IT-Academy

Spring intro

Автор: Юлий Слабко

Simplifying Java development

IT-Academy

- Возможность управления общими зависимостями в единственном репозитории
- Простая разработка с POJOs (Plain Old Java Objects)
- Слабая связь через dependency injection и ориентация на интерфейсное взаимодействие
- Декларативная разработка через применение аспектов и общих соглашений
- Сокращение объема программного кода через аспекты и шаблоны
- Упрощенная конфигурация приложения

Injecting dependencies

IT-Academy

```
public class SenderService {

    private EmailSender sender;

    public SenderService() { }

    public SenderService(EmailSender sender) {
        this.sender = sender;
    }

    public void sendMessage(Message message) {
        sender.send(message);
    }


    public void setSender(EmailSender sender) {
        this.sender = sender;
    }
}
```

```
public class EmailSender {

    void send(Message message) {
        try {
            Transport.send(message);
        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }
}
```

Injecting dependencies

IT-Academy

```
public class SenderService {

    private ISender sender;

    public SenderService() { }

    public SenderService(ISender sender) {
        this.sender = sender;
    }

    public void sendMessage(Message message) {
        sender.send(message);
    }


    public void setSender(ISender sender) {
        this.sender = sender;
    }
}
```

```
interface ISender {
    void send(Message message);
}
```

```
public class EmailSender implements ISender {

    public void send(Message message) {
        try {
            Transport.send(message);
        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }
}
```

Injecting dependencies

- Момент заключается в том, что Person не связан с конкретной реализацией Sender. Ему не важно какой вид адреса передается в конструктор, т.к. передаются классы-потомки Sender. Таким образом, главный *бенефит DI* – слабая связь. Если объект знает о связи по интерфейсу, таким образом зависимость может быть вынесена с различными реализациями, без информации о конкретной реализации.

Injecting dependencies

IT-Academy

```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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    <bean id="emailSender" class="by.it.academy.services.sender.EmailSender"/>
    <bean id="senderService" class="by.it.academy.services.sender.SenderService">
        <property name="sender" ref="emailSender"/>
    </bean>
</beans>
```

Injecting dependencies

IT-Academy

```
public class MainLoader {
    public static void main(String[] args) {
        ApplicationContext context = new
       ClassPathXmlApplicationContext("senderContext.xml");
        SenderService service = context.getBean("senderService", SenderService.class);
        service.sendMessage(new POP3Message(null, 12));
    }
}
```

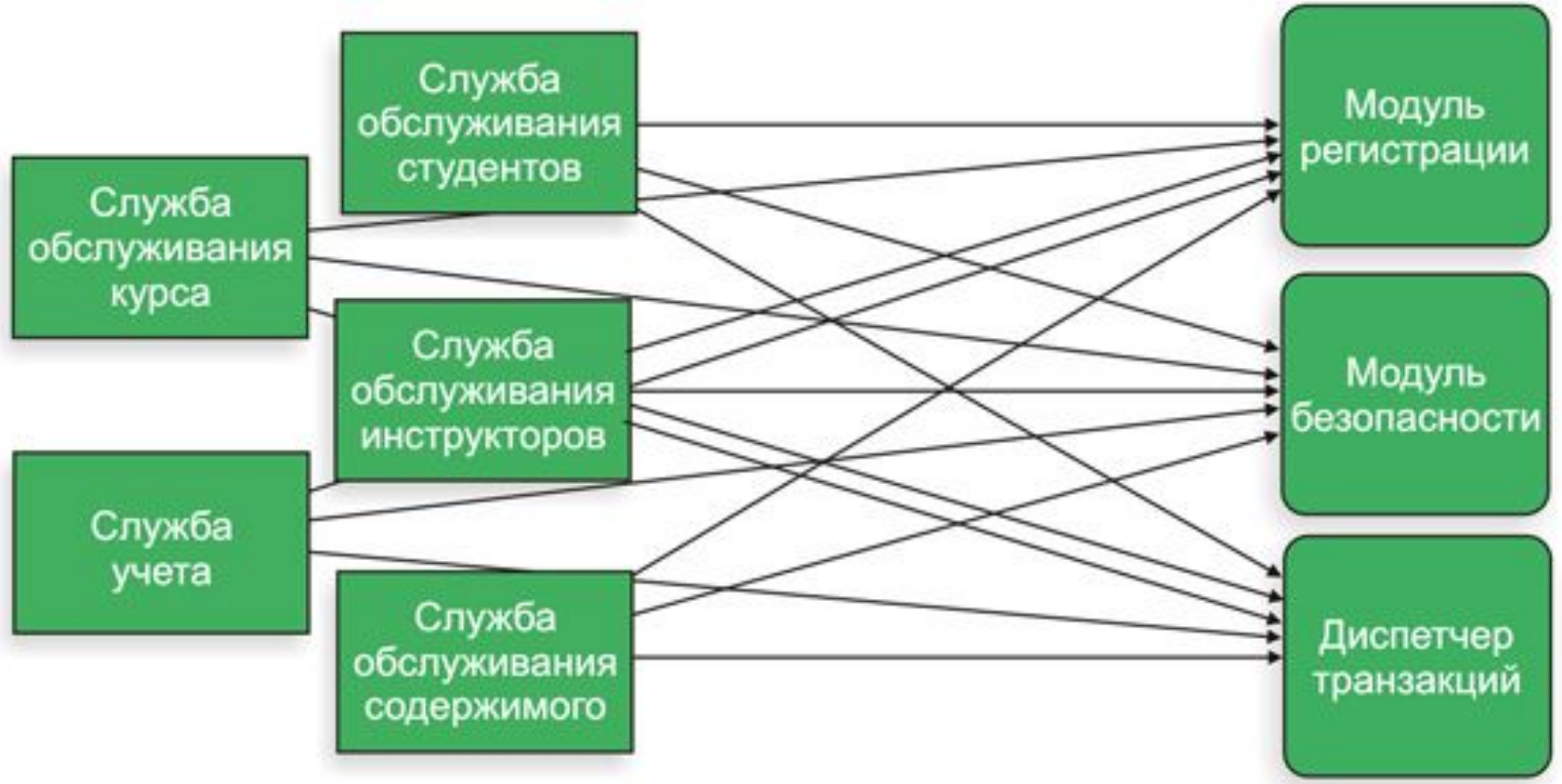
Spring dependencies

IT-Academy

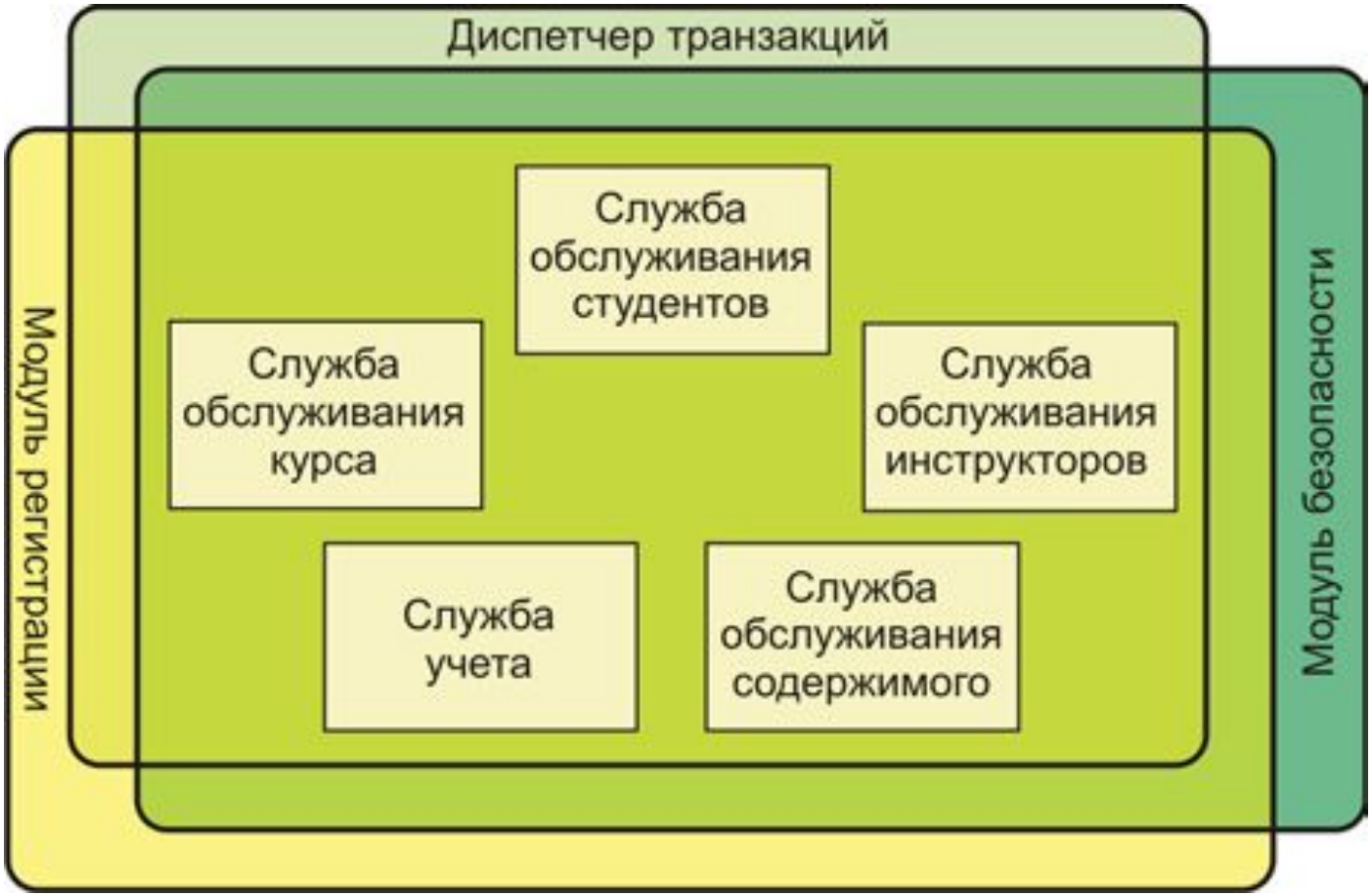
```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-aop</artifactId>
  <version>${springVersion}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-beans</artifactId>
  <version>${springVersion}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>${springVersion}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context-support</artifactId>
  <version>${springVersion}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>${springVersion}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-expression</artifactId>
  <version>${springVersion}</version>
</dependency>
```

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-orm</artifactId>
  <version>${springVersion}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-test</artifactId>
  <version>${springVersion}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-tx</artifactId>
  <version>${springVersion}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>${springVersion}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>${springVersion}</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-oxm</artifactId>
  <version>${springVersion}</version>
</dependency>
```

Applying aspects



Applying aspects



Applying aspects

```java
package by.academy.it.beans.first;

public interface IAddress {
    String getStreet();
}
```

```java
package by.academy.it.beans.first;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class Address implements IAddress {
    private Long id;
    private String flat;
    private String street;
    private String city;
    private String country;
}
```

```java
package by.academy.it.beans.first;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor @AllArgsConstructor
public class Person {
    private Integer id;
    private String name;
    private String surname;
    private String beanName;

    private IAddress address;

    public String getStreet() {
        return address.getStreet();
    }
}
```

Applying aspects

IT-Academy

```
public class Notifier {
    public void notifyBefore() {
        System.out.println("Notification before executing getStreet()");
    }

    public void notifyAfter() {
        System.out.println("Notification after executing getStreet()");
    }
}
```

Applying aspects. aopContext.xml

IT-Academy

```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:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">
    <bean id="address" class="by.academy.it.pojos.Address">
        <property name="street" value="Mira"/>
    </bean>
    <bean id="person" class="by.academy.it.pojos.Person">
        <property name="address" ref="address"/>
    </bean>
    <bean id="notifier" class="by.academy.it.pojos.Notifier"/>
    <aop:config>
        <aop:aspect id="note" ref="notifier">
            <aop:pointcut id="advice" expression="execution(* *.getStreet(..))" />
            <aop:before pointcut-ref="advice" method="notifyBefore"/>
            <aop:after pointcut-ref="advice" method="notifyAfter"/>
        </aop:aspect>
    </aop:config>
</beans>
```

Applying aspects. pom.xml

```
<properties>
    <spring.version>4.3.11.RELEASE</spring.version>
    <aspect.version>1.8.4</aspect.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
        <version>${aspect.version}</version>
    </dependency>
</dependencies>
```

Applying aspects

IT-Academy

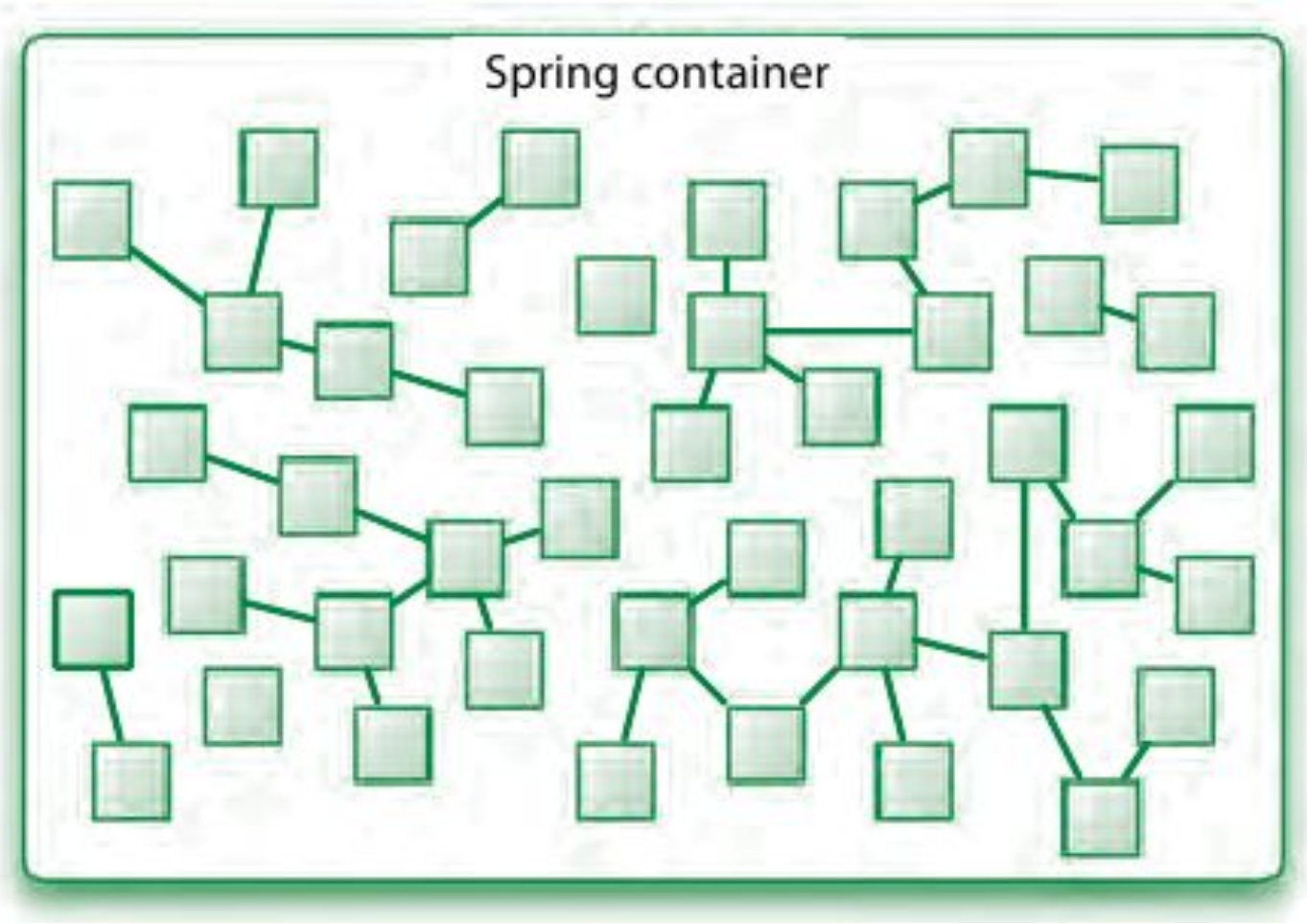
```
public class MainLoader {
    public static void main(String[] args) {
        ApplicationContext context = new
       ClassPathXmlApplicationContext("aopContext.xml");
         Person person = context.getBean("person", Person.class);
         person.getStreet();
    }
}
```

```
INFO: Loading XML bean definitions from class path resource [spring-config.xml]
Notification before executing getStreet()
Notification before executing getStreet()
Notification after executing getStreet()
Notification after executing getStreet()
Mira
```

IT-Academy

ВАШИ ВОПРОСЫ?

Containing your beans

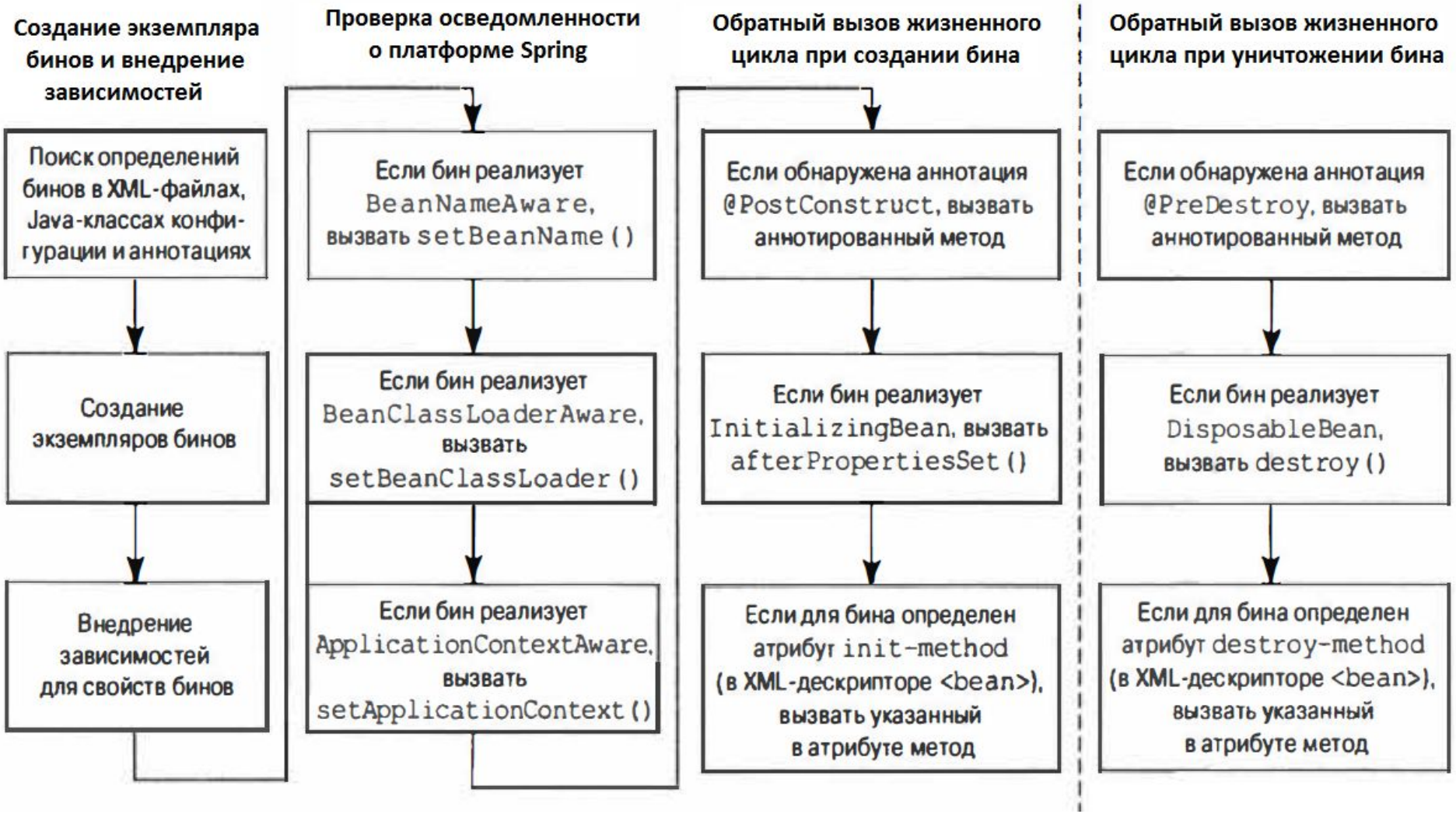


Conteiner interfaces

- **BeanFactory**

- **ApplicationContext**
  - **ClassPathXmlApplicationContext**
  - **FileSystemXmlApplicationContext**
  - **GenericApplicationContext**
  - **XmlWebApplicationContext**

A bean's life

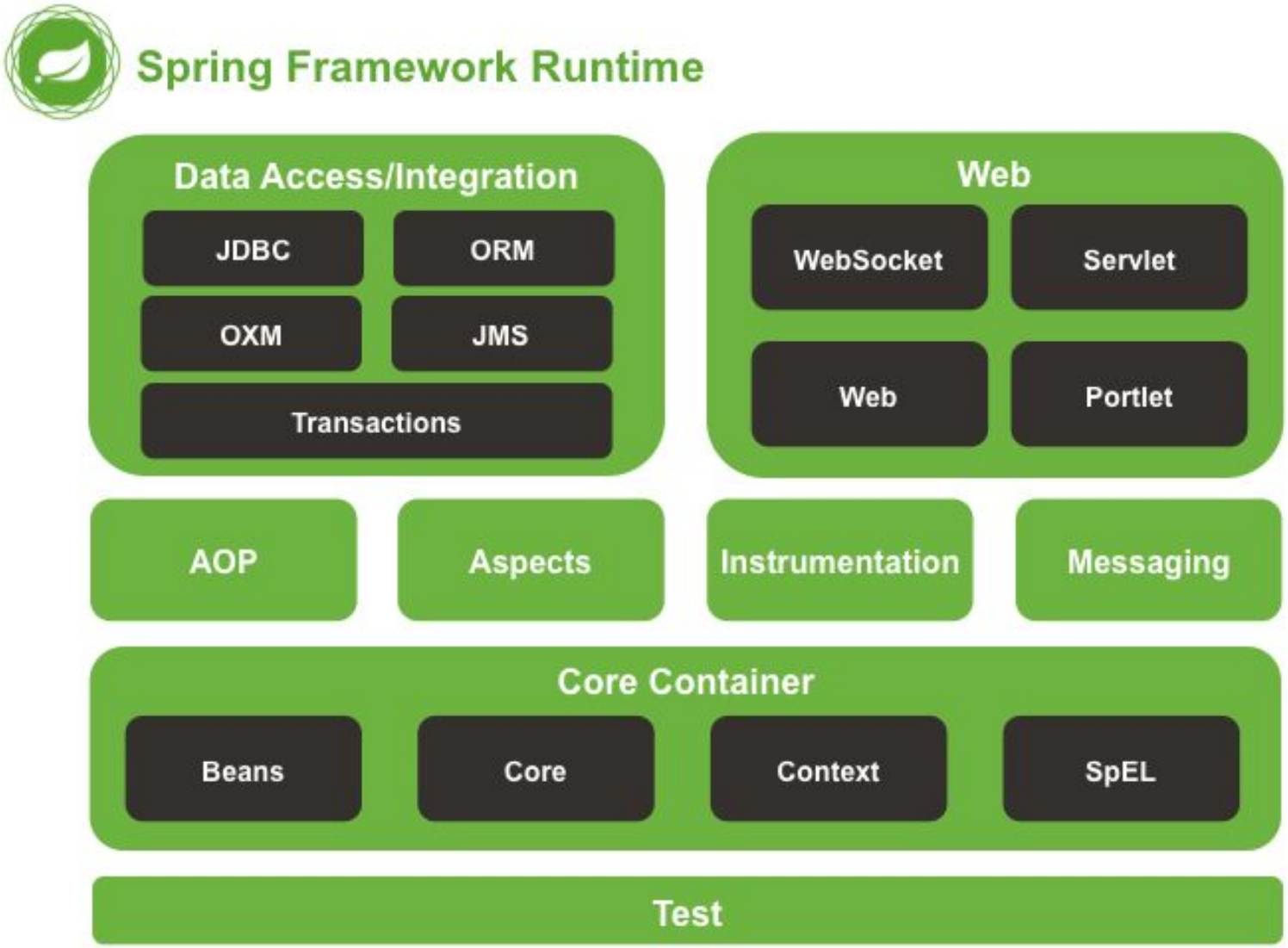

Создание экземпляра бинов и внедрение зависимостей
Поиск определений бинов в XML-файлах, Java-классах конфигурации и аннотациях
Создание экземпляров бинов
Внедрение зависимостей для свойств бинов
Проверка осведомленности о платформе Spring
Если бин реализует BeanNameAware, вызвать setBeanName()
Если бин реализует BeanClassLoaderAware, вызвать setBeanClassLoader()
Если бин реализует ApplicationContextAware, вызвать setApplicationContext()
Обратный вызов жизненного цикла при создании бина
Если обнаружена аннотация @PostConstruct, вызвать аннотированный метод
Если бин реализует InitializingBean, вызвать afterPropertiesSet()
Если для бина определен атрибут init-method (в XML-дескрипторе <bean>), вызвать указанный в атрибуте метод
Обратный вызов жизненного цикла при уничтожении бина
Если обнаружена аннотация @PreDestroy, вызвать аннотированный метод
Если бин реализует DisposableBean, вызвать destroy()
Если для бина определен атрибут destroy-method (в XML-дескрипторе <bean>), вызвать указанный в атрибуте метод

Spring modules

IT-Academy

GroupId	ArtifactId	Description
org.springframework	spring-aop	Proxy-based AOP support
org.springframework	spring-aspects	AspectJ based aspects
org.springframework	spring-beans	Beans support, including Groovy
org.springframework	spring-context	Application context runtime, including scheduling and remoting abstractions
org.springframework	spring-context-support	Support classes for integrating common third-party libraries into a Spring application context
org.springframework	spring-core	Core utilities, used by many other Spring modules
org.springframework	spring-expression	Spring Expression Language (SpEL)
org.springframework	spring-instrument	Instrumentation agent for JVM bootstrapping
org.springframework	spring-instrument-tomcat	Instrumentation agent for Tomcat
org.springframework	spring-jdbc	JDBC support package, including DataSource setup and JDBC access support
org.springframework	spring-jms	JMS support package, including helper classes to send and receive JMS messages
org.springframework	spring-messaging	Support for messaging architectures and protocols
org.springframework	spring-orm	Object/Relational Mapping, including JPA and Hibernate support
org.springframework	spring-oxm	Object/XML Mapping
org.springframework	spring-test	Support for unit testing and integration testing Spring components
org.springframework	spring-tx	Transaction infrastructure, including DAO support and JCA integration
org.springframework	spring-web	Web support packages, including client and web remoting
org.springframework	spring-webmvc	REST Web Services and model-view-controller implementation for web applications
org.springframework	spring-webmvc-portlet	MVC implementation to be used in a Portlet environment
org.springframework	spring-websocket	WebSocket and SockJS implementations, including STOMP support

Spring modules



Spring namespaces

IT-Academy

- **aop** – предоставляют элементы для декларирования аспектов, и для автоматического проксирования @AspectJ – аннотированные классы как Spring аспекты.
- **beans** – базовые примитивы Spring namespace, включая декларирование бинов и как они должны быть связаны.
- **context** – приходят с элементами для конфигурирования Spring контекст приложения, включая возможность для автоопределения и автосвязи бинов и введения объектов не прямо управлемые Springom.

Spring namespaces

IT-Academy

- **jee** – предлагает интеграцию с JAVA EE API таких как JNDI и EJB
- **jms** - предоставляет конфигурационные элементы для декларирования message-driven POJOs
- **lang** – включает декларирование бинов, которые реализованы на Groovy, JRuby или BeanShell скриптов.

Spring namespaces

IT-Academy

- **mvc** - включает Spring MVC возможности, такие как аннотационно-ориентированных контроллеров, view-контроллеров, и интерсептеров.
- **oxm** – поддержка конфигурации Spring object-to-XML маппинг возможности.
- **tx** – предоставляет декларативные транзакционные конфигурации.

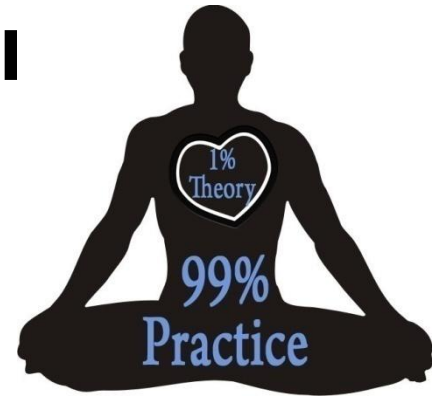
Вопросы

IT-Academy

Практика

IT-Academy

Создайте проект Spring.
Создайте конфигурационный файл и положите его в classpath.
Загрузите контекст Spring.
Получите из контекста бин и вызовите его методі



Литература

- http://www.tutorialspoint.com/spring/index.htm
- http://docs.spring.io/spring-framework/docs/4.2.3.RELEASE/spring-framework-reference/htmlsingle/
- https://docs.spring.io/spring/docs/current/spring-framework-reference/index.html

IT-Academy

Спасибо за внимание