Focused forward

Netcracker

Lecture 6. Spring Framework. Spring MVC. Introduction.

Alexei Khudnitsky

Что такое Spring?

Spring Framework (или коротко **Spring**) — универсальный фреймворк с открытым исходным кодом для Java-платформы.

Обеспечивает базовую поддержку управления зависимостями, управление транзакциями веб-приложений, доступ к данным, обмен сообщениями и многое другое.

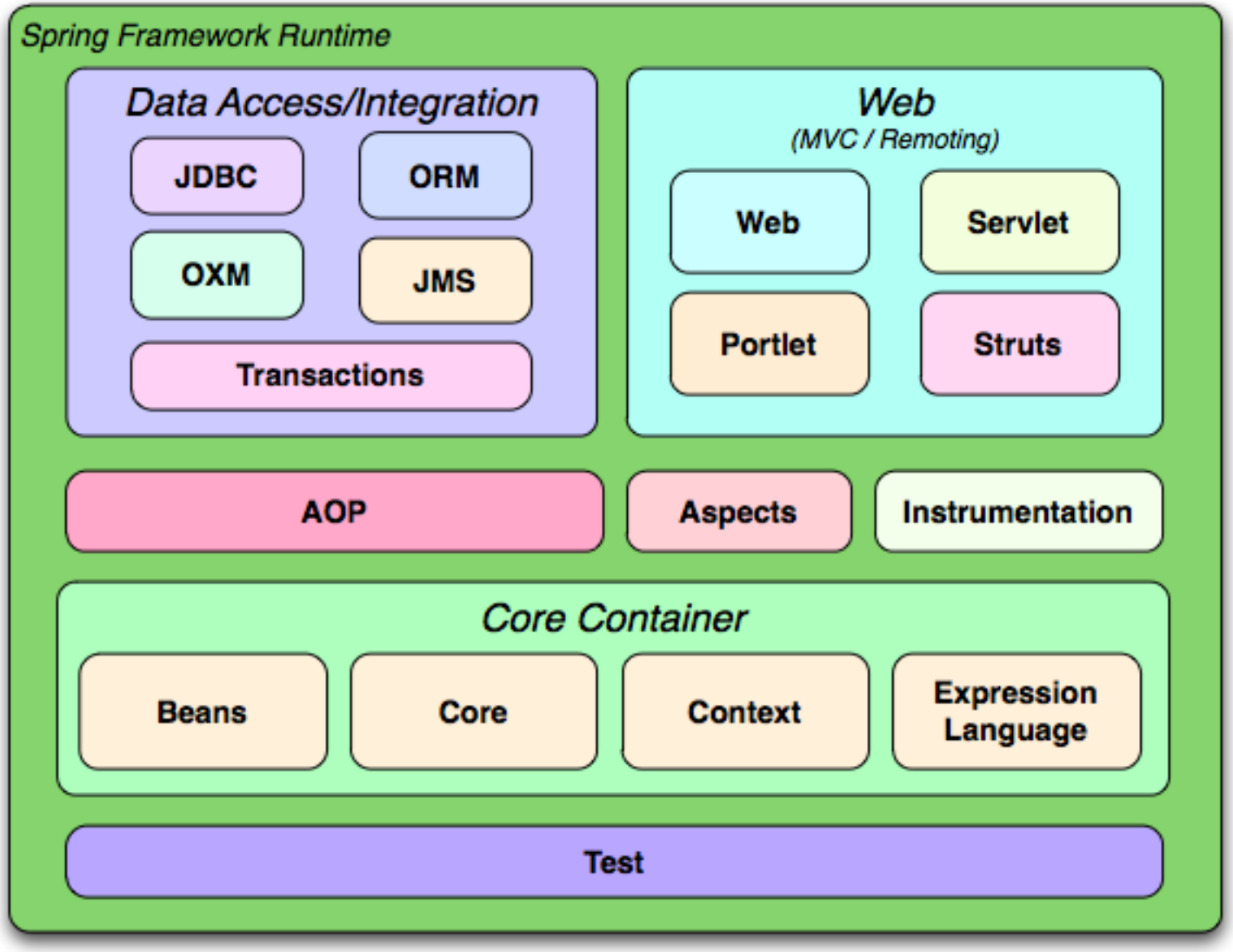
Достоинства:

- 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

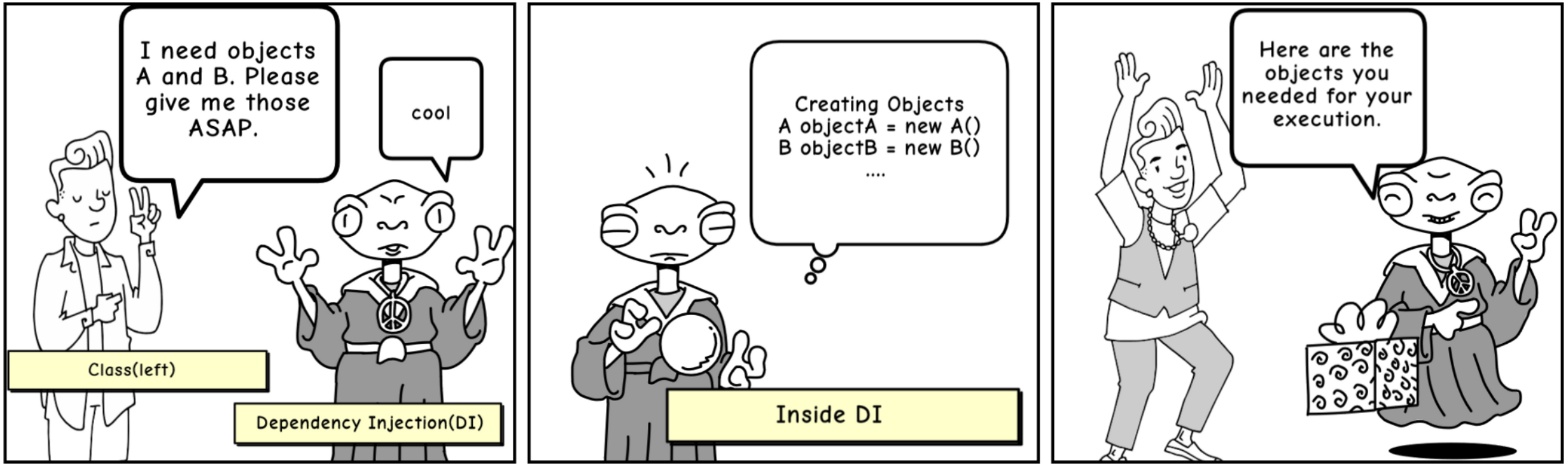
Что дает Spring

- Внедрение зависимости
- Аспектно-ориентированное программирование, включая декларативное управление транзакциями
- Создание Spring MVC web-приложений и RESTful web-сервисов
- Начальная поддержка JDBC, JPA, JMS
- Многое другое...

Spring Framework



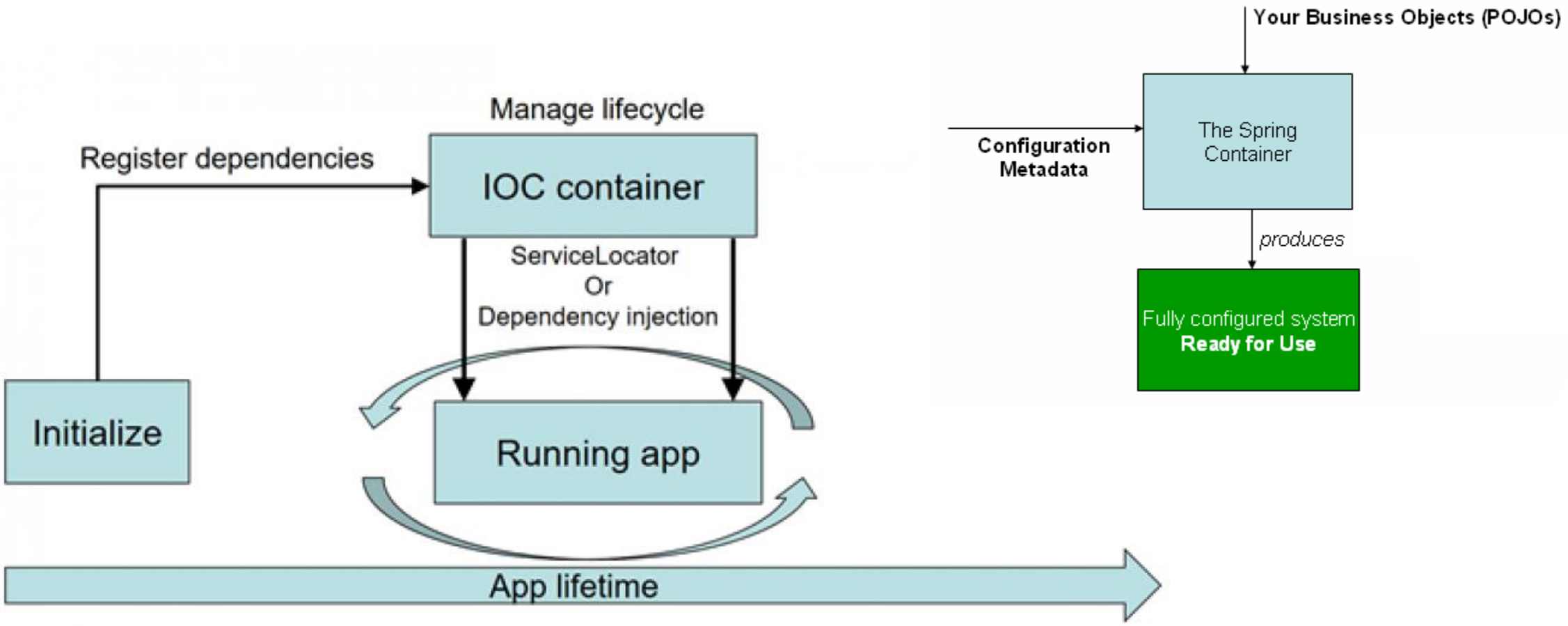
Dependency Injection



Inversion of Control & Dependency Injection

- **Inversion of Control (инверсия управления)** — это некий абстрактный принцип, набор рекомендаций для написания слабо связанного кода. Суть которого в том, что каждый компонент системы должен быть как можно более изолированным от других, не полагаясь в своей работе на детали конкретной реализации других компонентов.
- **Dependency Injection (внедрение зависимостей)** — это одна из реализаций этого принципа (помимо этого есть еще Factory Method, Service Locator).

IoC container



Spring BeanFactory Container

org.springframework.beans.factory.BeanFactory

Spring ApplicationContext Container

org.springframework.context.ApplicationContext

Чаще всего используются следующие реализации AppicationContext:

- **FileSystemXmlApplicationContext**
- **ClassPathXmlApplicationContext**
- **WebXmlApplicationContext**

Inversion of Control & Dependency Injection

```
public class ScheduleManager {
    public Schedule getSchedule(){
        // Do Something by init schedule...
        return null;
    }
}
```

```
public class ScheduleViewer {
    private ScheduleManager scheduleManager = new ScheduleManager();

    public void renderSchedule()
    {
        scheduleManager.getSchedule();
        // Do Something by render schedule...
    }
}
```

ScheduleViewer scheduleViewer =
new ScheduleViewer(new ScheduleManager());

```
public interface IScheduleManager {
    Schedule getSchedule();
}
```

```
public class ScheduleManager implements IScheduleManager{
    public Schedule getSchedule(){
        // Do Something by init schedule...
        return null;
    }
}
```

```
public class ScheduleViewer {
    private IScheduleManager scheduleManager;

    public ScheduleViewer(IScheduleManager scheduleManager) {
        this.scheduleManager = scheduleManager;
    }

    public void renderSchedule()
    {
        scheduleManager.getSchedule();
        // Do Something by render schedule...
    }
}
```

Spring core annotations. @Autowired

```
1  class Car {
2      Engine engine;
3
4      @Autowired
5      Car(Engine engine) {
6          this.engine = engine;
7      }
8  }
```

```
1  class Car {
2      Engine engine;
3
4      @Autowired
5      void setEngine(Engine engine) {
6          this.engine = engine;
7      }
8  }
```

```
1  class Car {
2      @Autowired
3      Engine engine;
4  }
```

Spring core annotations. @Bean

```java
@Bean
Engine engine() {
    return new Engine();
}
```

```java
@Bean("engine")
Engine getEngine() {
    return new Engine();
}
```

Spring core annotations. @Qualifier

```
class Bike implements Vehicle {}

class Car implements Vehicle {}
```

```
@Autowired
Biker(@Qualifier("bike") Vehicle vehicle) {
    this.vehicle = vehicle;
}
```

```
@Autowired
@Qualifier("bike")
Vehicle vehicle;
```

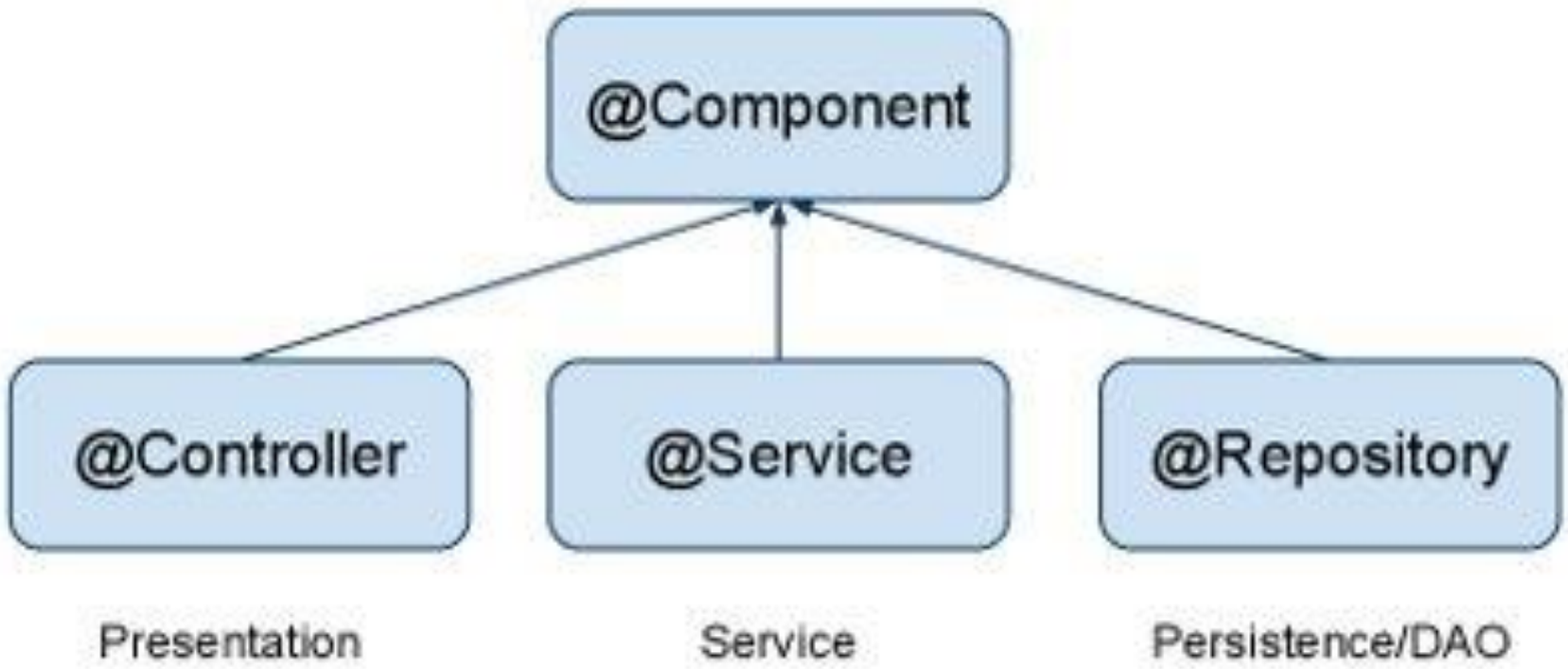
Spring core annotations. @Scope

```
1  @Component
2  @Scope("prototype")
3  class Engine {}
```

Spring bean annotations. *@ComponentScan*

```
1  @Configuration
2  @ComponentScan(basePackages = "com.baeldung.annotations")
3  @ComponentScan(basePackageClasses = VehicleFactoryConfig.class)
4  class VehicleFactoryConfig {}
```

Spring bean annotations. @Component, @Repository, @Service, @Controller



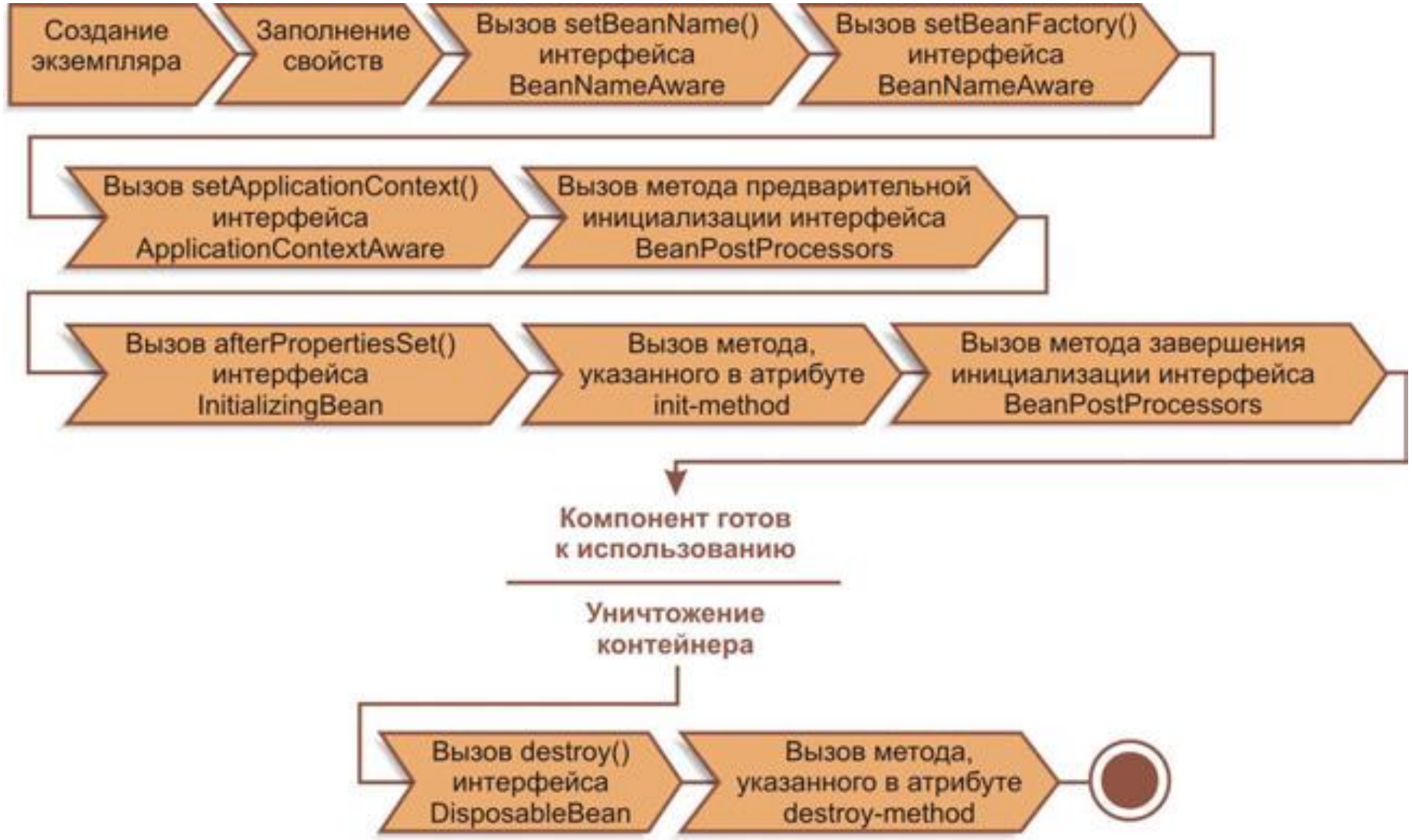
Spring bean annotations. @Configuration

```
1  @Configuration
2  class VehicleFactoryConfig {
3
4      @Bean
5      Engine engine() {
6          return new Engine();
7      }
8
9  }
```

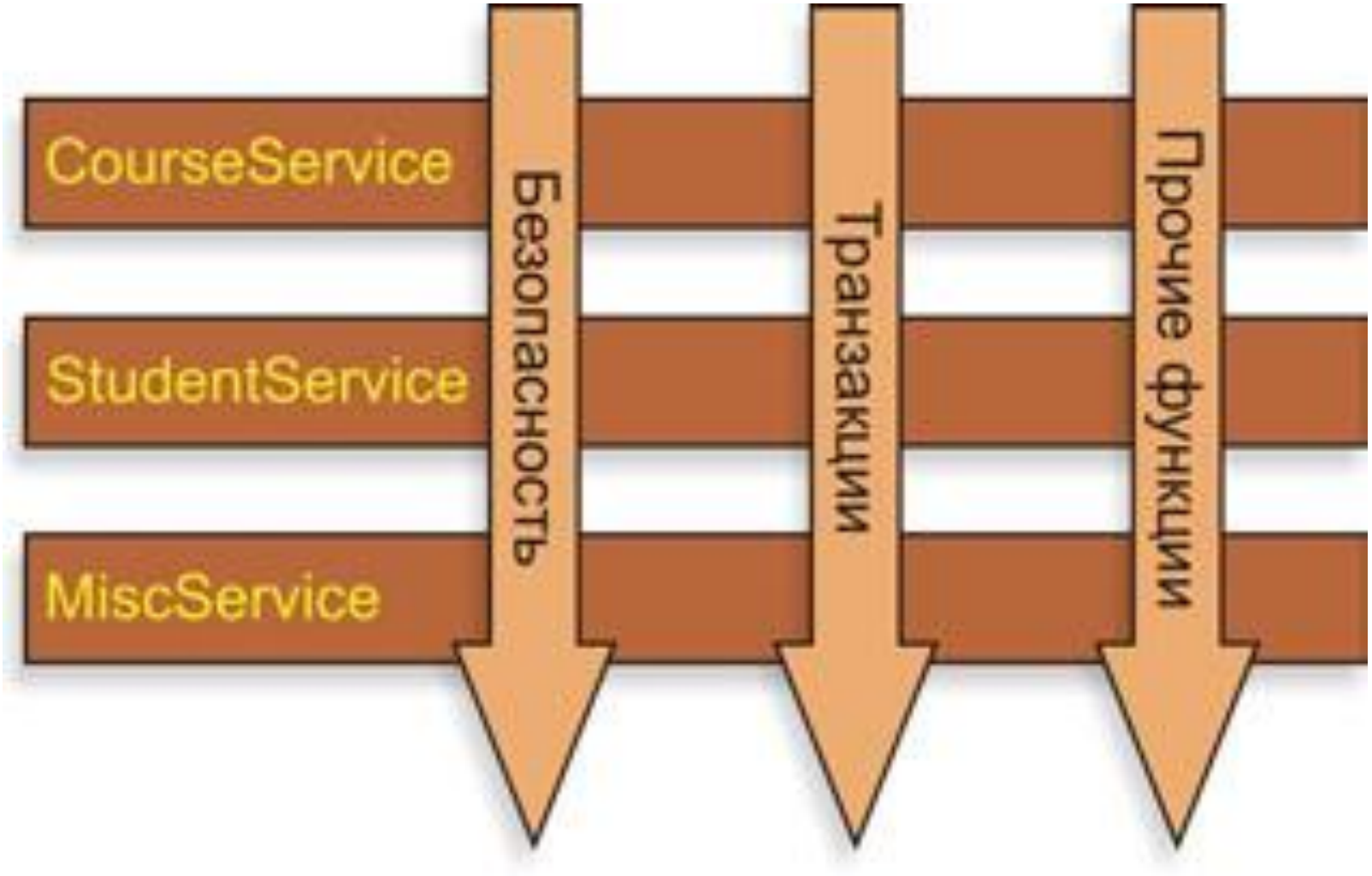
Bean scopes

Scope	Description
singleton	(Default) Scopes a single bean definition to a single object instance per Spring IoC container.
prototype	Scopes a single bean definition to any number of object instances.
request	Scopes a single bean definition to the lifecycle of a single HTTP request; that is, each HTTP request has its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring `ApplicationContext`.
session	Scopes a single bean definition to the lifecycle of an HTTP `Session`. Only valid in the context of a web-aware Spring `ApplicationContext`.
global session	Scopes a single bean definition to the lifecycle of a global HTTP `Session`. Typically only valid when used in a portlet context. Only valid in the context of a web-aware Spring `ApplicationContext`.
application	Scopes a single bean definition to the lifecycle of a `ServletContext`. Only valid in the context of a web-aware Spring `ApplicationContext`.

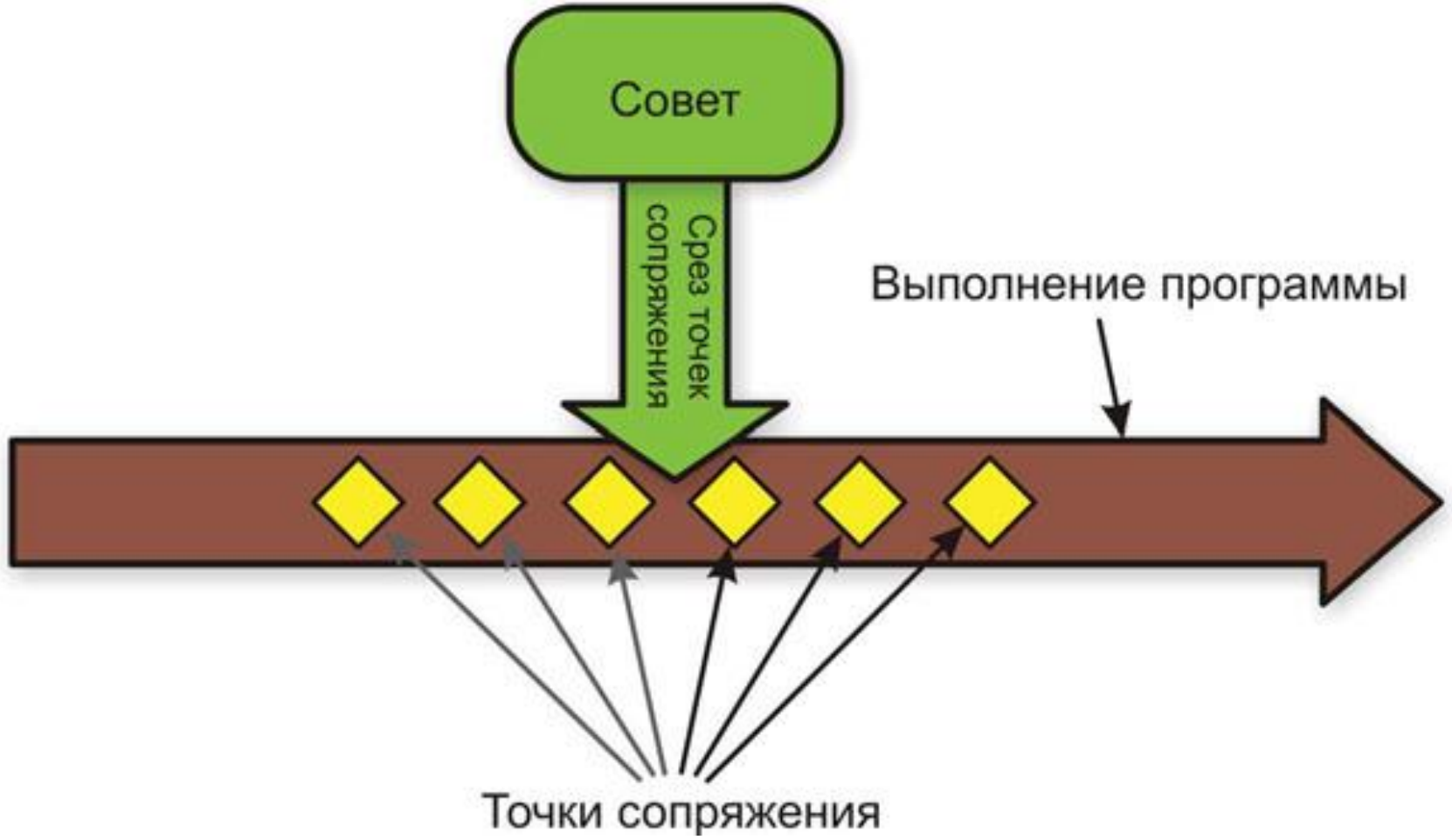
Bean lifecycle



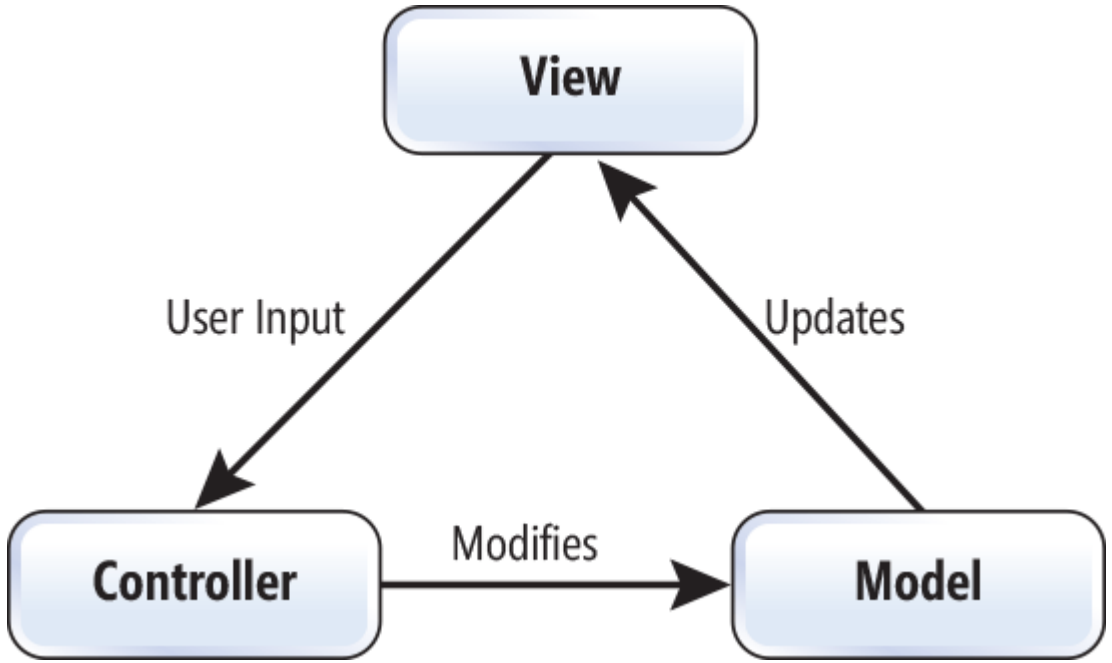
Spring AOP


CourseService
StudentService
MiscService
Безопасность
Транзакции
Прочие функции

Spring AOP

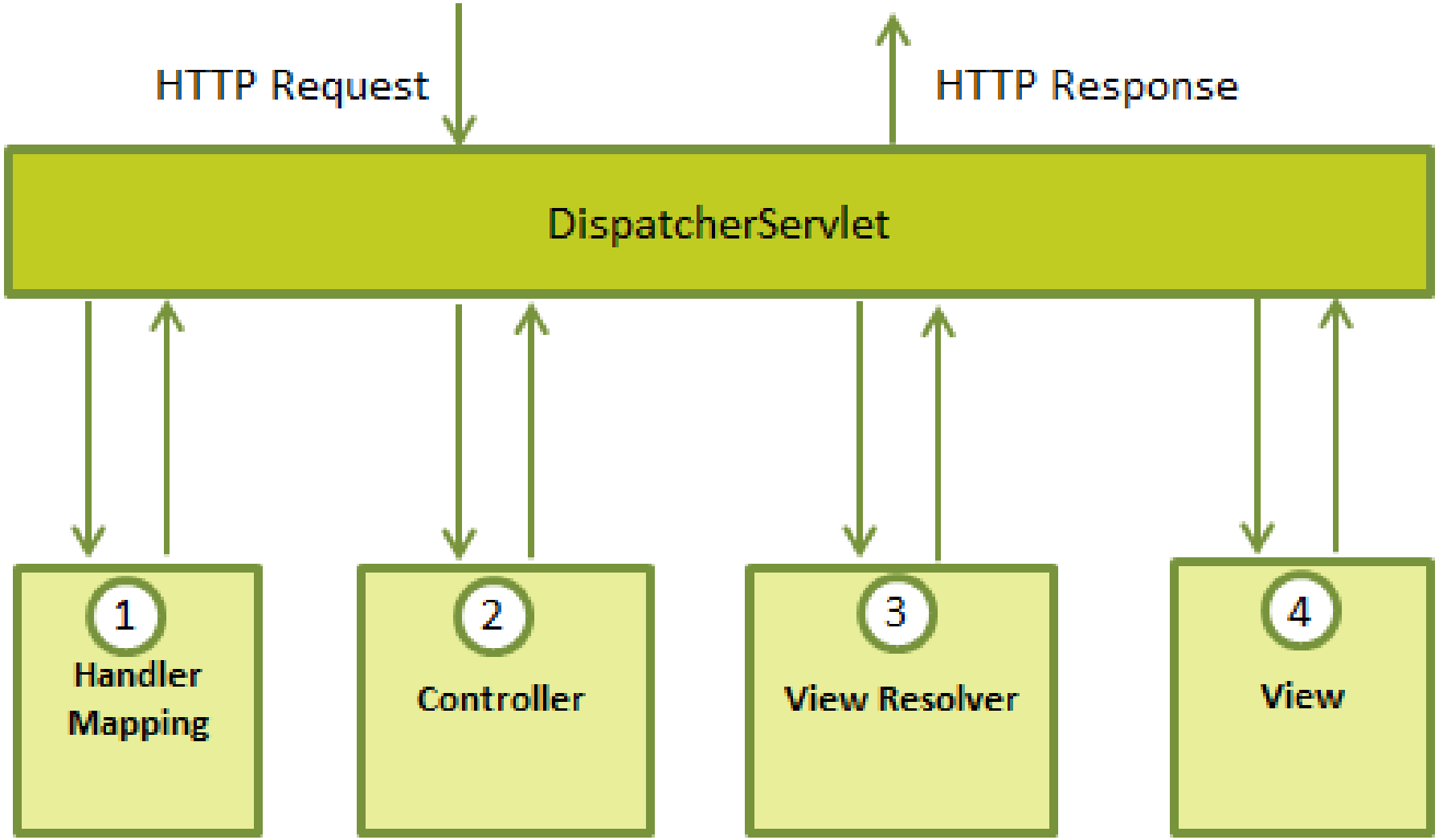

Совет
Срез точек сопряжения
Выполнение программы
Точки сопряжения

MVC



- Model:
  - Represents the state of the application's domain
  - Exposes operations that can be performed on the Model by the Controller
- Controller:
  - Provides an interface between the Model and the View to facilitate user actions on the Model
- View
  - Responsible for rendering the application's UI
  - Displays the state of the Model (or part of it)

Spring MVC


HTTP Request
HTTP Response
DispatcherServlet
1
Handler Mapping
2
Controller
3
View Resolver
4
View

Литература

1. https://docs.spring.io/spring/docs/4.0.x/spring-framework-reference/html/overview.html
2. https://habr.com/post/131993/ (IoC, DI, IoC-контейнер — Просто о простом)
3. https://o7planning.org/ru/10127/spring-tutorial-for-beginners#a4654239 (Руководство Spring для начинающих)
4. https://www.baeldung.com/spring-intro (Spring Framework Introduction)
5. https://habr.com/post/336816/ (Spring MVC — основные принципы)
6. https://www.tutorialspoint.com/spring/spring_mvc_hello_world_example.htm
7. Spring в действии, Крейг Уоллс

Q&A

Thank You