

j2ee + Spring

Лекция 1.

Введение

Немного о себе



Программист (Java EE, iOS + Android, C++, ...)

ACS – разработка систем для федеральных заказчиков

Хобби: разрабатываю игры

E-mail: mrdekk@yandex.ru

Приходите к нам работать!

Что предстоит изучить

- Платформа j2ee

 - Что такое Enterprise и зачем он нужен

 - Некоторые архитектурные вопросы Enterprise приложений

 - Spring Framework как средство сохранить разум при Enterprise разработке

 - Основные технологии для повседневного использования

- Практические пример применения изучаемых технологий

 - Все о чем рассказал – все попробуем

 - Практики гораздо больше чем теории (можете задавать вопросы)

 - Сделаем интернет-витрину в качестве учебного проекта

Материалы к лекциям

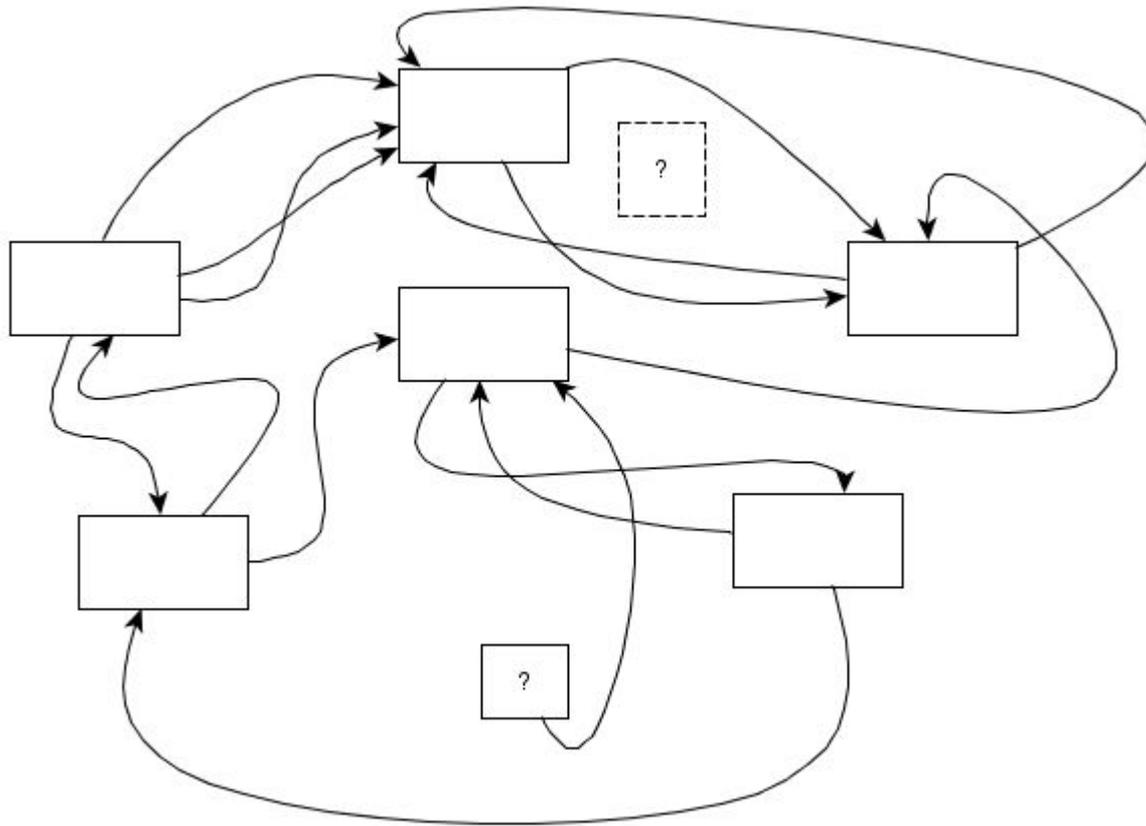
- Хорошая новость

Все материалы лекций, практик и мой код будут на github

Вот тут: https://github.com/mrdekk/j2ee_course

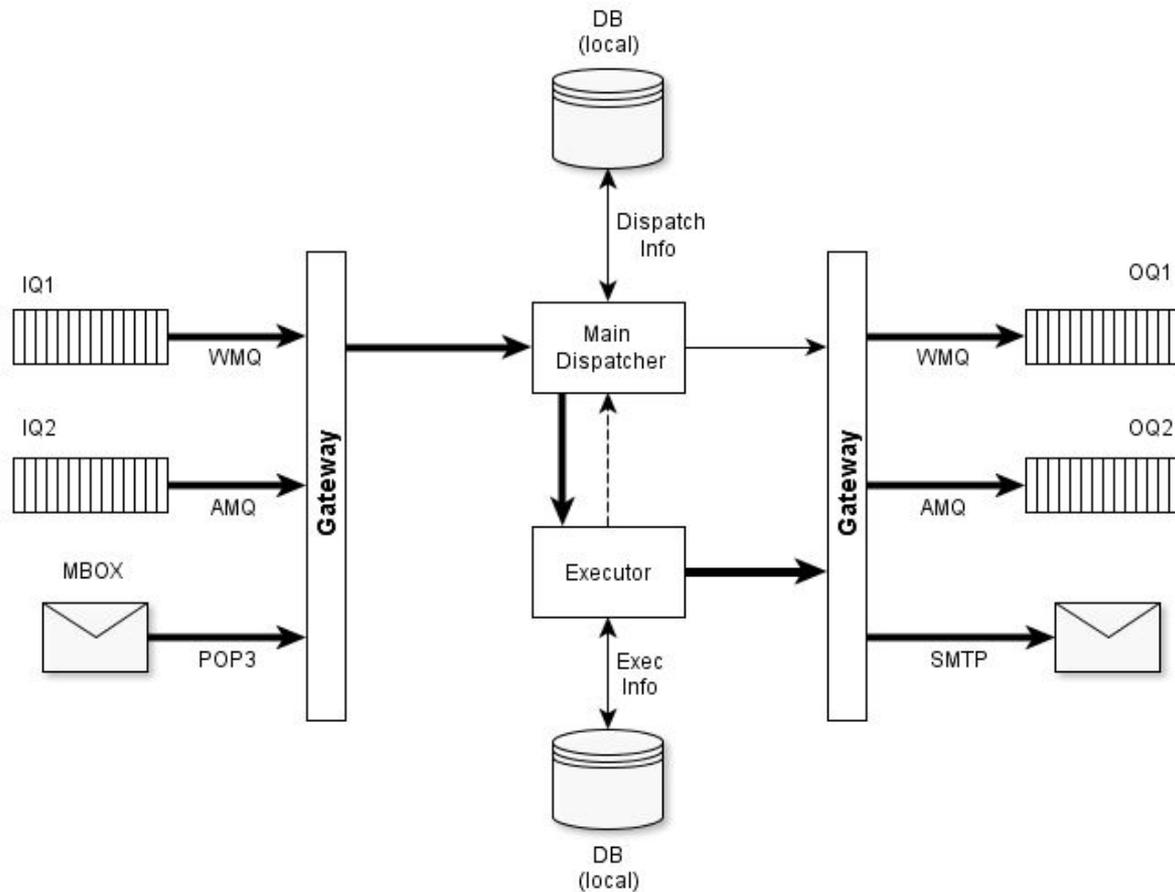
Архитектура приложений

- Плохая архитектура. Что вы думаете об этой картинке?



Архитектура приложений

- Хорошая архитектура. А что об этой картинке?



Важно!

При проектировании архитектуры приложения

- Разбить логику приложения на слабо-связанные модули, модули разбить на слои.
- Описать (или хотя бы разработать) схему связи этих модулей
- Модули реализовать в виде классов (набора классов)
- Связи реализовать через механизм DI/IoC

DI := Dependency Injection

IoC := Inversion of Control

Inversion of Control

Inversion of Control есть паттерн объектно-ориентированного программирования, который позволяет снизить связность объектов.

Разберем задачу на примере. Предположим, что у нас есть задача интеллектуального управления кондиционером в комнате. У нас есть датчик, устройство управления кондиционером и собственно модуль который нам надо реализовать.



Какие варианты модулей Вы бы предложили?

Inversion of Control

Как это делают обычно?

```
public class Main_1
{
    public static void main( String[ ] args )
    {
        Sensor sensor = new Sensor( "http://automation.local/sensor" );
        Actuator actuator = new Actuator( "http://automation.local/actuator" );

        IntellectualController controller = new IntellectualController( "Controller.1" );
        controller.setSensor( sensor );
        controller.setActuator( actuator );

        while ( true )
        {
            controller.doTheWork( );
        }
    }
}
```

Какие проблемы Вы здесь видите?

Inversion of Control

Как это делают обычно?

```
public class Main_1
{
    public static void main( String[ ] args )
    {
        Sensor sensor = new Sensor( "http://automation.local/sensor" );
        Actuator actuator = new Actuator( "http://automation.local/actuator" );

        IntellectualController controller = new IntellectualController( "Controller.1" );
        controller.setSensor( sensor );
        controller.setActuator( actuator );

        while ( true )
        {
            controller.doTheWork( );
        }
    }
}
```

Какие проблемы Вы здесь видите?

1. А что если мы захотим поменять урлы? ~~Конфиги~~
2. А что если мы захотим создать другой датчик? ~~Условия~~
3. А что если мы захотим создать другое исполнительное устройство?
~~Условия~~
4. А что если мы захотим делать работу не постоянно, а запланировать? ...
5. ...

Inversion of Control

Компоненты

1. Датчик (ISensor)
2. Исполнительное устройство (IActuator)

```
public interface ISensor
{
    public double getTemperature( );
}
```

```
public interface IActuator
{
    public void start( );
    public void stop( );
}
```

```
public class Sensor implements ISensor
{
    + public Sensor( String url ) {}
    + public double getTemperature( ) {}
}
```

```
public class Actuator implements IActuator
{
    + public Actuator( String url ) {}
    + public void start( ) {}
    + public void stop( ) {}
}
```

Тем самым мы абстрагируемся от деталей реализации конкретных модулей и получаем возможность менять реализацию, когда это необходимо.

Dependency Injection

Dependency Injection – один из подходов к реализации Inversion of Control.

```
IntellectualController controller = new IntellectualController( "Controller.1" );  
controller.setSensor( sensor );  
controller.setActuator( actuator );
```

Однако, делать это надо не вручную!

IoC-контейнер

Как правило для этого предназначен IoC контейнер.

Задачи:

1. Создавать объекты
2. Устанавливать зависимости

```
<?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.xsd" >

  <bean id="sensor" class="ru.mrdekk.j2ee.Sensor" />

  <bean id="actuator" class="ru.mrdekk.j2ee.Actuator" />

  <bean id="controller" class="ru.mrdekk.j2ee.IntellectualController" >
    <property name="sensor" ref="sensor" />
    <property name="actuator" ref="actuator" />
    <property name="limit" value="22.0" />
  </bean>

</beans>
```

Spring Framework



Spring Framework

Компоненты Spring Framework:

- **IoC-контейнер**
- **Фреймворк доступа к данным**
- ***Фреймворк управления транзакциями***
- **Фреймворк MVC**
- Фреймворк удалённого доступа
- Фреймворк аутентификации и авторизации
- Фреймворк удалённого управления
- **Фреймворк работы с сообщениями**
- ***Тестирование***

Maven

Но прежде о maven

The logo for Apache Maven, featuring the word "maven" in a bold, lowercase, sans-serif font. The letter "a" is highlighted in orange, while the remaining letters "m", "v", "e", "n" are in black.

Если кратко – декларативная система сборки

- Свойства
- Зависимости и Репозитории
- Профили
- Сборка и плагины

Maven

СВОЙСТВА

```
<properties>
  <!-- Libraries -->
  <spring-version>2.5.6</springVersion>
  <org.slf4j-version>1.7.2</org.slf4j-version>
  <log4j-version>1.2.15</log4j-version>
  <junit-version>4.11</junit-version>
  <joda.time-version>2.0</joda.time-version>

  <!-- Other settings -->
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

  <!-- params -->
  <work.encoding>UTF-8</work.encoding>
</properties>
```

```
${spring-version}
${org.slf4j-version}
${log4j-version}
...
```

Maven

Зависимости

```
<dependencies>
  <dependency>
    <groupId>javax.activation</groupId>
    <artifactId>activation</artifactId>
    <version>${javax.activation-version}</version>
  </dependency>
  <dependency>
    <groupId>javax.xml</groupId>
    <artifactId>jaxrpc-api</artifactId>
    <version>1.1</version>
  </dependency>
</dependencies>
```

Maven

Репозитории

```
<repositories>
  <repository>
    <id>eclipse-platform</id>
    <layout>p2</layout>
    <url>http://download.eclipse.org/...</url>
  </repository>
</repositories>
```

Maven

Профили

```
<profiles>
  <profile>
    <id>oas</id>
    <properties>
      <bindingsPrefix>java:comp/resource/</bindingsPrefix>
      <profileName>oas</profileName>
      <jndiFactoryBean>springJNDIPropertyFactory</jndiFactoryBean>
      <eclipseLinkTargetServer>OC4J</eclipseLinkTargetServer>
    </properties>
  </profile>
  <profile>
    <id>wls</id>
    <properties>
      <bindingsPrefix></bindingsPrefix>
      <profileName>wls</profileName>
      <jndiFactoryBean>jndiPropertyExtractor</jndiFactoryBean>
      <eclipseLinkTargetServer>Weblogic_10</eclipseLinkTargetServer>
    </properties>
  </profile>
</profiles>
```

Maven

Сборка и профили

```
<build>
  <plugins>
    <plugin>
      <groupId>org.eclipse.tycho</groupId>
      <artifactId>tycho-maven-plugin</artifactId>
      <version> `${tycho-version}</version>
      <extensions>>true</extensions>
    </plugin>
  </plugins>
</build>
```

```
# maven clean package -P wls
```

Spring Framework

Вернемся к Spring

DEMO !

См. в github

- `pictures/lectures 1`
- `code.samples/lec.1.spring`

Задача

1. Создать проект maven с поддержкой Spring Framework
 1. Корневой проект: packaging = pom, groupId = ru.<surname>.<initials>, artifactId = mart-parent, version = 1.0
 2. Дочерний проект: packaging = jar, groupId = ru.<surname>.<initials>, artifactId = lec1, version = 1.0
2. Создать необходимые классы, интерфейсы, определения bean'ов согласно определению задачи:
 1. Интерфейс склада IWarehouse
 1. void addProduct(IProduct product, double quantity)
 2. double removeProduct(IProduct product)
 3. List< String > listProducts()
 2. Интерфейс категории ICategory
 1. String getName()
 3. Интерфейс товара IProduct
 1. String getName()
 2. ICategory getCategory()
 3. double getPrice()

Количество товара не является свойством товара, это свойство товара на складе. Товары на склад могут поступать, товары со склада можно забирать. В каждый момент времени мы можем получить выписку о товарах на складе.

Предусмотреть возможность изменения типа склада.

Задача

Предусмотреть возможность изменения типа склада. Под этим понимается другая реализация интерфейса склада. Проверяться будет наличие двух разных вариантов реализации склада, а также возможность их замены через определение bean'ов.

Задачу оформить в виде проекта. Проверятся будут классы и определения bean'ов.

Ремарка

Если вы владеете системой контроля версий GIT, то целесообразно вести работы на GitHub'е.