**Exercise 1: Getting Started with Hello World**

step1. Launching Java IDE eclipse.

step2. Download Tomcat server in version (8.5) (also try version 9.0) and unzip to folder.

step3. Create a Maven Project in maven-archetype-webapp with groupId being com.mitac and artifactId being pset.spring01.

step4. Meet JSP servelet wasn’t found error

step5. Deploy TomCat server to solve the problem.

step5-1. Setting JAVA\_HOME environment variable directing to your java VM where is the installed directory of your JDK.

step5-2. Connect IDE eclipse to your Tomcat: in Window -> Server -> Runtime Environment -> Add -> Choose TomCat Server version -> browse and found it -> Choose, Apply and Close.

step5-3. set project runtime environment. Right click your maven web project (pset.spring01) -> properties -> Java Build Path -> Libraries -> Add Library -> Server Runtime -> Choose Tomcat server what you just set to connect to eclipse. -> solve the JSP problem in step4.

step6. Create “package” under src following the problem set

step7. Create “ContextLoadTest” class under src/test/java and enter the code.

step8. Found lots of errors in it and solving it with including necessary jar files in pom.xml. Find the requested dependency xml with Google.

step9. According to Spring Test, Use @Autowired to get injection and get HelloWorld Instance.

step10. print out value of it.

Asking:

Q1: What is the function of @RunWith?

A1: It will use the class after the annotation as [Junit running environment](https://www.cnblogs.com/peak911/p/9293145.html) to test program. In this case it uses SpringJUnit4ClassRunner as [unit test](https://docs.spring.io/spring-batch/docs/current/reference/html/testing.html).

Q2: What is the function of @ContextConfiguration?

A2: It will dynamic generate an instance of ApplicationContext which is used to read and load Beans definition xml file. And usually it is operated with unit test in Spring. In default it will load the “Application.xml” in the same location, and you can assign a “locations” parameter to it to locate your xml file.

With your header in xml, the annotation could know the file and realize its form of beans.

Q3: What is the function of @Test, @After and @Before?

@Test is an annotation, defined under junit.test. This is used to test the “function” of the offered code working or not. In this case is to “print” the value.

@Before is an annotation, defined under junit.test. This used to do the pre-condition of the main tested code. Something like before operation on DB, we need to connect it first. Or login the page before we use it. In this case is to getBean before we test its print function.

@After is an annotation, defined under junit.text. This is used to do the past-result of the main tested code. Something like after operation on DB, we need to disconnect it. Or logout the page after we used it. In this case we don’t have any action on it. (Or we may try to destroy the Bean instance to release the memory?)

Q4: What is the function of @Autowired?

A4: It is an annotation to do a request to Spring IoC Container to get a Bean instance. Usually we use this annotation in unit test. Instead, we use ApplicationContext with method .getBean() to get Bean instance. (Later in Exercise3 we will meet)

Reference:

[JavaSpring Tutorial](https://www.tutorialspoint.com/spring/index.htm)

[Factory Pattern 工廠模式](https://blog.amowu.com/factory-pattern/)

[來點比薩吧 – 工廠方法與抽象工廠](https://blog.techbridge.cc/2017/05/22/factory-method-and-abstract-factory/)

[Spring AOP idea about Aspect……](https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/440077/)

[AOP idea and term](https://openhome.cc/Gossip/SpringGossip/AOPConcept.html)

[IoC DI 深入淺出](https://dotblogs.com.tw/daniel/2018/01/17/140435)

[IoC - SpringWebMVC](https://openhome.cc/Gossip/CodeData/JavaTutorial/SpringWebMVC.html)

[為什麼要使用interface?](https://medium.com/@jessie75919/%E5%B0%8F%E6%95%85%E4%BA%8B%E5%9C%96%E8%A7%A3%E7%89%A9%E4%BB%B6%E5%B0%8E%E5%90%91%E7%9A%84-interface-%E6%98%AF%E4%BB%80%E9%BA%BC-%E7%82%BA%E4%BB%80%E9%BA%BC%E9%9C%80%E8%A6%81-interface-%E4%BB%80%E9%BA%BC%E6%99%82%E5%80%99%E8%A6%81%E4%BD%BF%E7%94%A8-interface-b8a7405adf6c)

[JavaMaven的pom.xml是什麼?](https://matthung0807.blogspot.com/2018/08/java-mavenpomxml.html)

Exercise 2: Reading properties files in Spring

step1. Create hello.properties with content “hi=hi” under src/main/resource.

step2. Refer to [PropertyPlaceHolderConfigurer](https://docs.spring.io/spring-framework/docs/2.5.6/reference/beans.html#beans-factory-placeholderconfigurer).

step3. According to step2. Set PropertyPlaceHolderConfigurer Bean in ctx\_ut.xml.

step4. set the property of it in name=”location” and assign its value to be the location of hello.properties

step5. In property’s value of Bean of HelloWorld. Use the form “${\_key\_}” to match the value in properties file being: \_key\_=\_value\_.

Asking:

Q1: How did PropertyPlaceHolder work in brief?

A1: We set PropertyPlaceHolderConfigurer in xml file and the ApplicationContext will catch this, automated generate a Bean instance of it. Following the property location we assign to and getting the value back.

Reference:

[Official document](https://docs.spring.io/spring-framework/docs/2.5.6/reference/beans.html#beans-factory-placeholderconfigurer)

[OpenhomeCC: PropertyPlaceholderConfigurer](https://openhome.cc/Gossip/SpringGossip/PropertyPlaceholderConfigurer.html)

Exercise 3: Understand bean scope

step1. Use @Autowired to get Bean of ApplicationContext in ContextLoadTest.

step2. Use the method .getBean(\_id\_) of ApplicationContext to get HelloWorld Bean instance.

step3. Use method .setMessage(\_HelloWorld\_) of HelloWorld. Then use .getMessage() to print message.

step4. Set the Bean scope to “prototype” and do step2 and step3 again. Set different Message in step3 And we should got different value.

Asking:

Q1: Why there are different between step2 and step3?

A1: Due to the definition of the scope of Bean. Originally, scope is default setting as “singleton”, which means each time we use ApplicationContext to query IoC container to return an instance, the instance will refer to the same one address in memory, which means it’s the same instance.

And follow the exercise, we set the scope as “prototype”. This means each time we use ApplicationContext to query an instance, IoC container will create a new instance which refer to different memory address and return.

So actually we set and get message in step3 is refer to the same instance. And in step4 is refer to different instances. Of course, the first one should be the same value and the later should be different.

Reference:

[Official Bean Scope Document](https://www.tutorialspoint.com/spring/spring_bean_scopes.htm)

Exercise 4: Play with initial method

step1: create SeeYou class and bean definition following.

step2: add new property to hello.property.

step3: define initial method in Bean SeeYou

step4: change the scope to singleton.

Asking:

Q1: How do you arrive initial method?

A1: There are two ways, one is use class constructor, and the other is define init-method parameter in Beans xml file.

Constructor way is that we define a constructor in class file. When the IoC container create Bean, constructor will activate automate. If we would give parameter to constructor, use “constructor-arg.”

init-method is defined in Bean xml file. No any difference between the init-method and other method in class.

Q2: Have you ever known about destruction-method?

A2: Yes, I’ve read the Bean document and know something about it. Use destroy-method in Bean xml file to achieve it. (Note: Use context.close() to activate destroy-method)

Reference:

[Official Bean Definition Document](https://www.tutorialspoint.com/spring/spring_bean_definition.htm)

[Spring – 關於Bean的Init and destroy](https://www.itread01.com/p/328640.html)

[No default constructor found – Stack Overflow](https://stackoverflow.com/questions/25272543/no-default-constructor-found-nested-exception-is-java-lang-nosuchmethodexceptio)

Exercise 5: depends-on and initial priority

step1. Create Goodbye class and define Bean in ctx\_ut.xml.

step2. Add property to hello.properties.

step3. Run unit test and see the result is different or not.

Asking:

Q1. What is the function of “depends-on”?

A1. To force the order of the Bean generation according to depends-on. If you use “depends-on=classA” in classB. Then the Bean of classA must create before Bean of classB, vice versa.

In real application, something like login, verify or permission will use this to ensure the Bean is generated in right order.

Q2. Are the message of HelloWorld and Goodbye the same?

A2. Due to the scope “singleton,” they will be the same.

Q3. How about after depends-on?

A3.Becase SeeYou is depends-on Goodbye, Bean of SeeYou will generate after Bean of Goodbye. So they will be different that HelloWorld will be the same to SeeYou.

Reference:

[Official Bean Annotation Document](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/DependsOn.html)

[Spring - @DependsOn](https://blog.csdn.net/andy_zhang2007/article/details/78859949)

[ref vs depends-on attributes in Spring – Stack Overflow](https://stackoverflow.com/questions/12663491/ref-vs-depends-on-attributes-in-spring)

Exercise 6: Start up web with spring

step1. Refer to document.

step2. Register WebApplicationContext in JSP file to let scriptlet could use ApplicationContext in JSP file.

step3. Import all classes in order to use them.

step4. Add filter and servlet to web.xml

step5. Complete

step6. Because web.xml MUST BE under WEB-INF folder (According to StackOverflow and wiki), so we could leave it alone in the folder.

step7. Change the contextConfigLocation param-value in web.xml to classpath:ctx\_ut.xml in order to assign the location. And also, in ctx\_ut.xml change the PropertyPlaceHolderConfigurer location value to classpath:hello.properties.

These setting could make all xml, properties or any inside data to be resources folder so that we could have much more maintainable and readable in the structure of the project folder.

Asking:

Q1: How did you activate Tomcat server?

A1: According to the exercise1.

Q2: What is filter and servlet?

A2: In my opinion, these meet the simple MVC structure although there is SpringMVC framework already (different, but met the spirit). I think Filter (Controller) will catch Http Request before it arrives at servlet. Filter will allocate different servlet (also Controller) to deal with different Http Request. And servlet will access Bean file (Model). Finally, servlet will return response (View) by Filter to its source request.

(Note: the difference between filter and servlet is that filter didn’t generate response but only transmit. Something like preprocess.)

Reference:

[SpringMVC框架教學](http://tw.gitbook.net/spring/spring_web_mvc_framework.html)

[什麼是MVC?好吃嗎?](https://ithelp.ithome.com.tw/articles/10191216)

[第一個SpringMVC](https://openhome.cc/Gossip/SpringGossip/FirstSpringMVC.html)

[SpringMVC Framework using filter](https://dotblogs.com.tw/Johnson_note/2018/11/07/005416)

[透過filter處理http請求](https://medium.com/chikuwa-tech-study/spring-boot-%E7%AC%AC15%E8%AA%B2-%E9%80%8F%E9%81%8Efilter%E6%93%B7%E5%8F%96http%E8%AB%8B%E6%B1%82%E8%88%87%E5%9B%9E%E6%87%89-d4a4d353433f)

[Servlet,Filter,Listener,Interceptor的作用和區別](https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/12583/)

[關於classpath和contextConfigLocation](https://blog.csdn.net/yongh701/article/details/78548506)