CZU: 371.3:004

# МЕТОДОЛОГИЯ СОЗДАНИЯ SPRING ПРИЛОЖЕНИЙ

**Ольга ЧЕРБУ**, доктор физико-математических наук, доцент
Государственный университет Молдовы
**Юрий СТАРЕНКО**, студент
**Андрей БОРОДАЙ**, студент
**Штефан БОБЛИК**, студент
Центр передового опыта в области информатики и информационных технологий

**Резюме.** Spring Framework — универсальный фреймворк с открытым исходным кодом для Java-платформы. Spring обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе Java. Spring может быть рассмотрен как коллекция меньших фреймворков или фреймворков во фреймворке. Большинство этих фреймворков может работать независимо друг от друга, однако они обеспечивают большую функциональность при совместном их использовании.

**Abstract**. Spring Framework is an open source general purpose framework for the Java platform. Spring provides solutions to many of the challenges faced by Java developers and organizations looking to build an information system based on the Java platform. Spring can be thought of as a collection of smaller frameworks or frameworks in a framework. Most of these frameworks can work independently of each other, however they provide more functionality when used together.

**Ключевые слова**: spring, bean, maven.
**Keywords**: spring, bean, maven.

## Вступление

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Spring обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе Java [1], [2].

В процессе создания такого рода приложений необходимо сначала работать с Maven-ом, с целью подключения зависимостей. Сначала мы создаем Maven проект из архетипа maven-archetype-webapp.Этот архетип помогает нам создать проект с уже готовыми необходимыми папками. Прописываем зависимости в конфигурационном файле Pom.xml.Этими зависимостями будут: Spring Context, Spring Core, Spring Beans.Фреймворк Spring состоит из нескольких модулей, IOC Container, AOP, DAO, Context, ORM, WEB MVC.

## Применяемые методы и материалы

**Мóдуль** — функционально законченный фрагмент программы. Во многих языках (но далеко не обязательно) оформляется в виде отдельного файла с исходным кодом или поименованной непрерывной её части. Некоторые языки предусматривают объединение модулей в пакеты.

Spring может быть рассмотрен как коллекция меньших фреймворков или фреймворков во фреймворке. Большинство этих фреймворков может работать независимо друг от друга, однако они обеспечивают большую функциональность при совместном их использовании.

Модуль IOC делает код слабосвязанным. В таком случае нет необходимости изменять код, если наша логика переносится в новую среду. В среде Spring за внедрение зависимости отвечает контейнер IOC. Мы предоставляем метаданные контейнеру IOC в виде файла XML или аннотации.

Преимущество внедрения зависимостей: делает код слабо связанным настолько простым в обслуживании, упрощает тестирование кода.

Модуль АОП поддерживает реализацию аспектно-ориентированного программирования, который вы можете использовать для отделения кода.

Стандарт объекта доступа к данным (DAO) — это структурный шаблон, который позволяет нам изолировать уровень приложения / логики от уровня сохраняемости (обычно это реляционная база данных, но это может быть любой другой механизм сохраняемости) с помощью абстрактного API. Функциональность этого API заключается в том, чтобы скрыть от приложения все сложности, связанные с выполнением операций CRUD в базовом механизме хранения. Это позволяет обоим слоям развиваться отдельно, ничего не зная друг о друге.

Контекст приложения Spring, которые отвечают за создание экземпляров, настройку и сборку bean-компонентов путем чтения метаданных конфигурации из XML, аннотаций Java и / или кода Java в файлах конфигурации.

Spring предоставляет API для простой интеграции Spring с такими фреймворками ORM, как Hibernate, JPA (Java Persistence API), JDO (Java Data Objects), Oracle Toplink и iBATIS [4].

API Java Persistence спецификация API Java EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных

Hibernate — это ORM-библиотека для Java, разработанная Гэвином Кингом и выпущенная в начале 2002 года. Кинг разработал Hibernate как альтернативу entity-компонентам для обеспечения устойчивости. Фреймворк был настолько популярен и настолько востребован в то время, что многие из его идей были приняты и кодифицированы в первой спецификации JPA.

Структура Spring Web model-view-controller (MVC) разработана на основе DispatcherServlet, который отправляет запросы обработчикам, с настраиваемыми сопоставлениями обработчиков, разрешением представления, языковым стандартом и разрешением темы, а также поддержкой загрузки файлов. Обработчик по умолчанию основан на аннотациях @Controller и @RequestMapping, предлагая широкий спектр гибких методов обработки. С появлением Spring 3.0 механизм @Controller также

позволяет создавать веб-сайты и приложения RESTful с помощью аннотации @PathVariable и других функций.

В Spring Web MVC вам не нужно реализовывать специфичный для платформы интерфейс или базовый класс [3]. Связывание данных Spring очень гибкое: например, он рассматривает несоответствие типов как ошибки проверки, которые могут быть оценены приложением, а не как системные ошибки. Таким образом, вам не нужно дублировать свойства бизнес-объектов в виде простых, нетипизированных строк в объектах формы для обработки недопустимых отправлений или для правильного преобразования строк. Вместо этого часто предпочтительнее выполнять прямую привязку к бизнес-объектам.

**Полученные результаты**

Как создать Spring приложение:

1) Создаем Maven проект.Добавляем архетип maven-archetype-webapp.Данный архетип генерирует структуру проекта по файлам.

`maven-archetype-webapp` is an archetype which generates a sample Maven webapp project:

```
project
|-- pom.xml
`-- src
    `-- main
        `-- webapp
            |-- WEB-INF
            |   `-- web.xml
            `-- index.jsp
```

Подключаем зависимости в файле pom.xml. Нам нужны: Spring Beans,Sprint Context, Spring Core.Далее мавен попросит установить все зависимости и мы будем готовы к работе.

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>5.3.9</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-beans</artifactId>
  <version>5.3.9</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>5.3.9</version>
</dependency>
```

**Рис. 1. Maven**

2) Создаем папку resources.Помечаем ее типом resources в IDE.

3) Создаем папку джава и помечаем ее как Source Root.

4) Создаем обычный класс в папке джава, например TestBean.

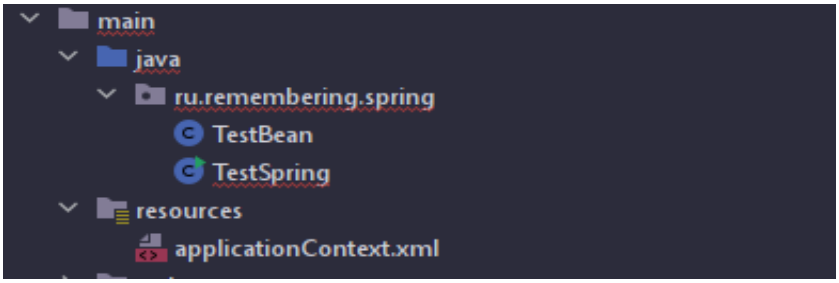



**Рис. 2. Classes**

5) Создем конструктор + сеттеры и геттеры.

```
package ru.remembering.spring;

public class TestBean {
    private String name;

    public TestBean(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

**Рис. 3. Class TestBean**

6) В папке ресурсов мы создаем applicationConfig.xml.

```
<bean                                                    id="testBean"
      class="ru.remembering.spring.TestBean">
    <!--<constructor-arg> - указывает Spring'у что Spring container должен
искать зависимости в конструкторе класса, определенный в атрибуте бина.--
>
    <constructor-arg                                     value="СЕЙТТ"/>
</bean>
```

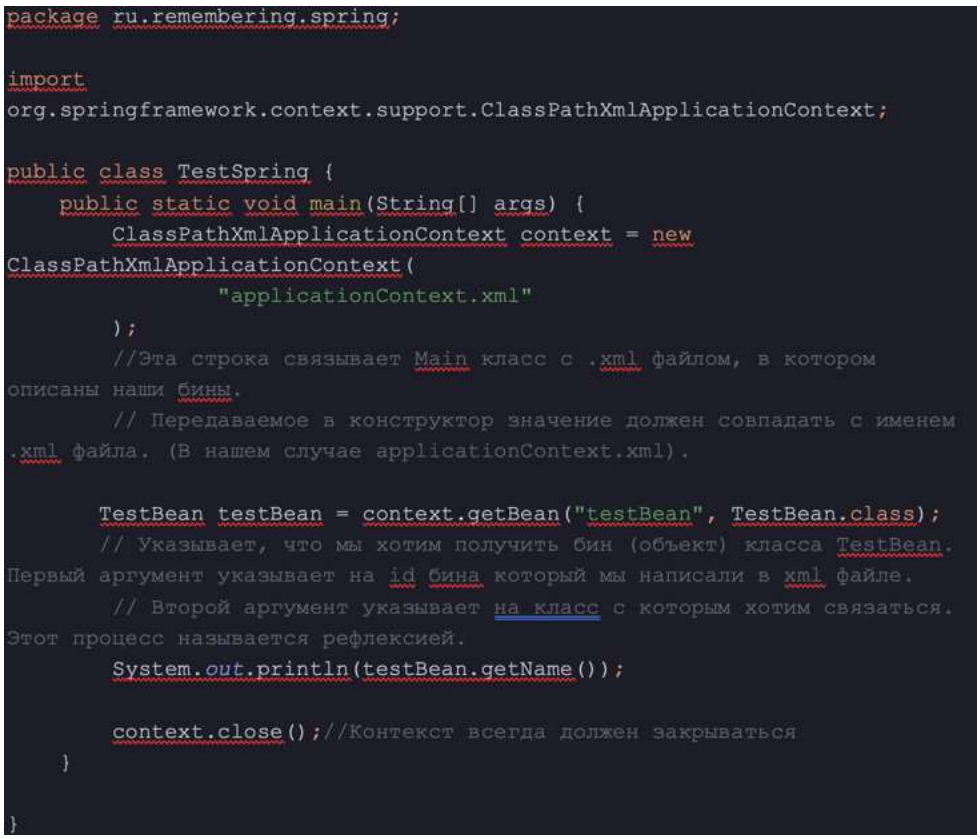
**Рис. 4. applicationConfig.xml**

7) В нем мы прописываем “путь к классу” TestBean.Тем самым мы связываем класс тестбин и класс ТестСпринг.

8) Создаем клссс тестспринг (Рис. 5).

**Выводы**

Модуль IOC делает код слабосвязанным. В таком случае нет необходимости изменять код, если наша логика переносится в новую среду. В среде Spring за внедрение зависимости отвечает контейнер IOC. Мы предоставляем метаданные контейнеру IOC в виде файла XML или аннотации. Преимущество внедрения зависимостей: делает код слабо связанным настолько простым в обслуживании, упрощает тестирование кода.



```
package ru.remembering.spring;

import
org.springframework.context.support.ClassPathXmlApplicationContext;

public class TestSpring {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext(
                "applicationContext.xml"
        );
        //Эта строка связывает Main класс с .xml файлом, в котором
описаны наши бины.
        // Передаваемое в конструктор значение должен совпадать с именем
.xml файла. (В нашем случае applicationContext.xml).

       TestBean testBean = context.getBean("testBean", TestBean.class);
        // Указывает, что мы хотим получить бин (объект) класса TestBean.
Первый аргумент указывает на id бина который мы написали в xml файле.
        // Второй аргумент указывает на класс с которым хотим связаться.
Этот процесс называется рефлексией.
        System.out.println(testBean.getName());

        context.close();//Контекст всегда должен закрываться
    }

}
```

**Рис. 5. Class TestSpring**

**Библиография**


1. Spring Framework Documentation (Accessed on 06.09.2021) https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/orm.htm.
2. Spring Boot Tutorial (Accessed on 06.09.2021) https://www.javatpoint.com/spring-boot-tutorial
3. Introduction to Spring Web MVC framework (Accessed on 06.09.2021) https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html
4. JPA и спящий режим (Accessed on 06.09.2021) https://ru.realiventblog.com/18-what-is-jpa-introduction-to-the-java-persistence-api