DI-контейнер

Вольф Илья, гр. 19213

Чиёсов Игорь, гр. 19214

Функционал

- Язык: Java
- Поддержка аннотаций: Component, Inject, Value, Autowired, Scope
- Внедрение в: поля и конструкторы
- Поддержка JSON конфигурационных файлов для хранения значений, которые могут внедряться через аннотацию Value
- Scopes: Singleton, Prototype, Thread
- Нахождение и (частичное) разрешение циклических зависимостей
- Взаимодействие с DI-контейнером: получение объектов, мониторинг

Разделение обязанностей

Илья

- Внедрение через поля
- Разрешение циклов
- Реализация Scopes
- Реализация тест-кейсов

Игорь

- Внедрение через конструкторы
- Обнаружение и маркировка циклов
- Архитектура
- Реализация тест-кейсов

Апи контроллера

@Component

```java
@Component("rockMusic")
public class RockMusic implements Music {
    @Override
    public String getSong() {
        return "Highway to Hell";
    }
}
```

```java
@Component
public class ClassicalMusic implements Music {
    @Override
    public String getSong() {
        return "In the Hall of the Mountain King ";
    }
}
```

@Scope

```
@Component("classicalMusic")
@Scope(SINGLETON)
public class ClassicalMusic im
    @Override
```

```
@Component
@Scope(PROTOTYPE)
public class MusicPlayer {
```

```
@Scope(THREAD)
@Component("rockMusic")
public class RockMusic impl
    @Override
    public String getSong()
```

Внедрение в поля

```java
@Inject
private MusicPlayer musicPlayer;
```

```java
@Component
public class MusicPlayer {

    @Inject("classicalMusic")
    private Music classicalMusic;

    @Inject("rockMusic")
    private Music rockMusic;
```

```json
{
  "cost" : 599999,
  "name" : "Mac Pro",
  "list" : [
    "a",
    "bc",
    "def"
  ]
}
```

```java
@Value("$name")
private String name;

@Value("$cost")
private int cost;

@Value("$list")
private ArrayList<String> words;
```

```java
@Value("iMac")
private String name;

@Value("179999")
private int cost;
```

Инъекция в конструктор

```
@Autowired
public RemoteDataSource(@Value("$url") String url) {
    this.url = url;
}
```

```
@Autowired
public UsersController(UsersRepository users) {
    this.users = users;
}
```

```
@Autowired
public MailController(
        @Inject("googleMailer") MailService googleMailer,
        @Inject("yandexMailer") MailService yandexMailer
) {
    this.googleMailer = googleMailer;
    this.yandexMailer = yandexMailer;
}
```

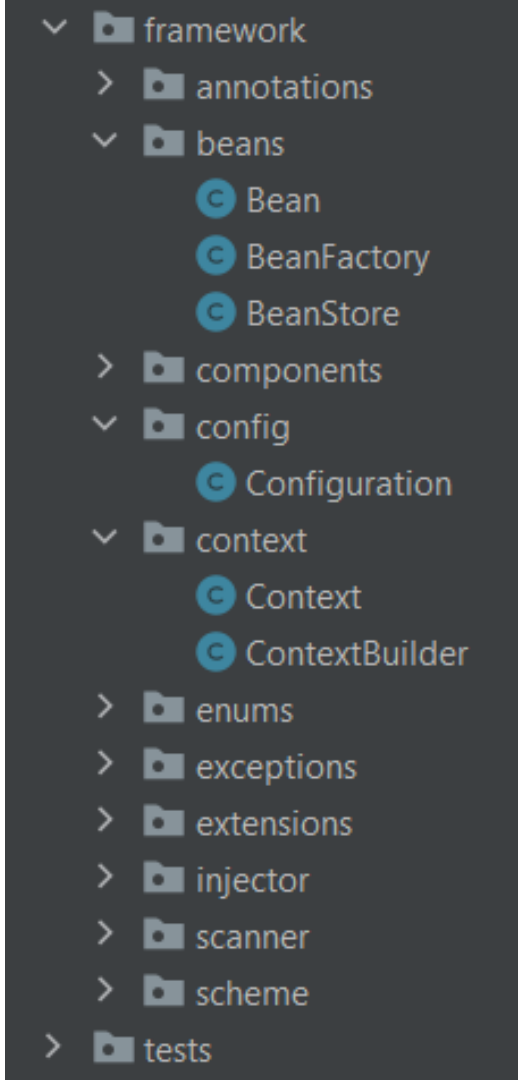
```
@Autowired
public GenericController(
        GenericRepository<User> users,
        GenericRepository<Mail> mails
) {
    this.users = users;
    this.mails = mails;
}
```

Получение объектов с контейнера

```
public static void main (String[] args) {
    var context :Context = new ContextBuilder()
            .setConfiguration("config.json")
            .Build();

    context.run(Server.class);

    var mailer :GoogleMailService = context.getType(GoogleMailService.class);
    mailer.sendMail();

    var usersController :UsersController = context.getType(UsersController.class);
    usersController.addUser( name: "Nick");
    usersController.addUser( name: "Tom");
    usersController.everyoneSayHello();

    var mailsController :MailsController = context.getType(MailsController.class);
    mailsController.sendGoogleMail();
    mailsController.sendYandexMail();
}
```

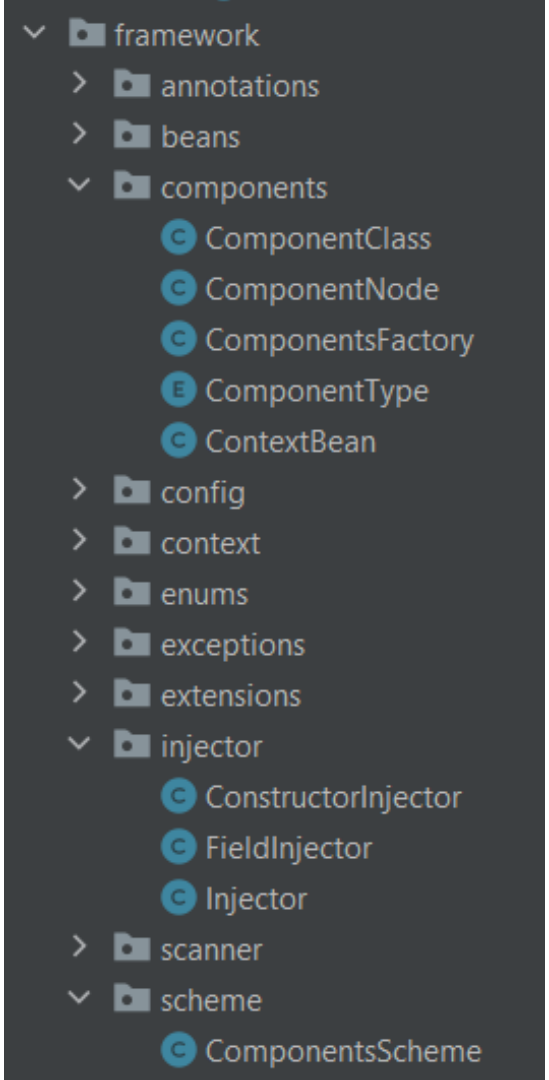
Архитектура

- Context - основа контейнера
- ContextBuilder - позволяет собрать контекст с различными параметрами (в частности подключение конфига)
- Configuration - отвечает за json-конфигурацию
- Bean - хранит всю информацию об объекте
- BeanStore - хранилище бинов
- BeanFactory - отвечает за создание бинов



Архитектура

- Injector - отвечает за внедрение зависимостей (в поля, в конструкторы)
- ComponentScheme - граф компонентов, результат препроцессинга.
- ComponentNode - узловые компоненты со всей необходимой информацией для создания бинов.



Демонстрация