Spring Basics Workshop – Exercises

PROCEDURE

This workshop provides better understanding of the basics of the spring-framework, spring-mvc and spring-boot. It has been divided into sub-sections like spring-framework I, spring-framework II and spring-boot. Exercises are provided at the end of each sub-section, which could be done in group or alone, to apply and refresh the concepts that have been explained.

SYSTEM REQUIREMENTS

- ⇒ OpenJDK 11
- ⇒ Eclipse / Intellij IDEA
- ⇒ STS Plugin – Nice to have
- ⇒ Exercise templates are available @ https://github.com/karthibg/spring-workshop-templates
- ⇒ Exercise solutions are available @ https://github.com/karthibg/spring-workshop

PARTICIPANTS

Software-Developer and -Architects, who want to gain better understanding of the most popular spring framework.

Contact

1. **Michael Inden**

   CTO & Teamleiter SW-Entwicklung & Leiter ASMIQ Academy

   ASMIQ AG, Thurgauer Str. 39, 8050 Zürich
   E-Mail: michael.inden@asmiq.ch
   Kursangebot: https://asmiq.ch/

2. Bollu Ganesh Karthikeyan

   Lead Engineer / Trainer

   ASMIQ AG, Thurgauer Str. 39, 8050 Zürich
   E-Mail: karthikeyan.bolluganesh@asmiq.ch
   Kursangebot: https://asmiq.ch/

Spring Framework I Exercises

1. Explore template project

Import the maven project **"project-template"** into your IDE and answer the following questions:

a. This project has a transitive dependency on spring-core. True / False

b. spring-context contains the implementations of the IoC container. True / False

[Hint: Look into the pom.xml and maven dependencies]

2. Greeting App – Warm up

a. Create a **GreetingService** class with a method **sayHello**() to print the following sentence "Welcome to HBT :-) " [Without using Annotations]

b. Create a **GreetingApp** class with the `main` method and use the GreetingService and call the sayHello() method. Run the app and verify the output.

c. Now, the instantiation of the GreetingService directly with "new" keyword is forbidden. How would you instantiate them? Run the app and check that it prints out the same output. [Hint: Use ClassPathXmlApplicationContext]

[Hint: Create spring's IoC configuration metadata for this GreetingService class. Use the below configuration as the template]

```
<?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">

<!--TODO -->

</beans>
```

d. The above app simply prints the hardcoded "Welcome to HBT :-) ", this is a good start. However it would be better if it's configurable.

[Hint: use constructor injection.]

3. ASMIQ ACADEMY APP – INITIAL SETUP

We have received a requirement for the Asmiq Academy to create an application that enables to place an order for the courses for a customer. In Asmiq Academy we have roughly designed the classes and it is shown below Figure 1. The AsmiqAcademyApp should do the following:

- Use the top-level AsmiqAcademyService to retrieve the list of courses, (which design pattern?)
- Filter for a particular course and place an order for a customer
- Sms have to be sent after placement of an order

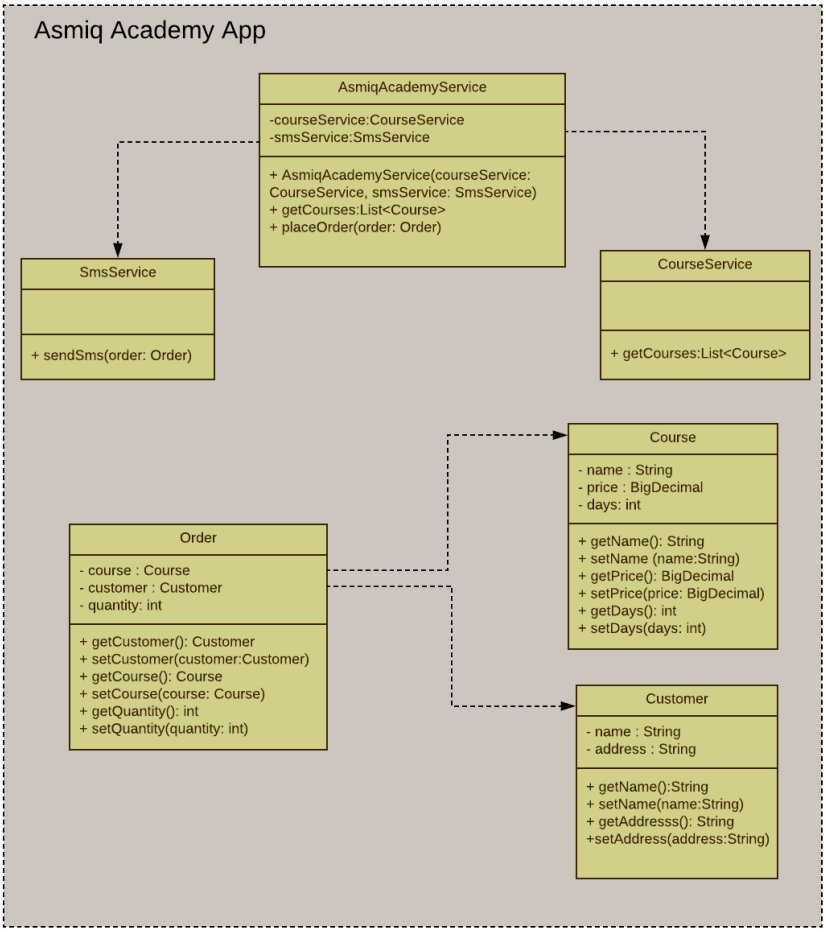


FIGURE 1

a. Implement the AsmiqAcademyApp by using the Spring IoC wherever possible through

I. XML configuration with autowiring

II. Java configuration with autowiring

[Hint: use the project template "**3-asmiq-academy-app**"]

4. ASMIQ ACADEMY APP – INTERFACE EXTRACTION

Upon reviewing the class design, our team lead mentioned that the AsmiqAcademyService should depend on abstractions and not on concrete implementations. Hence we came up with the following two new interfaces as shown in Figure 2.

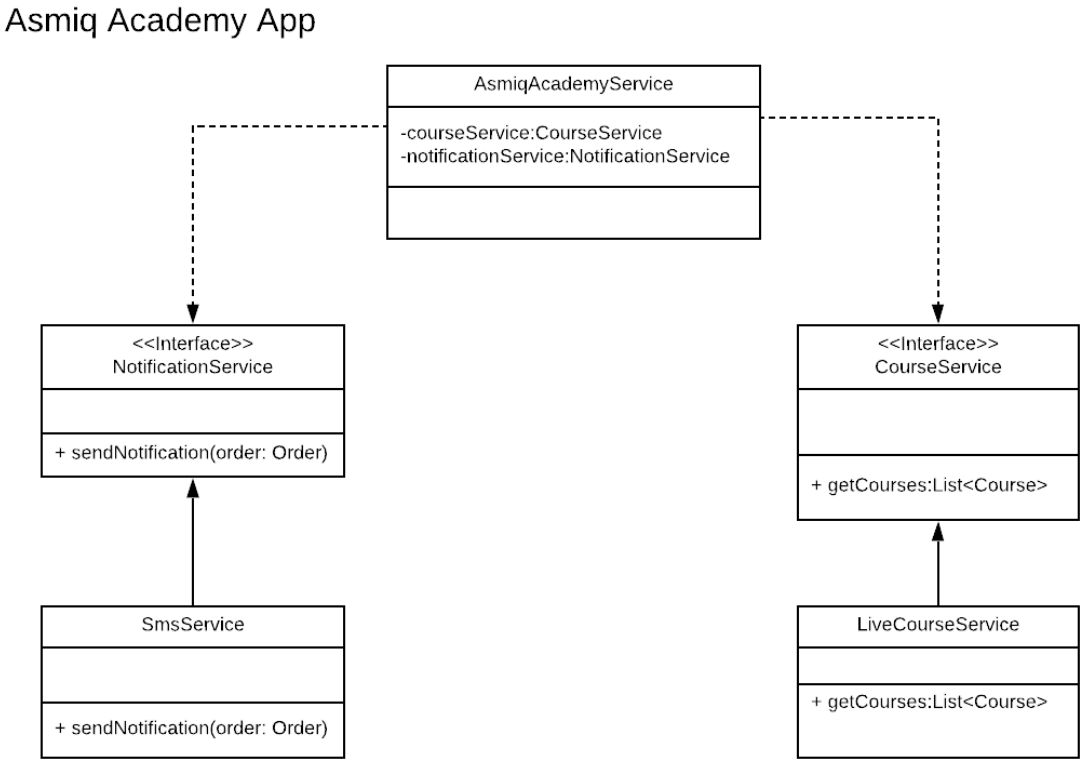


FIGURE 2

Exercise: As a result of the extraction of the two new interfaces, modify our AsmiqAcademyService App with these changes and do DI/IoC with Java configuration

5. ASMIQ ACADEMY APP – BEAN RESOLUTION STRATEGIES

As we have started receiving enquiries/registration from the customers outside switzerland, we have to notify their bookings through E-Mail. Besides some customers requested us to provide online courses. Hence we have to extend the design (Figure 3) to accomodate these new requirements.

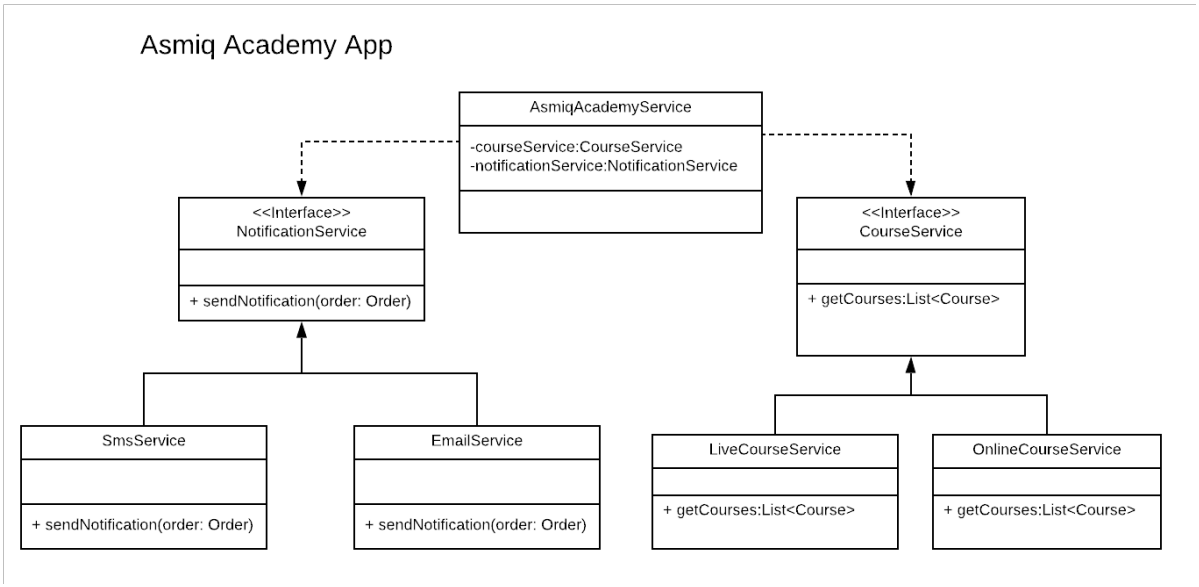


FIGURE 3

a. Extend our AsmiqAcademy App with two new services, run the app and check for the result.

[Hint: org.springframework.beans.factory.NoUniqueBeanDefinitionException: No qualifying bean of type 'ch.karthi.asmiq.NotificationService' available: expected single matching bean but found 2: emailService,smsService]

b. Solve the issue with NoUniqueBeanDefinitionException

[Hint: try with @Primary, @Qualifier, autowire byName, @Bean@Primary]

6. ASMIQ ACADEMY APP – HANDLING NON-MANDATORY / MANDATORY DEPENDENCIES

We are planning to enhance our app to include services like payment,feedback,exam and certification service as shown below Figure 4. However, at the moment we have agreed with SixPayment for payment, SurveyMonkey for feedback and Prometric for examination and we are still in search for certification providers.

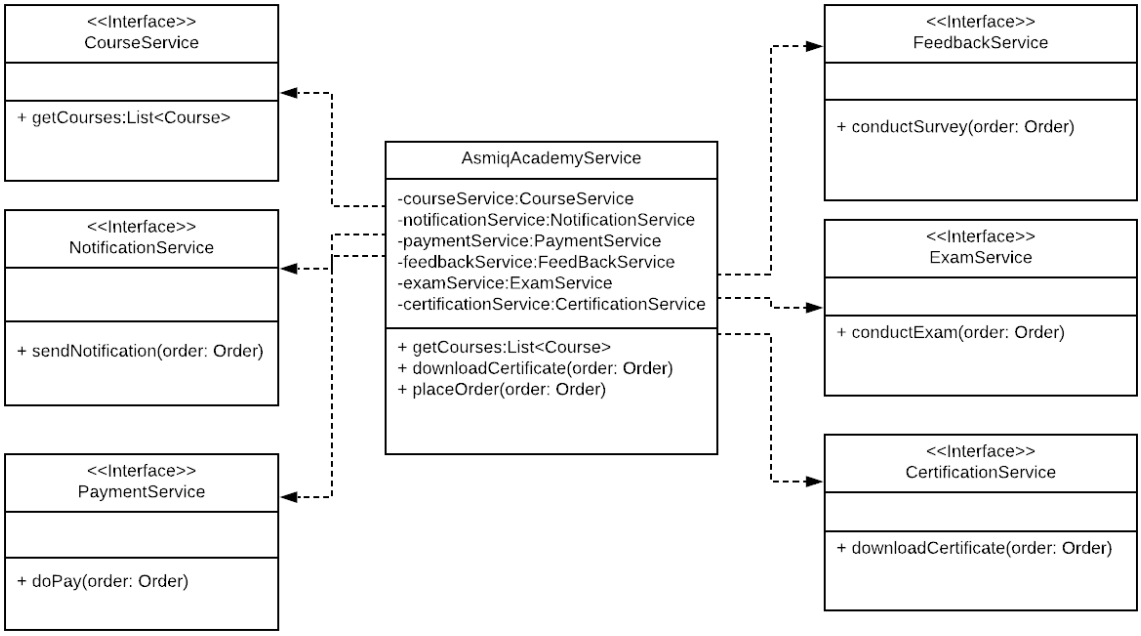


FIGURE 4

a. Enhance the AsmiqAcademyService as shown in the diagram by taking into consideration that when the AsmiqAcademyService is instantiated by the container all the services have to be injected **except the CertificationService (non-mandatory**). Constructor injection is not applicable here, hence find a better solution.

[Hint: use the maven project "6-asmiq-academy-app" as the template]

7. ASMIQ ACADEMY APP – CONFIGURATION PROPERTY EXTERNALIZATION.

The Implementation of the SixPaymentService is shown below. It shows that the value for discountPercent has been hardcoded in the source-code itself.

```
@Component
public class SixPaymentService implements PaymentService {

	private BigDecimal discountPercent = new BigDecimal("0.25");

	@Override
	public void doPay(Order order) {
		BigDecimal coursePrice = order.getCourse().getPrice();
		BigDecimal discountPrice = coursePrice.multiply(discountPercent);
		BigDecimal totalPrice = coursePrice.subtract(discountPrice);
		//log the order details with discounts
	}

}
```

a. Modify the SixPaymentService class to externalise the discountPercent through some property file (forexample: sixPaymentService.properties) that is placed in the classpath.

[Hint: Use " 7-asmiq-academy-app" as the template]

8. ASMIQ ACADEMY APP – DEPENDECY INJECTION OF THIRD-PARTY NON-SPRING COMPONENTS.

Customers requested us to provide them some possibility to validate the certificates issued by third-party providers. Hence, we tried to provide a service called CertificateValidationService. However, due to time constraints, we decided to integrate an existing validation service provider (Karthi-CertValidator), which is available in the market and validate the certificates.

```
@Component
public class AsmiqCertificationService implements CertificationService {

	private KarthiCertificationValidator karthiCertificationValidator;

	public void downloadCertificate(Order order) {

		if(karthiCertificationValidator.isValid()) {
			LOG.info("Certificate Downloaded!!");
		} else {
			throw new RuntimeException("Invalid Certificate");
		}

	}
```

a. Implement the CertificateService as shown above.

   [Note: Use the project "8-asmiq-academy-app" as the template]

b. Run the AsmiqAcademyApp and check for the output. What went wrong ? and Why ?

c. Fix the problem until the "Certificate Downloaded" appears in the console.

Spring Framework II Exercises

9. ASMIQ ACADEMY APP – PRE INIT CONSISTENCY CHECKS

The SixPaymentService wants to make sure that the discount percent should not be more than 75%. If it is set to more than 75% the entire App should not start up.

a. Modify the SixPaymentService to accomodate this requirement.

[Hint: Use bean lifecycle callbacks

If needed, add the following dependecy

```
<dependency>
        <groupId>javax.annotation</groupId>
        <artifactId>javax.annotation-api</artifactId>
        <version>1.3.2</version>
</dependency>
```

]

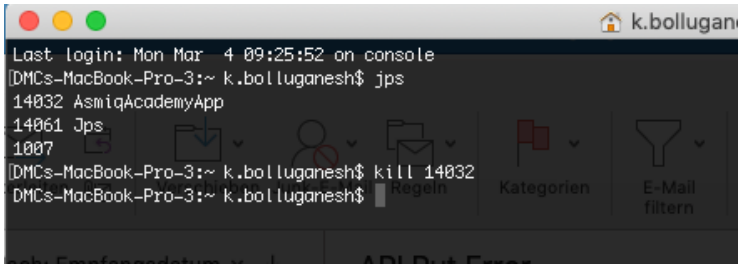
10. ASMIQ ACADEMY APP – GRACEFUL SHUTDOWN FOR CLEANUP/RESOURCE-HANDLING

It is our policy to gracefully shutdown our apps if the JVM process has been terminated intentionally/unintentionally.

a. modify our AsmiqAcademyApp to satisify this requirement.

[Hint: a. registerShutdownHook(); b. gracefully shutdown after 30 Seconds]

b. Run the app and kill the app's process through the kill <process_id> as shown below and check the console log.



11. ASMIQ ACADEMY APP – UNIT TESTING

-TODO in Spring course part II -

Spring Framework III Exercises

12. Explore Template Project

Import the maven project "13-web-projecttemplate" into your IDE and answer the following questions:

a) Run the class WebTemplateApp and fix the below exception.

Caused by: java.lang.IllegalArgumentException: port out of range:-1
at java.base/java.net.InetSocketAddress.checkPort(InetSocketAddress.java:143)
at java.base/java.net.InetSocketAddress.<init>(InetSocketAddress.java:188)
at org.apache.tomcat.util.net.NioEndpoint.initServerSocket(NioEndpoint.java:235)
at org.apache.tomcat.util.net.NioEndpoint.bind(NioEndpoint.java:210)

b) What are the advantages of using the embedded tomcat?

13. Greeting Rest Service

Using the "web-projecttemplate" project, create a Greeting REST-Service to:

a. expose an REST endpoint "/hello" to return the string "Welcome HBT ☺"

b. ensure that tomcat listens on port 7777.

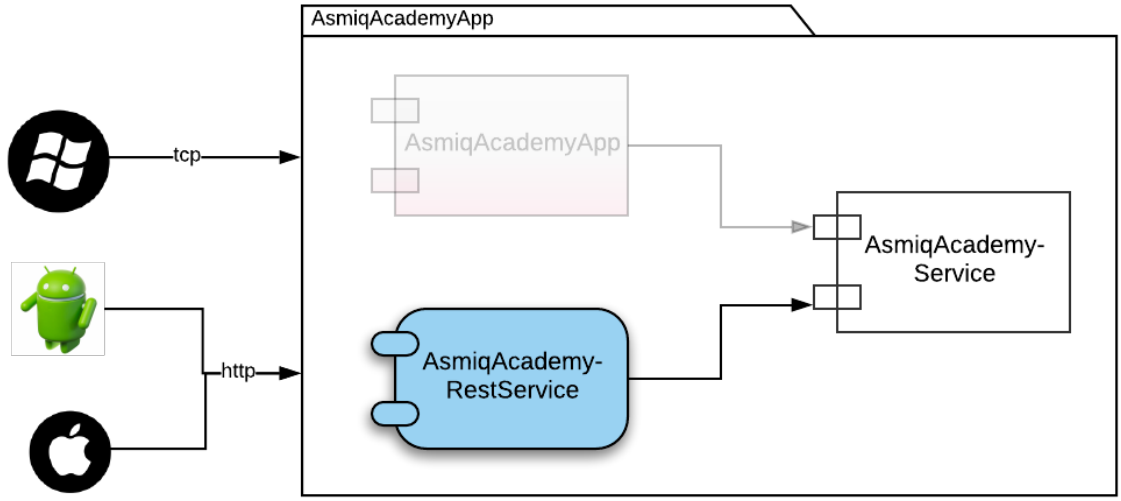
c. Run the app and call http://localhost:7777/hello and verify that the browser displays "Welcome HBT ☺" [Hint: use "&#x1f605" for smiley]

d. expose another REST endpoint "/hello/<name>" to return the string "Welcome" + <name>, <name> is any string, which would be entered by the user.

e. Run the app and call http://localhost:7777/hello/karthik and verify that the browser displays "Welcome karthik"

14. REST-SERVICE FOR ASMIQACADEMYAPP

In the last part of the exercise, we implemted a nice standalone AsmiqAcademyApp. Now, we have a new requirement from the customers to expose an REST endpoint "/courses" to retrieve the list of courses, so that they can consume this endpoint and list the courses in their GUIs/Apps.



a. Using the maven project " 15-asmiq-app-rest-service" do the following:

(1) Complete the TODOs in CourseController

(2) Run the App and execute http://localhost:8080/courses in the browser.

(3) Analyse and fix the following problem:

Type Exception Report

Message No converter found for return value of type: class java.util.ImmutableCollections$ListN

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

Exception

```
org.springframework.http.converter.HttpMessageNotWritableException: No converter found for return value of type: class java.util.ImmutableCollections$ListN
        org.springframework.web.servlet.mvc.method.annotation.AbstractMessageConverterMethodProcessor.writeWithMessageConverters(AbstractMessageConverterMethodProcessor.java:233)
        org.springframework.web.servlet.mvc.method.annotation.RequestResponseBodyMethodProcessor.handleReturnValue(RequestResponseBodyMethodProcessor.java:180)
        org.springframework.web.method.support.HandlerMethodReturnValueHandlerComposite.handleReturnValue(HandlerMethodReturnValueHandlerComposite.java:82)
        org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:119)
        org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:895)
        org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:800)
        org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:87)
        org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1038)
        org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:942)
        org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1005)
        org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:897)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:634)
        org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:882)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:741)
```

Note The full stack trace of the root cause is available in the server logs.

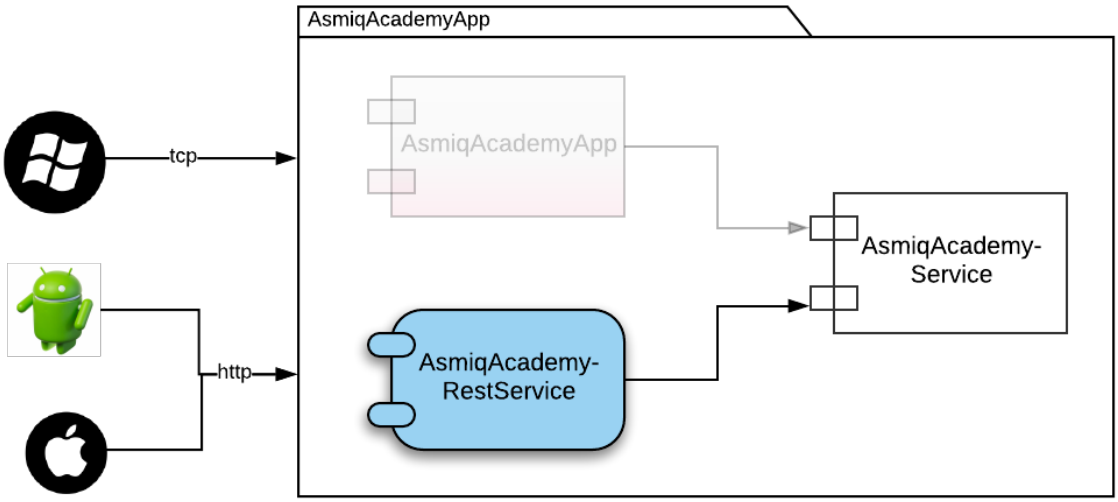
```
org.springframework.http.converter.HttpMessageNotWritableException: No converter
found for return value of type: class java.util.ImmutableCollections$ListN
```

[Hint: Check pom.xml and config class]

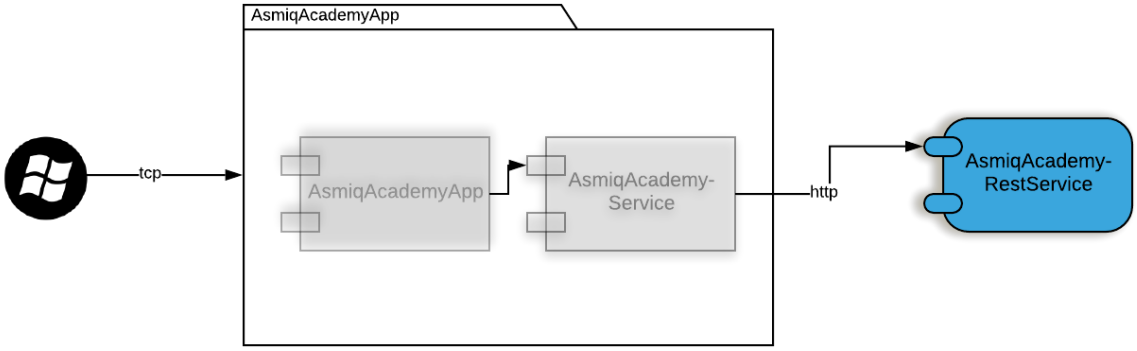
15. Discussion

Discuss the advantages and disadvantages of the integration of the AsmiqAcademyApp + AsmiqAcademyRestService approach A and approach B

Approach A



Approach B



Spring Boot I Exercises

16. Using Spring-Initializr

Using the Spring-Initializr (https://start.spring.io) create a spring boot app (java 11 version) called "**hbt-bootapp"** with the web-mvc dependency.

17. Greeting Rest Service – with spring boot

Using the **hbt-bootapp** complete the following:

a. Expose an endpoint **"/greeeting"** to return "hi HBT".

b. Expose an endpoint "/lotterynumbers" to return the next Saturdays's lottery number and this have to be secured with username "hbt" and password "password"

[Hint: add the spring-starter-security dependency]

c. Provide a welcome page, which redirects to the "/greeting" instead of the white label error page when we access http://localhost:8080

[Hint: Use the following snippet

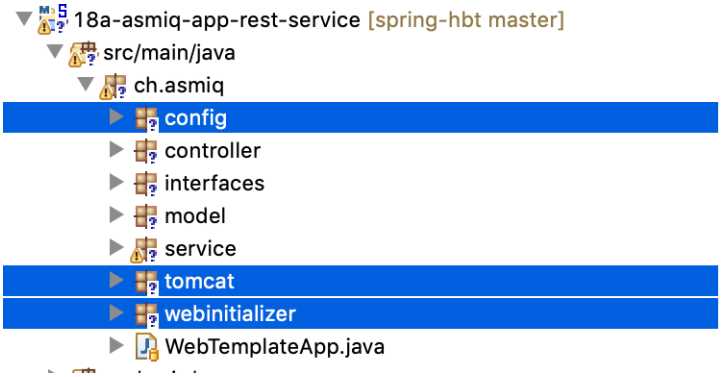
```
<!DOCTYPE html>
<html>
        <head>

                <meta http-equiv="refresh" content="0;URL='/greeting'" />
                <title>AsmiqAcademyApp</title>
        </head>
        <body>
                <b>Execute http://<host>:<port>/courses</b>
        </body>
</html>
```

18. MIGRATION OF ASMIQACADEMYAPP TO SPRING BOOT

In Question 15 we implemented a REST-Service for the AsmiqAcademyApp. Now migrate the entire AsmiqAcademyApp to Spring-Boot. To migrate to spring-boot do the following:

a. Use the maven project "18-asmiq-app-rest-service" [Note: This app listens again on port 8080]

b. Delete the highlighted packages in the above project. Why ?



c. Remove all the maven dependencies in pom.xml and add the "spring-boot-starter-web" as the only dependency. Does it sounds good ?

d. Complete the TODOs in the WebTemplateApp class

```
//TODO
public class WebTemplateApp {

	public static void main(String[] args) {
		//TODO
	}

}
```

e. Run the App and execute http://localhost:8080/courses and verify with the below result:

```
[{"name":"Java","price":50,"quantity":2},{"name":"Design
Patterns","price":60,"quantity":3},{"name":"Testing","price":40
,"quantity":1}]
```

f. One of customers required the courses response to be in XML instead of JSON. How would you do that ?

[Hint: Use Curl and accept header to test the xml/json response as shown below:

curl -v -H "Accept: application/json" http://localhost:8080/courses

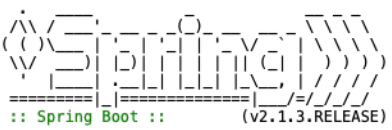
curl -v -H "Accept: application/xml" http://localhost:8080/courses]

19. TESTING

--TODO will be handled in Part II--

20. BANNER

change the default below shown Spring Boot banner



Like: "schönes WE", "auf zum Bier", "thank God it's Friday" , "Good bye ☺"

[Hint: use http://bit.ly/2T2ShFU or http://bit.ly/2HfReeo]