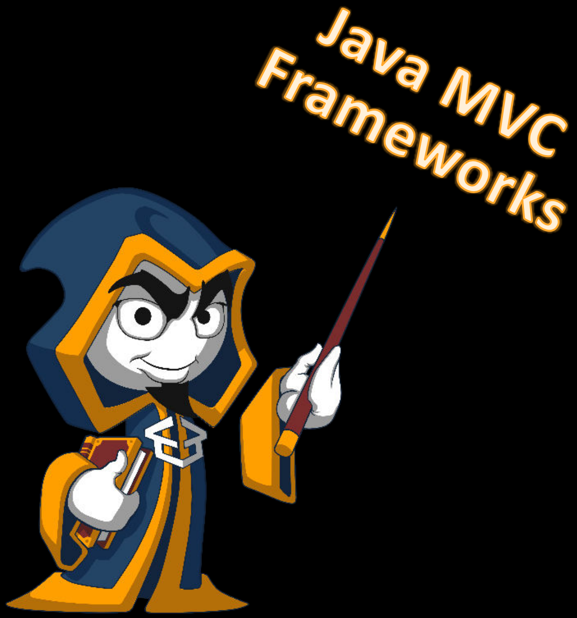
Java MVC Frameworks

The Right Way: Architecture

SoftUni Foundation



SoftUni Team

Technical Trainers

Software University

http://softuni.bg

spring

Table of Contents

Questions

SoftUni Foundation

sli.do

#java-web

SoftUni Foundation

Inversion of Control

Constructor vs Field Injection

Field Injection

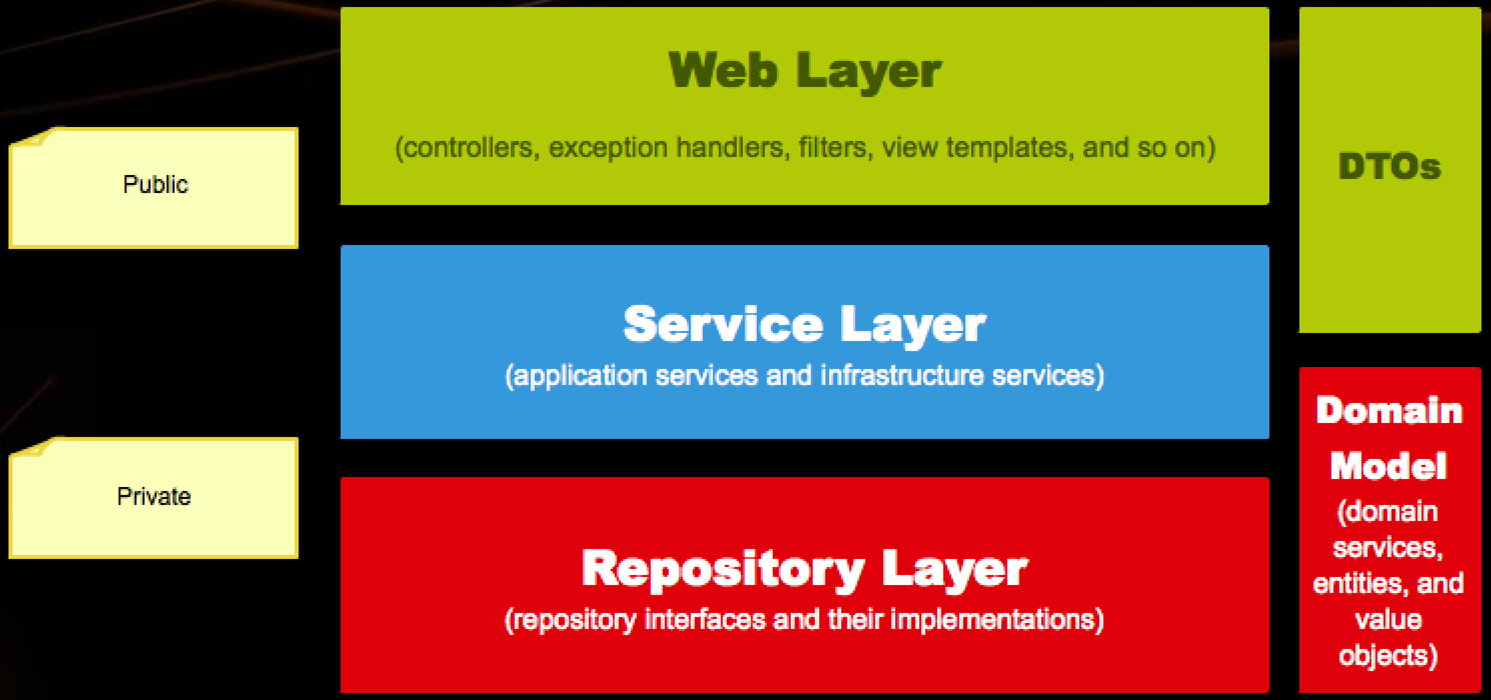
- Easy to write
- Easy to add new dependencies
- It hides potential architectural problems!

```
@Autowired
private ServiceA serviceA
@Autowired
private ServiceB serviceB
@Autowired
private ServiceC serviceC
```

Constructor Injection

- Time Consuming
- Harder to add dependencies
- It shows potential architectural problems!

```
@Autowired
public ControllerA(ServiceA serviceA, ServiceB serviceB,
ServiceC serviceC) {
   this.serviceA = serviceA;
   this.serviceB = serviceB;
   this.serviceC = serviceC;
}
```

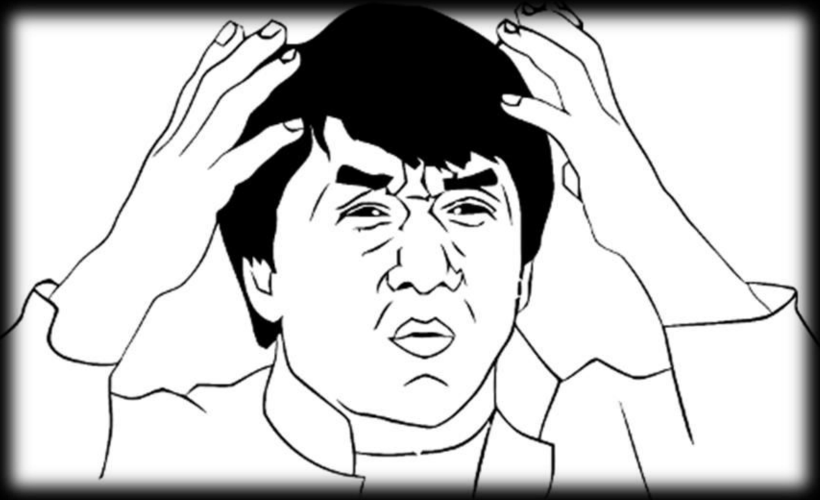

Public
Private
Web Layer
(controllers, exception handlers, filters, view templates, and so on)
Service Layer
(application services and infrastructure services)
Repository Layer
(repository interfaces and their implementations)
DTOs
Domain Model
(domain services, entities, and value objects)

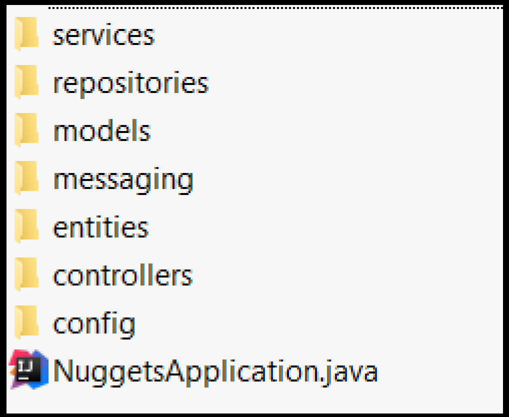
Layers

The Correct Project Structure

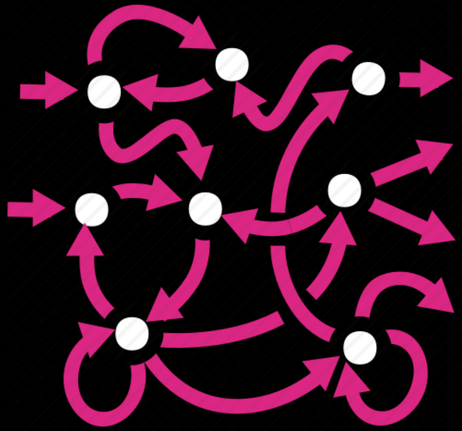
Layers

- We are used to splitting our code based on its functionality:





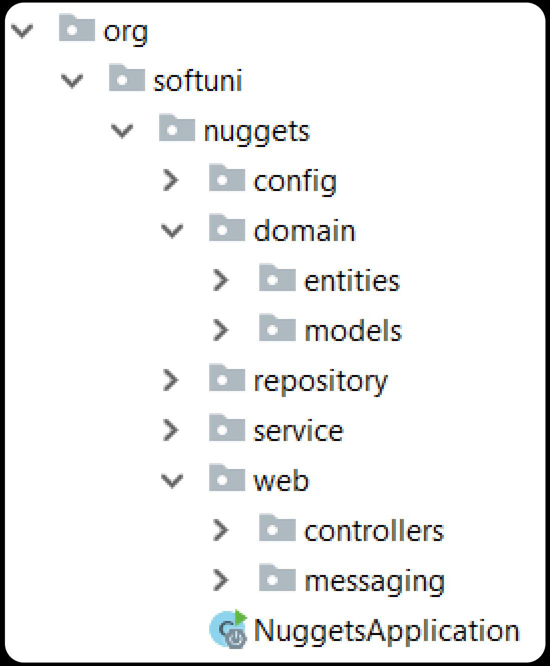
```
services
repositories
models
messaging
entities
controllers
config
NuggetsApplication.java
```



- It gets hard to navigate in bigger applications

Layers (2)

- Splitting the project into different modules
  - Each module corresponding to the application layer
  - Makes it easier to navigate



```
org
  softuni
    nuggets
      config
      domain
        entities
        models
      repository
      service
      web
        controllers
        messaging
      NuggetsApplication
```

Thin Controllers

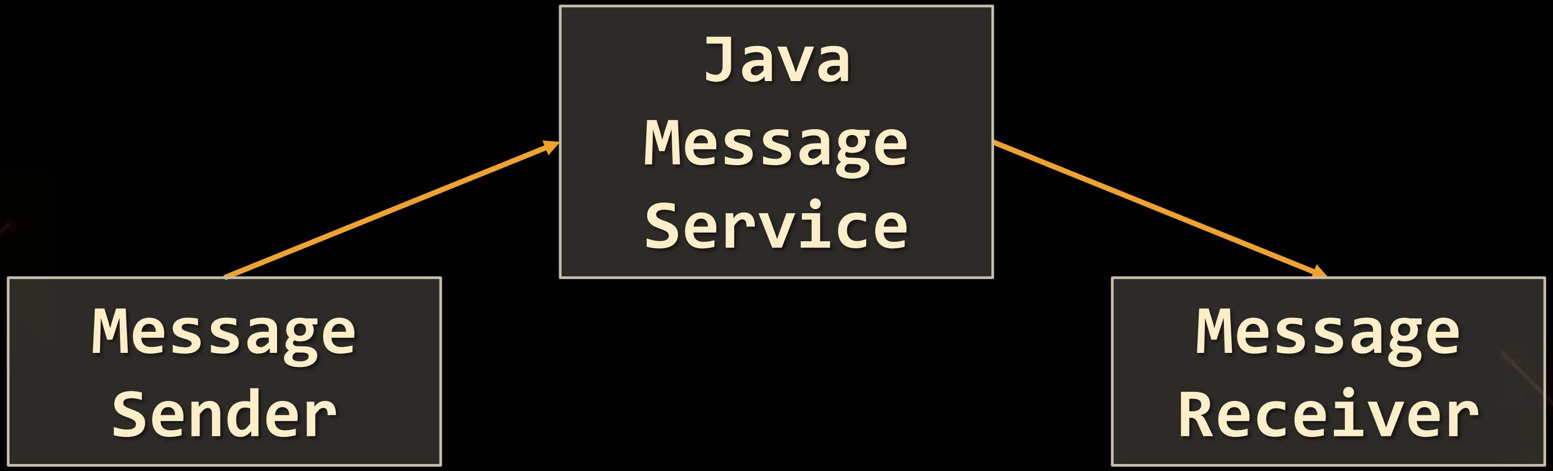
Creating Simple Components

Thin Controllers

- Controllers should follow well known principles such as DRY and KISS
- Should delegate functionality to the model layer
- The model layer consists of application logic, e.g. services, executors, strategies, mappers, DTOs, entities, etc.

Thin Controller Example

```java
@PreAuthorize("isAuthenticated()")
@GetMapping("{id}")
public ModelAndView details(ModelAndView modelAndView,
@PathVariable Long id) {
  GameDetailsView game = gameService.get(id);

  modelAndView.setViewName("index");
  modelAndView.addObject("game", game);
  modelAndView.addObject("title", game.getTitle());

  return modelAndView;
}
```

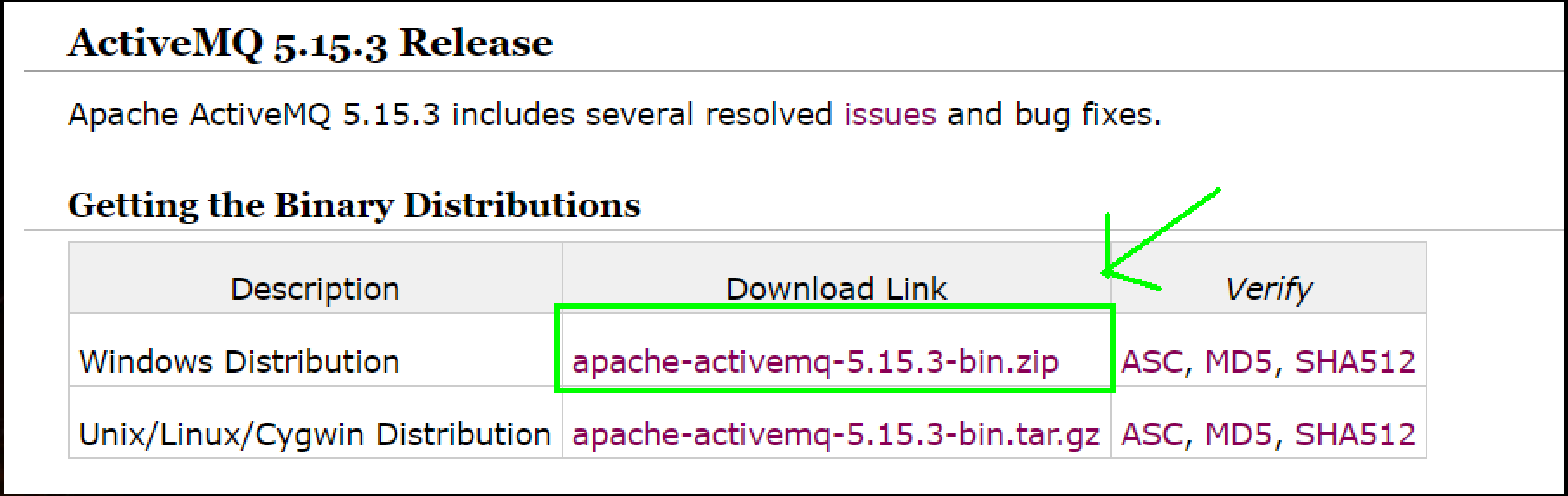


JMS

Sending message between applications

JMS - What we need?

- Apache ActiveMQ.
- Download here.



ActiveMQ 5.15.3 Release

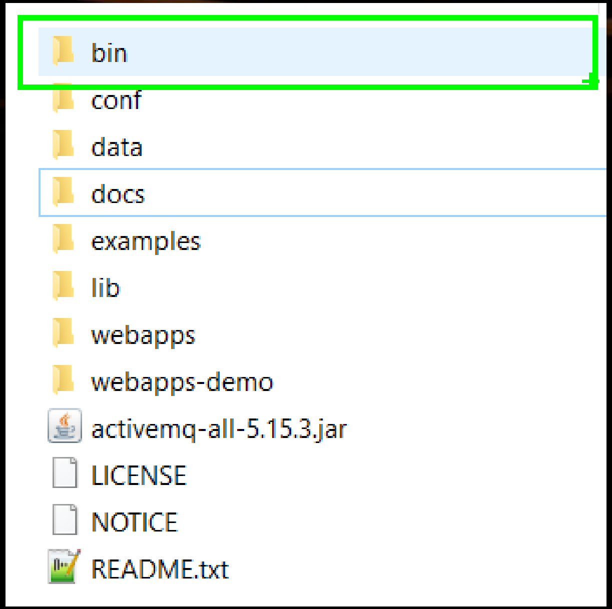
Apache ActiveMQ 5.15.3 includes several resolved issues and bug fixes.

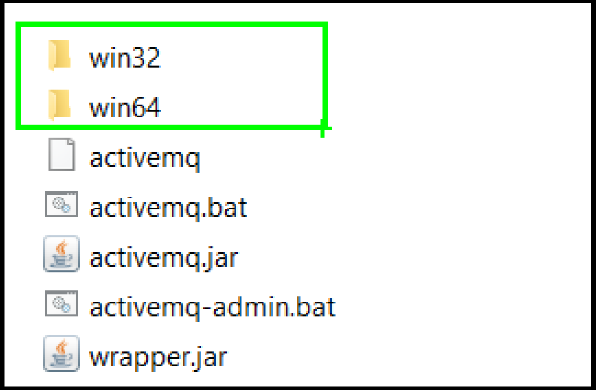
Getting the Binary Distributions

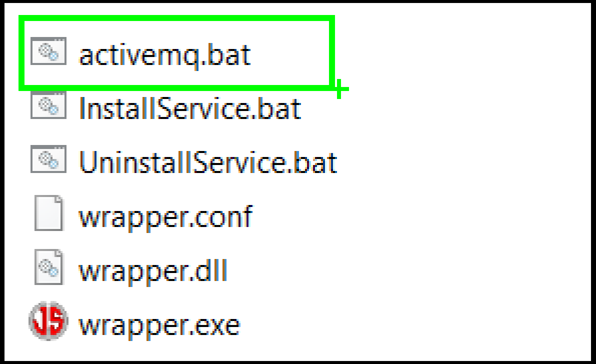
Description	Download Link	*Verify*
Windows Distribution	apache-activemq-5.15.3-bin.zip	ASC, MD5, SHA512
Unix/Linux/Cygwin Distribution	apache-activemq-5.15.3-bin.tar.gz	ASC, MD5, SHA512

JMS - What we need?

- Unzip the archive.
- Go to bin/ folder.
- Depending on your OS, chose one of the 2 folders.
- Run activemq.bat.


bin
conf
data
docs
examples
lib
webapps
webapps-demo
activemq-all-5.15.3.jar
LICENSE
NOTICE
README.txt


win32
win64
activemq
activemq.bat
activemq.jar
activemq-admin.bat
wrapper.jar


activemq.bat
InstallService.bat
UninstallService.bat
wrapper.conf
wrapper.dll
wrapper.exe

JMS - What we need?

SoftUni Foundation

- Apache ActiveMQ.
- Maven Dependencies:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-activemq</artifactId>
</dependency>
```

JMS - Connection Factory

- Creating connection to the ActiveMQ service

```
String DEFAULT_BROKER_URL = "tcp://localhost:61616";

@Bean
public ActiveMQConnectionFactory connectionFactory() {
  ActiveMQConnectionFactory connectionFactory =
                                new ActiveMQConnectionFactory();
  connectionFactory.setBrokerURL(DEFAULT_BROKER_URL);
  return connectionFactory;
}
```

JMS - Sending Messages

- You will need to create JmsTemplate Bean that will use the connection factory from the previous slide

```
@Bean
public JmsTemplate jmsTemplate(){
  JmsTemplate template = new JmsTemplate();
  template.setConnectionFactory(connectionFactory());
  return template;
}
```

JMS - Sending Messages (2)

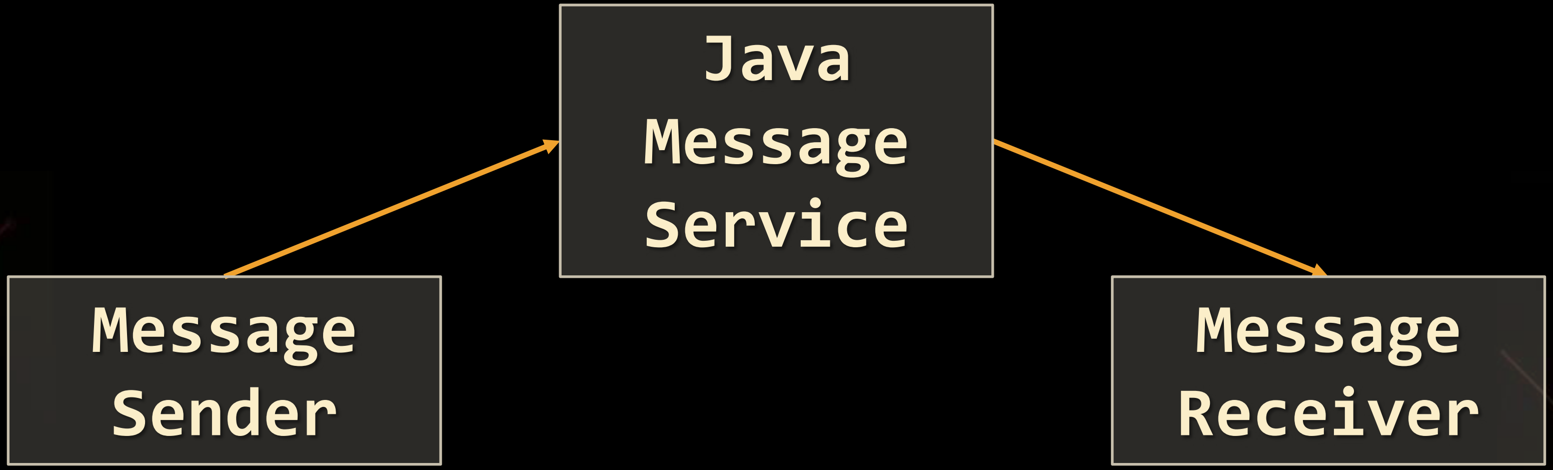
- To send a message you only need to inject the bean and use the `convertAndSend()` method:

```
@Autowired
private JmsTemplate jmsTemplate;

public void sendMessage(final String message) {
  jmsTemplate.convertAndSend(message);
}
```

JMS - Receiving Messages

- To receive a message in the other application just use the `@JmsListener` annotation:

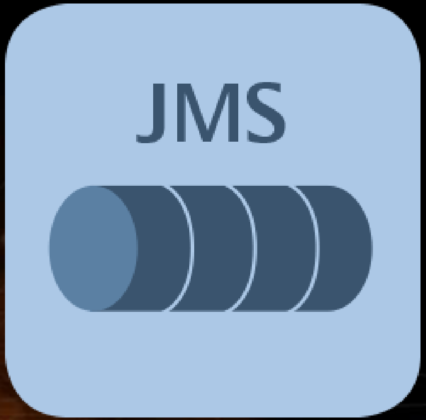
```
@JmsListener(destination = "message-queue")
public void readMessage(Message<String> message) throws
JMSException {

  System.out.println(message.getPayload());
}
```

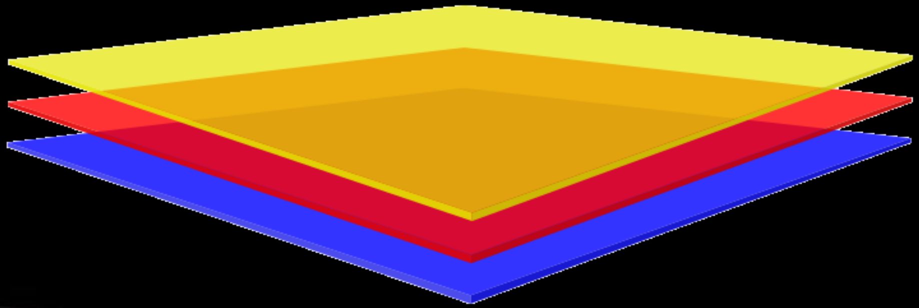


Live Demo

Summary

- Constructor injection – the best way for DI
- Splitting your application code by layers
  - Each layer has its own module
- Every component should be as "thin" as possible
- JMS lets multiple applications communicate





Java MVC Frameworks – Architecture

SoftUni Foundation

https://softuni.bg/courses/java-mvc-frameworks-spring

SoftUni Diamond Partners

SoftUni Foundation

INDEAVR
Serving the high achievers

INFRAGISTICS®

SoftwareGroup
doing it right

xssoftware

NETPEAK
SEO and PPC for Business

SUPER HOSTING®.BG

SoftUni Diamond Partners

SoftUni Foundation

SBTech we know sports

telenor

codexio

SmartIT

æternity VENTURES

LIEBHERR

License

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg