#### [Spring Core - Inversion of Control](javascript:void(0);)

**Objectives**

* Demonstrate dependency injection based on spring configuration file
  + IoC container, container injects dependent beans as well on creation of a bean, ref attribute
    - Dependency Injection - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-dependencies
* Demonstrate dependency injection based on annotations and autowiring
  + Load bean using getBean() specifying the bean name, @Autowired in setter method, @Component
    - Autowired - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-autowired-annotation
    - Component - https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-stereotype-annotations

**Spring Core – Implement data model for Employee using Spring XML Configuration**   
  
Implement data model for employee based on instructions provided below: 

* Create package com.fis.spring-learn.bean
* Create class com.fis.spring-learn.bean.Employee with following attributes

|  |  |
| --- | --- |
| **Type** | **Attribute Name** |
| int | id |
| String | name |
| double | salary |
| boolean | permanent |
| Date | dateOfBirth |

* Generate getters, setters and toString() methods.
* Include empty parameter constructor with debug log.
* Define values for the attributes in a new spring xml configuration file named employee.xml
* Display the employee details in a new method displayEmployee() in main method of SpringLearnApplication.java

**Spring Core - Implement data model for Department in Employee**   
  
In continuation to the previous problem, let us add department details to Employee. Refer instructions below. 

* Create com.fis.spring-learn.bean.Department
* Add attributes id and name
* Include empty parameter constructor with log
* Generate getters, setters and toString()
* Include department as an attribute in Employee class
* Include getters/setters for department
* Regenerate toString() method in Employee for inclusion of department
* In employee.xml, using ref attribute include department to employee. Refer sample spring xml code below.

   <bean id="departmentBean" class="com.fis.bean.Department">

        <property name="id" value="1" />

        <property name="name" value="Payroll" />

    </bean>

    <bean id="employee" class="com.fis.bean.Employee">

  ... other attributes of employee goes here

        <property name="department" ref="departmentBean" />

    </bean>

**Spring Core - Include Skill details for Employee**   
  
In continuation to the previous problem, incorporate list of skills for each Employee instance:

* Create new class com.fis.spring-learn.bean.Skill
* Create instance variables id and name with data types int and String respectively.
* Include, empty parameter constructor with log, getters, setters and toString()
* Add skills as instance variable of type Skill array in Employee.
* In Employee class, include getters, setters and update the toString() method for inclusion of Skill[]
* Create multiple skill instances in employee.xml and add them to array of Skills.
* Set the array of Skills to skills of Employee bean in configuration xml file
* Load the employee class from employee.xml and display the data.
* Refer sample code below inclusion of skills property for employee in employee.xml

       <property name="skills">

            <array>

                <ref bean="skillHtml" />

                <ref bean="skillCss" />

                <ref bean="skillJs" />

            </array>

        </property>

**Spring Core - Inversion of Control / Dependency Injection**   
  
In order to build a web service for managing employee details, controller, service and dao classes needs to be defined.  
  
As part of this hands on, creation of these classes needs to be done using Spring Core framework's Inversion of Control (IoC) and Dependency Injection.  
  
At the same time we need to implement these using annotations without spring configuration xml file. Refer steps below to gradually implement this. 

* Create all necessary classes based on the below specified class diagram. Do not include any constructor, toString() and getter methods.
* Include logs in constructors and setter methods on all these classes.
* Implement dependency injection in a staged manner as specified below one by one:
  + **Spring XML Configuration:**
    - In employee.xml define the beans for EmployeeController, EmployeeService and EmployeeDao using bean reference.
    - Load the EmployeeController bean in a new method getEmployeeController() in the SpringLearnApplication.java
    - Check the logs to see if EmployeeController, EmployeeService and EmployeeDao are loaded
  + **Spring XML Configuration with Autowiring:**
    - Include new bean configuration in employee.xml for the three classes using autowire property.
    - Modify getEmployeeController() method to load the new autowired bean from xml configuration.
    - Check the logs to see if all three classes are loaded.
  + **Loading beans using Annotations:**
    - Include @Component annotation at class level for EmployeeController, EmployeeService and EmployeeDao
    - Include @Autowired annotation in the respective setter method of EmployeeController and EmployeeService
    - Include a new method in SpringLearnApplication.java with below signature:

public static void displayEmployeeControllerAnnotation(ApplicationContext applicationContext)

* Use the below code in this method to load the controller bean:

        EmployeeController employeeController = (EmployeeController) applicationContext

                .getBean("employeeController");

* Invoke displayEmployeeControllerAnnotation() method in main method passing the applicaitonContext.

|  |
| --- |
|  |

Explain how Inversion of Control works.