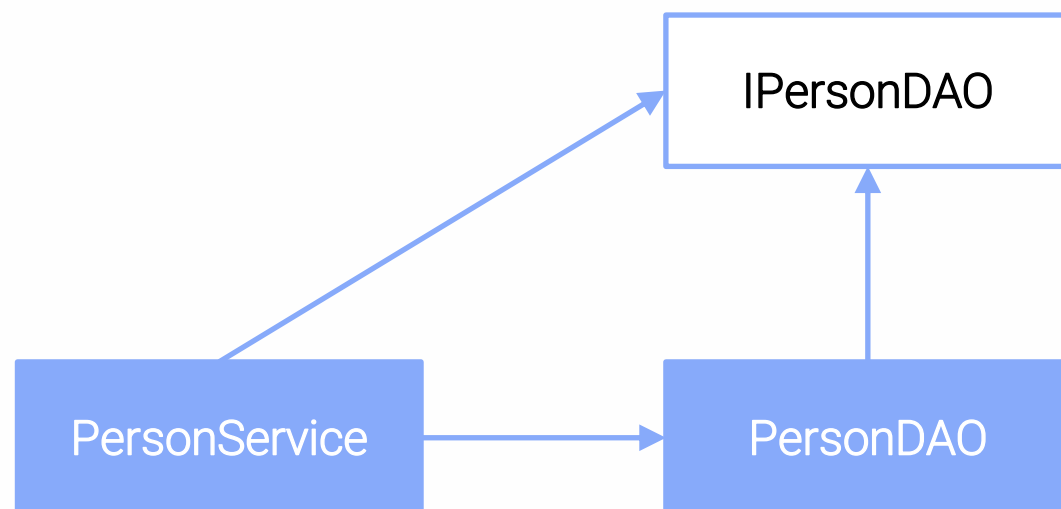
Практика



02

Spring IoC





```
class PersonService {

    private final IPersonDAO dao;

    public PersonService() {
        this.dao = new PersonDAO(
            "127.0.0.1:80"
        );
    }

    public Person getByName(...) { ... }
}

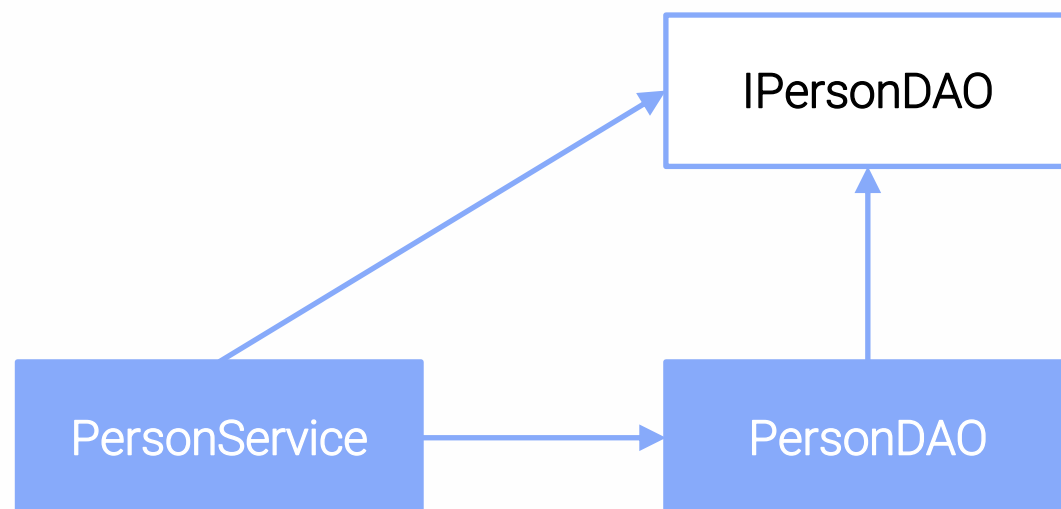
class PersonDAO implements IPersonDAO {

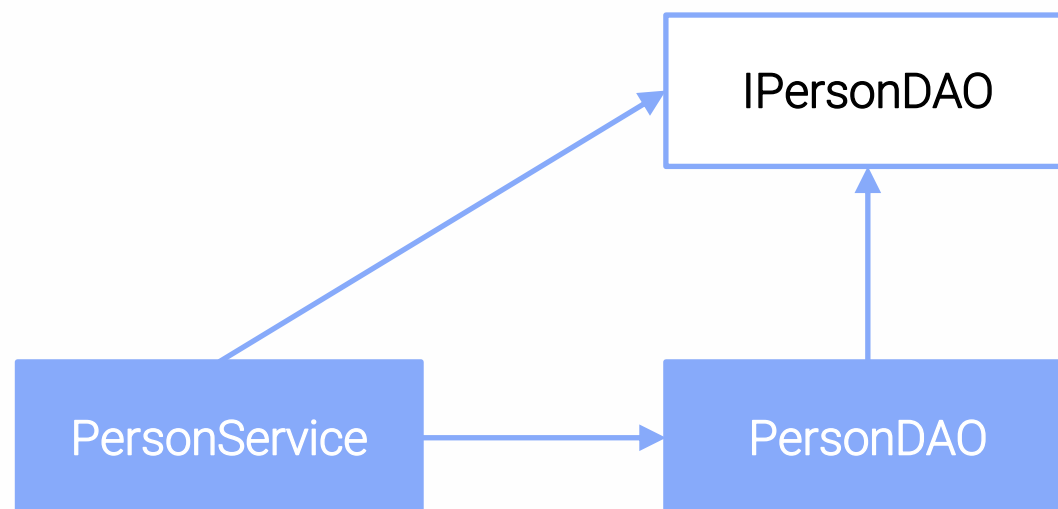
    private final String url;

    public PersonDAO(String url) {
        this.url = url;
    }

    public Person findByName(...) { ... }
}
```

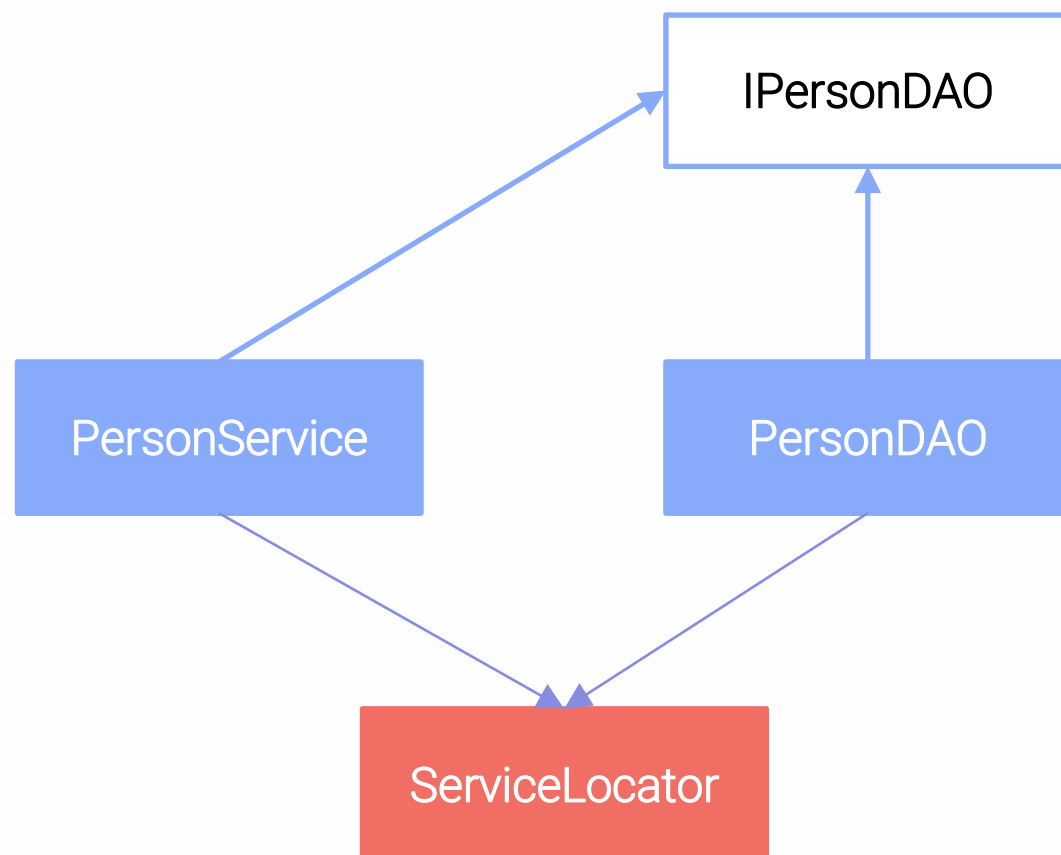
- Чем хорош?
- Чем плох?





- Класс **PersonService** напрямую зависит от **PersonDAO**.
- Невозможно тестировать **PersonService** отдельно от **PersonDAO**.
- Жизненный цикл классов связан напрямую.
- Невозможно заменить **PersonDAO** на другую реализацию.

Service Locator



```
class PersonService {

    private final IPersonDAO dao;

    public PersonService() {
        this.dao = ServiceLocator
            .getPersonDAO();
    }

    public Person getByName(...) { ... }
}

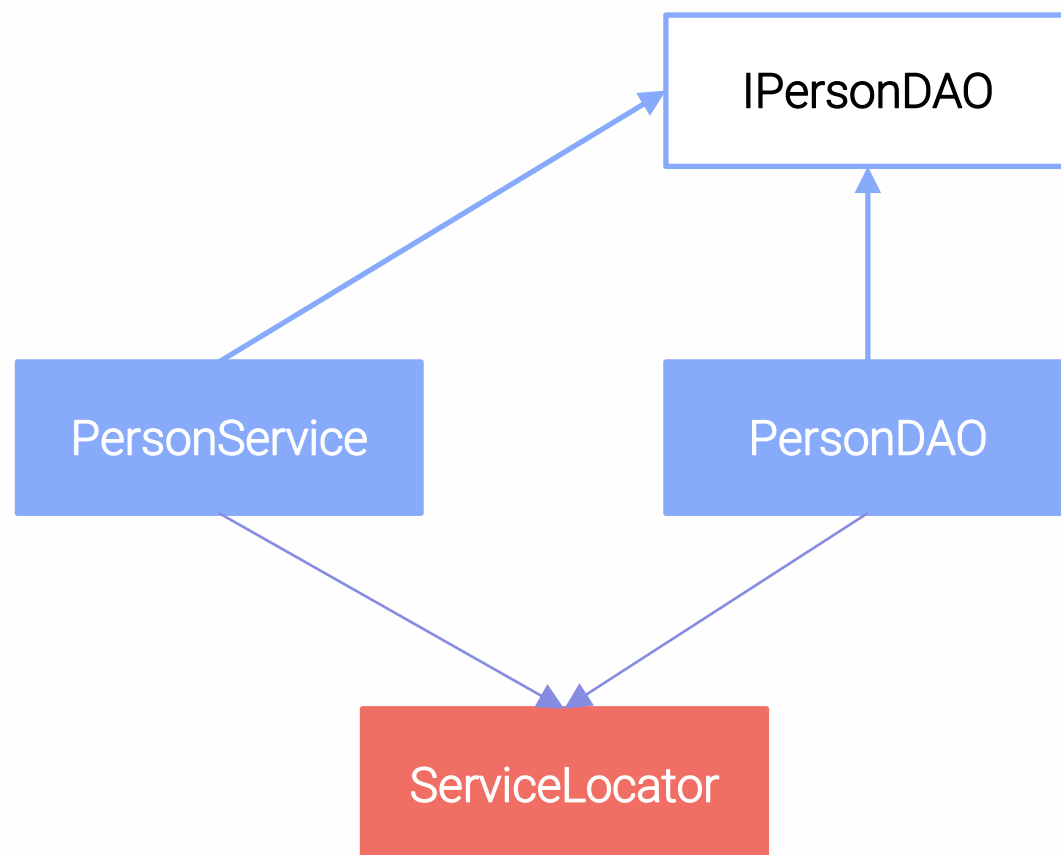
class PersonDAO implements IPersonDAO {

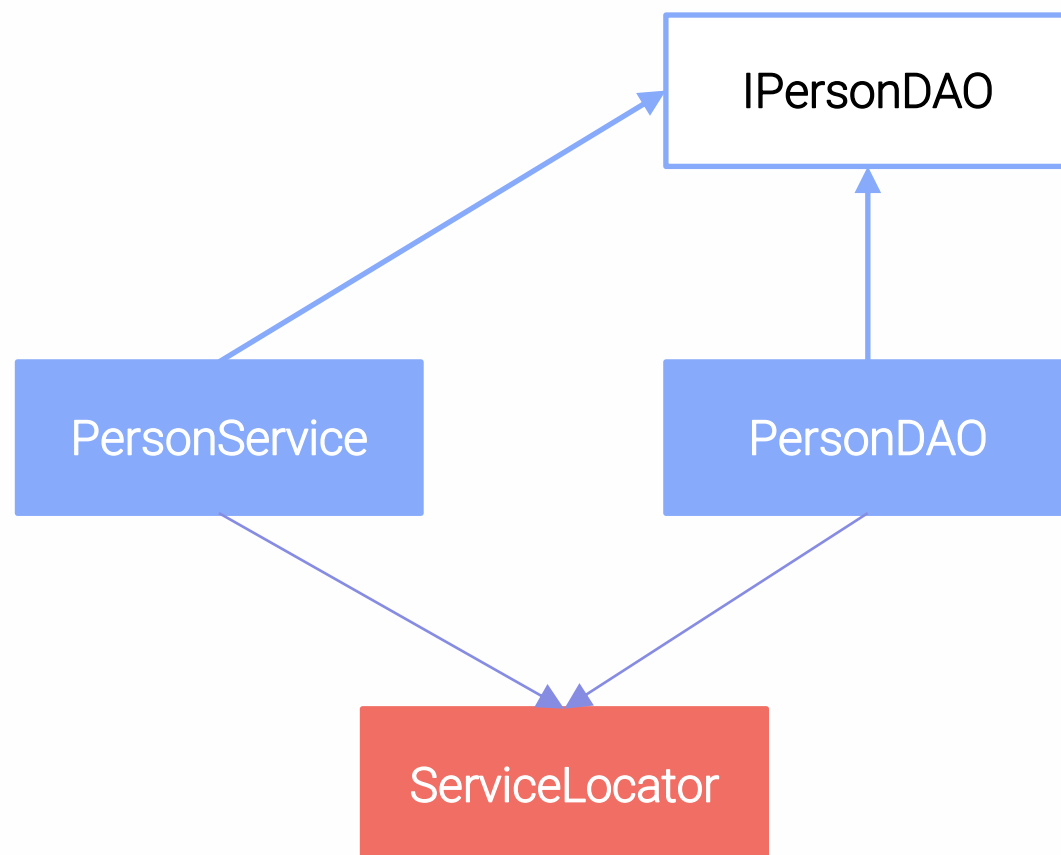
    private final String url;

    public PersonDAO(String url) {
        this.url = url;
    }

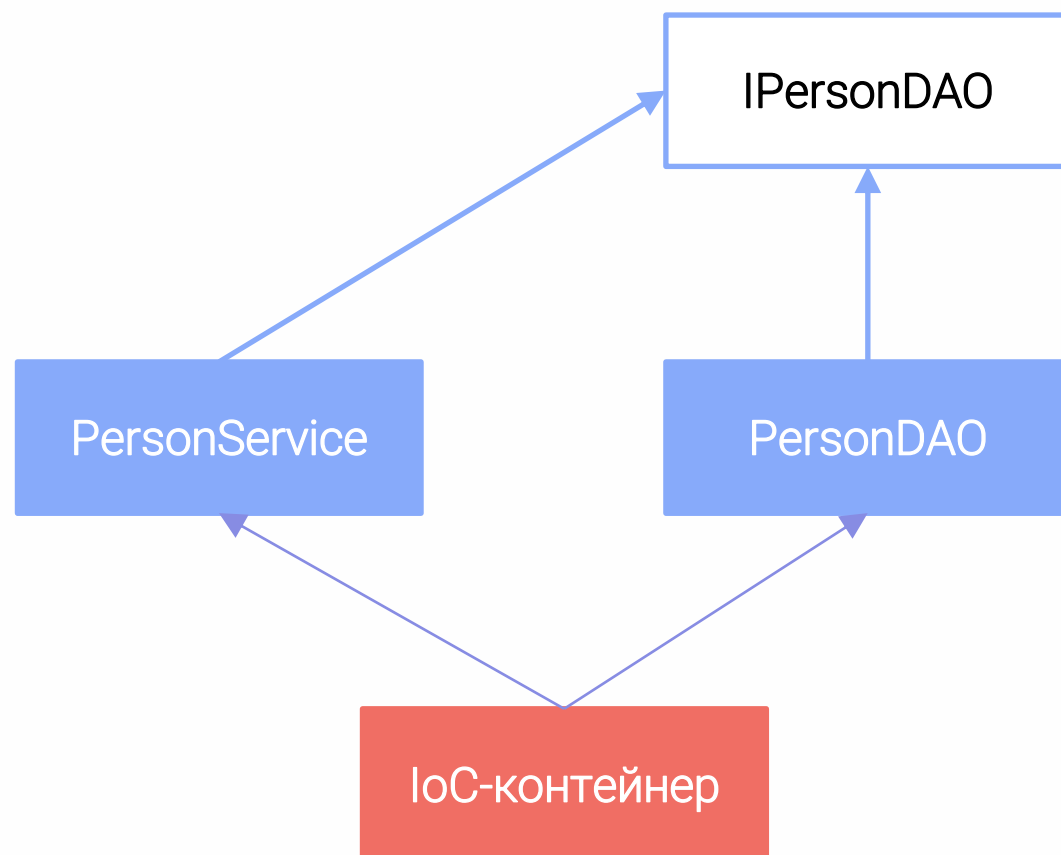
    public Person findByName(...) { ... }
}
```

- Чем хорош?
- Чем плох?





- Класс PersonService не зависит от PersonDAO, но зависит от ServiceLocator-а.
- Если постараться, то PersonService можно тестировать отдельно от PersonDAO.
- PersonDAO можно заменить на другую реализацию.



```
class PersonService {

    private final IPersonDAO dao;

    public PersonService(IPersonDao d) {
        this.dao = d;
    }

    public Person getByName(...) { ... }
}

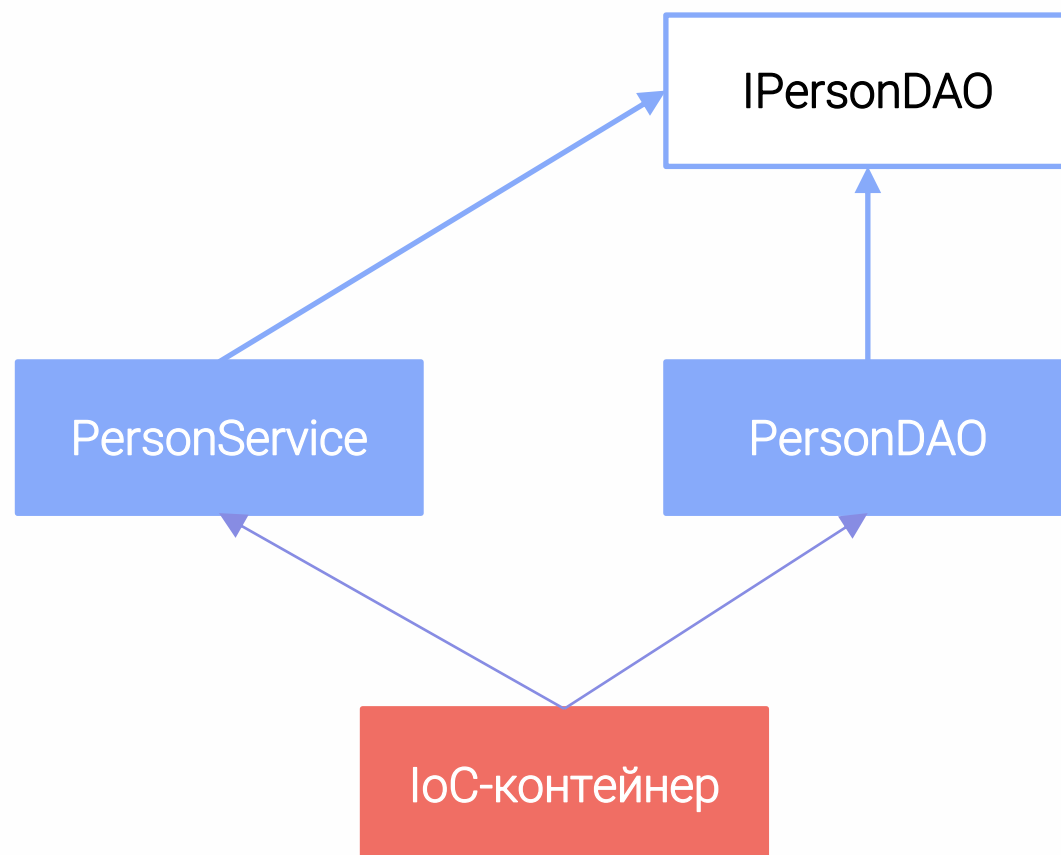
class PersonDAO implements IPersonDAO {

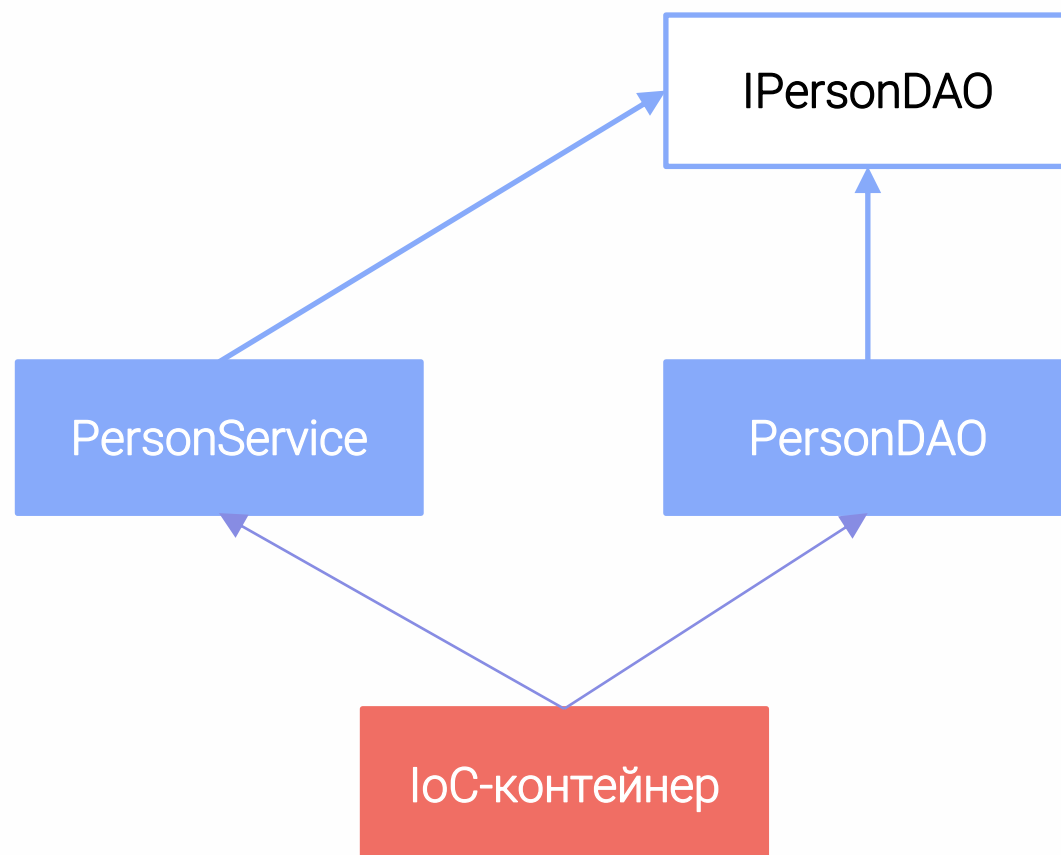
    private final String url;

    public PersonDAO(String url) {
        this.url = url;
    }

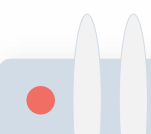
    public Person findByName(...) { ... }
}
```

- Чем хорош?
- Чем плох?





- Контейнер создаёт необходимые объекты и управляет жизненным циклом
- Класс **PersonService** не зависит от **PersonDAO**
- **PersonService** можно легко тестировать отдельно от **PersonDAO**
- **PersonDAO** можно заменить на другую реализацию

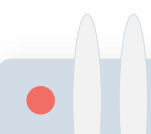


```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="personDao" class="ru.otus.spring.dao.PersonDao">
        <constructor-arg name="dbUrl" value="${db.url}"/>
    </bean>

    <bean id="personService" class="ru.otus.spring.service.PersonService">
        <constructor-arg name="dao" ref="personDao"/>
    </bean>

</beans>
```

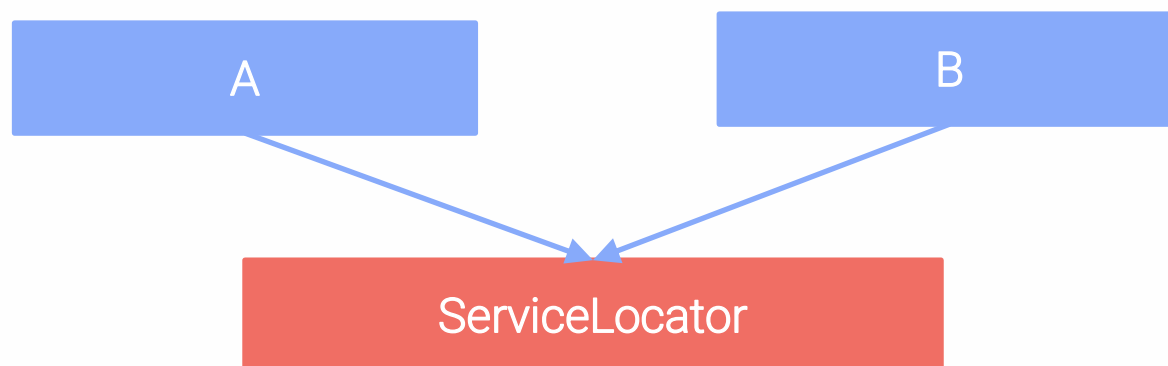
```
@Configuration
class AppConfig {

    @Bean
    IPersonDAO personDAO(@Value("${db.url}") String dbUrl) {
        return new PersonDAO(dbUrl);
    }

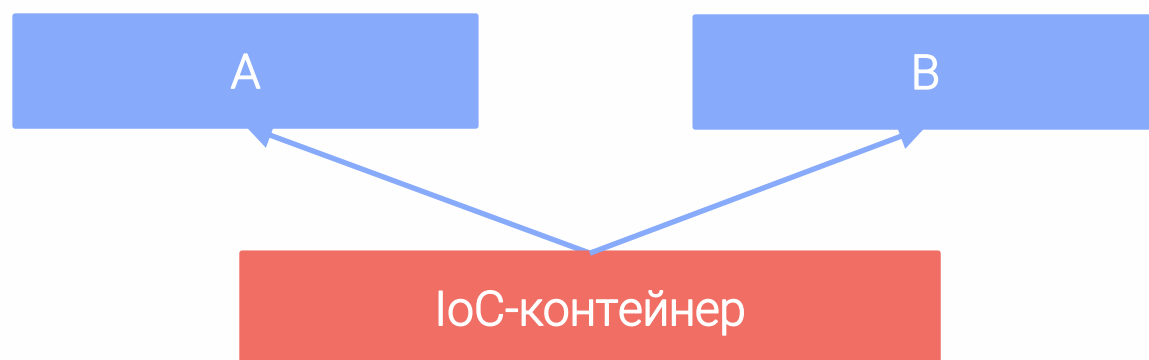
    @Bean
    PersonService personService(IPersonDAO dao) {
        return new PersonService(dao);
    }
}
```



Классический подход



Service Locator



IoC-контейнер

- **IoC** (Inversion of Control) – базовый принцип, на котором строится Spring.
- «**Голливудский принцип**» (Hollywood Principle):
«Не звоните мне, я сам Вам позвоню.»
- **DI** – Dependency Injection – это IoC в применении к зависимостям (не путать с DIP)

```
class PersonService {  
  
    private final IPersonDAO dao;  
  
    public PersonService() {  
        this.dao = ServiceLocator  
            .getPersonDAO();  
    }  
  
    public Person getByName(...) { ... }  
}  
  
class PersonDAO implements IPersonDAO {  
  
    private final String url;  
  
    public PersonDAO(String url) {  
        this.url = url;  
    }  
  
    public Person findByName(...) { ... }  
}
```

```
class PersonService {  
  
    private final IPersonDAO dao;  
  
    public PersonService(IPersonDao d) {  
        this.dao = d;  
    }  
  
    public Person getByName(...) { ... }  
}  
  
class PersonDAO implements IPersonDAO {  
  
    private final String url;  
  
    public PersonDAO(String url) {  
        this.url = url;  
    }  
  
    public Person findByName(...) { ... }  
}
```

Ссылка: DI vs. DIP vs. IoC

<http://sergeyteplovakov.blogspot.com/2014/11/di-vs-dip-vs-ioc.html>

Что там есть:

- Чем отличаются все эти понятия 😊



- Упрощает reuse компонентов.
- Упрощает unit-тестирование – это у нас важно на курсе 😊
- Чистый код – только бизнес-логика, никакого конфигурационного кода.

Ваши вопросы?

- Бывают разные.
- Конфигурируются разными способами.
- Конфигурируются разными языками XML, Java, Groovy.

IoC-контейнеры в Spring

- **XML** является традиционным способом задания конфигурации контейнера, хотя существуют и другие способы задания метаданных (аннотации, Java код и т.д.).
- Во многих случаях проще и быстрее конфигурировать контейнер с помощью аннотаций. (это мы пройдем позже).
- **Первое занятие и первое ДЗ** мы будем конфигурировать с помощью XML, дальше забудем как страшный сон.

Сначала читаются
метаданные



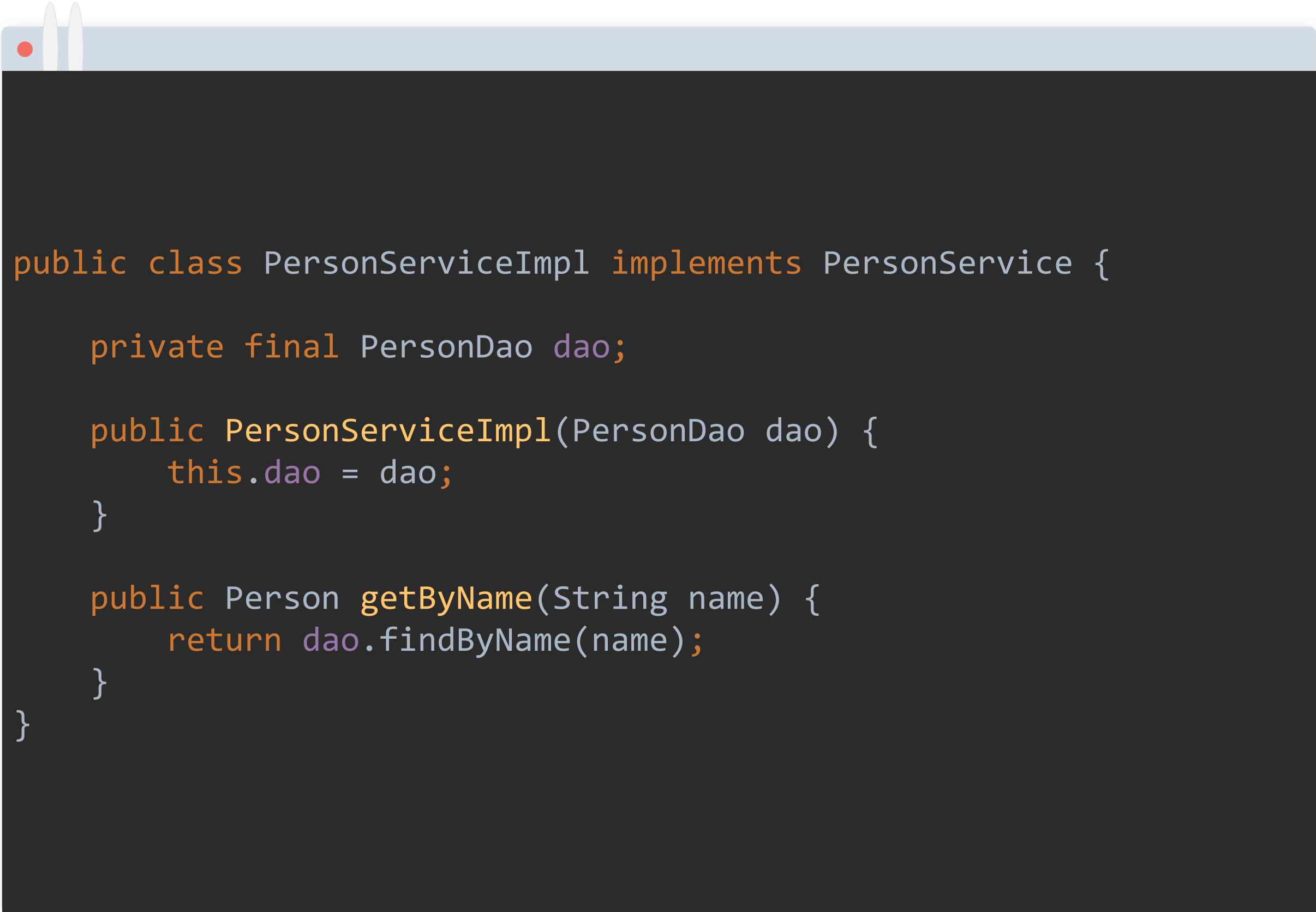
По метаданным читаются
все необходимые классы



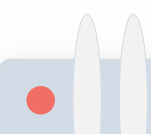
Создаются все бины




Задаются зависимости

A stylized code editor window with a light blue header bar and a dark gray body. On the left side of the header bar, there are two white vertical bars and a small red circle. The code is written in a monospaced font with syntax highlighting: keywords are orange, class names and variables are light blue, and literals are purple.

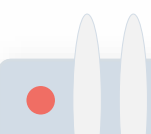
```
public class PersonServiceImpl implements PersonService {  
  
    private final PersonDao dao;  
  
    public PersonServiceImpl(PersonDao dao) {  
        this.dao = dao;  
    }  
  
    public Person getByName(String name) {  
        return dao.findByName(name);  
    }  
}
```



```
<dependency>  
  <groupId>org.springframework</groupId>  
  <artifactId>spring-context</artifactId>  
  <version>5.3.4</version>  
</dependency>
```



```
public class Main {  
  
    public static void main(String[] args) {  
        ClassPathXmlApplicationContext context =  
            new ClassPathXmlApplicationContext("/spring-context.xml");  
        PersonService service = context.getBean(PersonService.class);  
        Person ivan = service.getByName("Ivan");  
    }  
}
```



```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd">

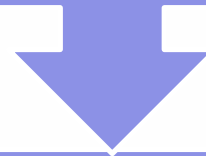
    <bean id="personDao" class="ru.otus.spring.dao.PersonDaoSimple">
    </bean>

    <bean id="personService" class="ru.otus.spring.service.PersonServiceImpl">
        <constructor-arg name="dao" ref="personDao"/>
    </bean>

</beans>
```

Ваши вопросы?

Организационные вопросы



Spring Framework



Spring IoC



Практика



03

Практика

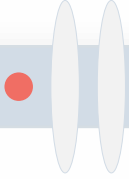
Ссылка: GitHub - репозиторий

<https://github.com/OtusTeam/Spring>

Здесь будут располагаться все упражнения,
которые мы будем делать на занятии



- Посмотреть реализацию Spring IOC на XML
- Выкачиваем репозиторий и просто смотрим

- 
1. `git clone git@github.com:OtusTeam/Spring.git`
или
`git clone https://github.com/OtusTeam/Spring.git`
 2. `cd ./<имя группы>/spring-01`
 3. Смотрим
`spring-01-solution/src/main/resources/spring-context.xml`
 4. Число 1 в чат, как посмотрите

Слушатели, которые смотрят нас в записи, могут поставить паузу.

Конец упражнения!
Ваши вопросы?

- Добавить Maven-зависимость spring-context в pom.xml

1. Посмотреть на <http://mvnrepository.com> текущую версию org.springframework:spring-context

2. Добавляем в **spring-01-exercise/pom.xml**

```
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-context</artifactId>  
    <version>????????????</version>  
</dependency>
```

3. Число 2 в чат, как сделаете

Слушатели, которые смотрят нас в записи, могут поставить паузу.

Конец упражнения!
Ваши вопросы?

- Создать контекст в main-методе
- Получить PersonService из контекста (его ещё там нет)
- Получить Person из PersonService

1. Добавляем в **spring-01-exercise/src/main/java/ru/otus/spring/Main.java**

```
ClassPathXmlApplicationContext context =  
    new ClassPathXmlApplicationContext("/spring-context.xml");  
PersonService service = context.getBean(PersonService.class);  
Person ivan = service.getByName("Ivan");
```

2. Число 3 в чат, как сделаете

Слушатели, которые смотрят нас в записи, могут поставить паузу.

Конец упражнения!
Ваши вопросы?

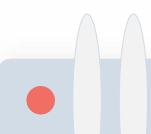
- Добавить бины в контекст.
- Настроить зависимость через <constructor-arg ref=.

1. Добавляем в
spring-01-exercise/src/main/resources/spring-context.xml

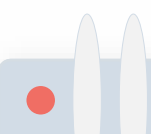
```
<bean id="personDao"  
      class="ru.otus.spring.dao.PersonDaoSimple">  
</bean>  
<bean id="personService"  
      class="ru.otus.spring.service.PersonServiceImpl">  
  <constructor-arg name="dao" ref="personDao"/>  
</bean>
```

2. Число 4 в чат, как сделаете.

Конец упражнения!
Ваши вопросы?



```
public class PersonServiceImpl implements PersonService {  
  
    private PersonDao dao;  
  
    public PersonServiceImpl() {  
    }  
  
    public void setDao(PersonDao dao) {  
        this.dao = dao;  
    }  
  
    public Person getByName(String name) {  
        return dao.findByName(name);  
    }  
}
```



```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="personDao" class="ru.otus.spring.dao.PersonDaoSimple">
    </bean>

    <bean id="personService" class="ru.otus.spring.service.PersonServiceImpl">
        <property name="dao" ref="personDao"/>
    </bean>

</beans>
```

- Поправить классы зависимостей
- Настроить зависимость через `<property ref=`



1. Правим **PersonServiceImpl.java**

2. Правим **spring-context.xml**

3. Число 5 в чат, как сделаете.

Слушатели, которые смотрят нас в записи, могут поставить паузу.

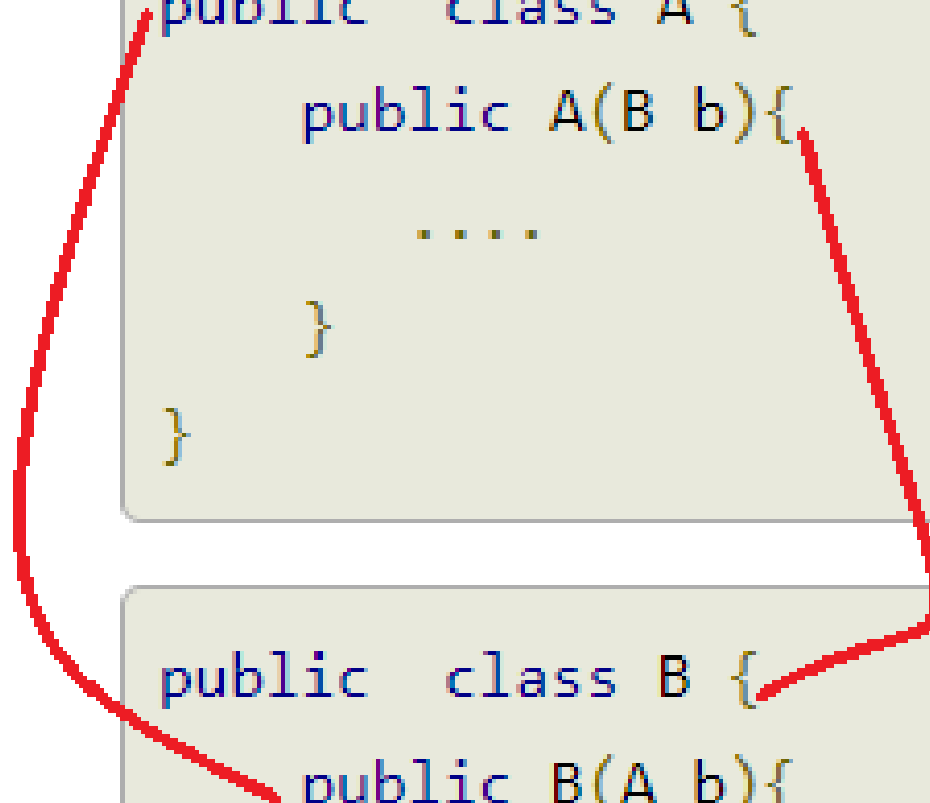
Конец упражнения!
Ваши вопросы?

- DI через конструктор – рекомендован.
- Но могут быть циклические зависимости (виноваты в этом – Вы!).
- Вы всегда получаете готовый к работе класс.
- С property может быть NPE.

Cyclic dependency

```
public class A {  
    public A(B b){  
        ....  
    }  
}
```

```
public class B {  
    public B(A b){  
        ....  
    }  
}
```



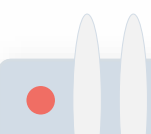
Упражнение 6*

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="personDao" class="ru.otus.spring01.dao.PersonDaoSimple">
        <property name="defaultAge" value="28"/>
    </bean>

    <bean id="personService" class="ru.otus.spring01.service.PersonServiceImpl">
        <property name="dao" ref="personDao"/>
    </bean>

</beans>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="personDao" class="ru.otus.spring.dao.PersonDaoSimple">
        <!-- 18 в PersonDaoSimple заменить на поле, задаваемое здесь -->
        <property name="defaultAge" value="24"/>
    </bean>

    <bean id="personService" class="ru.otus.spring.service.PersonServiceImpl">
        <property name="dao" ref="personDao"/>
    </bean>

</beans>
```

Конец упражнения!
Ваши вопросы?

- Бизнес-сервисы (DAO, Services);
- Подключения к внешним системам;
- Мапперы/Маршаллеры/конвертеры;
- Служебные бины (PersistenceContextManager...);
- Бизнес-бины стратегий (Паттерн стратегия).

- Бизнес-объекты (бины, пользователи*);
- Настройки, кроме пачек/файлов настроек*;
- Объекты, которые понадобятся только один раз в один момент (временные).
- Стандартные классы (String, InputStream, Locale*)
- Scanner в Вашем домашнем задании, которое сейчас задам 😊

Ваши вопросы?

Программа по проведению тестирования студентов

- В ресурсах хранятся вопросы и различные ответы к ним в виде CSV файла.
- Программа должна спросить у пользователя фамилию и имя, спросить вопросы из CSV-файла и вывести результат тестирования.

В рамках этого ДЗ (первая часть):

- Только чтение и вывод вопросов (!)

Описание части на это ДЗ:

- В ресурсах хранятся вопросы и различные ответы к ним в виде CSV файла (5 вопросов).
- Вопросы могут быть с выбором из нескольких вариантов или со свободным ответом - на Ваше желание и усмотрение.
- Приложение должна просто вывести вопросы теста из CSV-файла с возможными вариантами ответа.

Требования:

0. В приложении должна присутствовать объектная модель (отдаём предпочтение объектам и классам, а не строчкам и массивам/спискам строчек).
1. Все классы в приложении должны решать строго определённую задачу (см. п. 18-19 "Правила оформления кода.pdf", прикрепленные к материалам занятия).
2. Контекст описывается XML-файлом.
3. Все зависимости должны быть настроены в iOS контейнере.
4. Имя ресурса с вопросами (CSV-файла) необходимо захардкодить строчкой в XML-файле с контекстом.

5. CSV с вопросами читается именно как ресурс, а не как файл.
6. Scanner, PrintStream и другие стандартные типы в контекст класть не нужно!
7. Весь ввод-вывод осуществляется на английском языке.
8. Крайне желательно написать юнит-тест какого-нибудь сервиса (оцениваться будет только попытка написать тест).
9. Помним – «без фанатизма» ☺

Опционально (задание со "звёздочкой"):

- 1*. Приложение должно корректно запускаться с помощью
"java -jar"

Файл: Правила оформления кода

Что там есть:

- То, к чему мы будем особо придираться
- Ссылки на соглашения 😊



- Дедлайна нет! Ну кроме окончания курса 😊
- Но если опоздаете – будет проверена позже, после остальных
- Срок нашей проверки работы – 2 дня, но мы, обычно, проверяем быстрее
- Код, написанный в данном ДЗ будет использоваться дальше.
- Можно и делать задания раньше, чем мы их дадим, но будьте готовы переделать)

Факт сдачи:

0 – 1 - задание не сдано / сдано

Степень выполнения (value, количество работающего функционала, что примет заказчик, что будет проверять тестировщик):

0 – 4 - ничего не работает или отсутствует основной функционал /

основной функционал есть, всё хорошо работает, тесты и/или задание перевыполнено

Способ выполнения (code quality, качество выполнения, стиль кода, как ревью перед мержем):

0 – 5 - нужно править, мержить нельзя (нарушение соглашений) /

отличная работа + экстра балл*

Статус "Принято" ставится от 6 и выше баллов.

Идеальное, но не работающее, решение = 5 - не принимается.

Если всё работает, но стилю не соответствует (публичные поля, классы капсом) = 5 - тоже не принимается

Видео: ЛикБез – Введение в Maven

Что там есть:

- Введение в Maven)



Ваши вопросы?

- Разобрались с вопросами, где писать, куда сдавать ДЗ 😊
- Что такое Spring
- Что такое DI
- Как создавать контекст со Spring
- Ну и как конфигурировать бины в XML
- И получили первое домашнее задание 😊

Ваши вопросы?

**Пожалуйста, пройдите
опрос**

<https://otus.ru/polls/23473/>

За опросы - плюшки

Спасибо за внимание!
ЮС в помощь!

