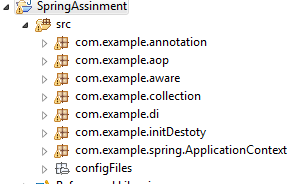
**Building First application**

1. Create a folder called configFiles under SpringAssignment/ src folder.



2. create a file named **beans.xml** in configFiles

***Add following lines into beans.xml***

<?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***

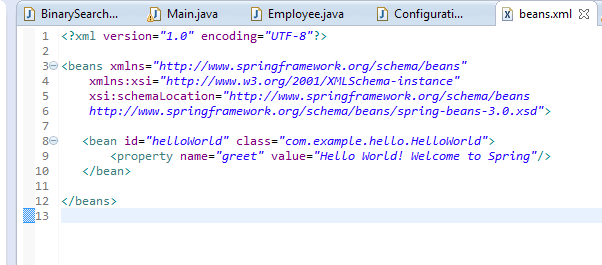
***http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">***

***<bean id="helloWorld" class="com.example.hello.HelloWorld">***

***<property name="greet" value="Hello World! Welcome to Spring"/>***

***</bean>***

***</beans>***



1. Create a package called com.example.hello under

SpringAssignment\src

5. create a java file HelloWorld.java under the package

com.example.hello

6. Write following in HelloWorld

***package com.example;***

***import org.springframework.context.ApplicationContext;***

***import org.springframework.context.support.ClassPathXmlApplicationContext;***

***public class HelloWorld {***

***private String greet;***

***public String getGreet() {***

***return greet;***

***}***

***public void setGreet(String greet) {***

***this.greet = greet;***

***}***

***public static void main(String[] args) {***

***ApplicationContext ctx = new ClassPathXmlApplicationContext("configFiles/beans.xml");***

***HelloWorld bean =(HelloWorld) ctx.getBean("helloWorld");***

***System.out.println(bean.getGreet());***

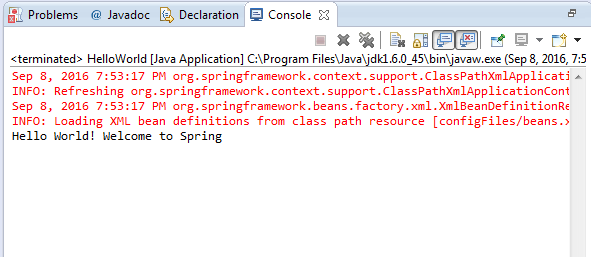
***}***

***}***



15. Run as java application output will be

Hello World! Welcome to Spring



**Understanding the Application**

Before understanding the application code, we try to understand How spring works?

1. When you start an application by invoking java main, Spring creates a Container and try to search Spring configuration file which is “beans.xml” in this case. Spring configuration file does not have a predefined name, you can name whatever you wish but you have to pass the Path.

2. By the given path container load the XML file.

3. Container parses the XML files and reads the Spring bean register in that file.

4. Now Container tries to instantiate those beans using java reflection by the fully qualified class name given in configuration file.

5. Then Bean life cycle phases are start. We will discuss it later.

6. Container tries to wire beans among themselves. The Dependency injection did here.

7. Spring container register beans information in the internal map where bean id is the Key of the map.

8. Now Beans are ready to use.

So, Spring Configuration files play a pivot role in Spring application, Spring container reads all instructions from that file. So if anything goes wrong in that file (say forgot to close a tag) or File path is wrong spring application will not load and it throws exception

***TIP: Always pay close attention to configuration file***

**Diagram:**

**No we are in a position to understand code,**

**beans.xml:**

Here we register our java class by bean tag. Please note we put a property id in bean tag. This is a unique id. No two beans should have the same Id, else Spring container will complain and throw an exception. id is an important property, always provides an id for each bean, later by this id we can get an instance of Spring bean from Spring container.

Next, we have written a property tag under bean tag where name attribute must be same as java attribute name (POJO attribute). Then we pass a value which will actually inject in the HelloWorld bean.

In the main method, we spawn the Spring container by the following statement

**ApplicationContext ctx = new ClassPathXmlApplicationContext("configFiles/beans.xml");**

We use ***ClassPathApplicationContext*** which will search configuration file in the classpath. If it is not found spring will throw an exception.

In next line, we try to get the HelloWorld bean by id, which was stated in the configuration file.

After that, we invoke getGreet() method which shows the value which was passed from the configuration file.

Here, Internally Spring injects the value into the java(POJO) property via a Setter method. So you have to provide setter method in HelloWorld unless Spring can’t inject the value and eventually throw an exception.