Hibernate tool
Spring Framework
      a. Spring Core(Core java till collections)
      b. Spring DataJPA(CoreJava + JDBC + Hibernate)
      c. Spring MVC(CoreJava + Servlet + JDBC)

Spring
  It is a framework which can be used to develop applications in less line and faster way(RAD)
  Rapid Application Development, by using inbuilt technologies and Design patterns.
      a. Spring is a Framework
      b. Compare to other techology and framework, Spring is faster in coding and execution.
      c. Spring internally has few Design patterns like Template,Factory, Proxy, MVC, FrontController etc.
      d. Spring Supports End-End Application Development.
      e. It support 4 layers of Coding
            a. Presentation Layer
                  Contains view logic which is show to the end user
                        eg: JSP, Thymleaf, Velocit, Freemarker.
            b. Service Layer
                  Contains buisness logic like calculation,validation,....
                        eg: Spring AOP,SpringTransaction,....
            c. DataAccess Layer
                  This layer is used to perform persistence operation like save,update,delete,select.
                        eg: SpringJDBC,SpringORM,SpringDataJPA,.....
            c. Integration Layer
                  This Layer is used to link difference applications like EmailService, JMS, WebServices,.....
                        eg: SpringReST

++++++++++
SpringCore
++++++++++
 => In this module of Spring, we learn about rules and regulations to work with "Spring-Container".
 => SpringContainer takes care of
      a. Creating an object.
      b. Providing data to object.
      c. Linking object to another object.
      d. Destroy the object.

 => Spring Container needs two inputs from the programmer.
      a. Spring Bean(Java class)
      b. Spring Configuration file(XML/Java/Annotations)

By taking these information, container will create an object with the data.
That object programmer can read and use it for some testing puropose.

++++++++++
SpringBean
++++++++++
  => It is a class given by programmer which follows the rules of Springcontainer
  => If we follow the rules and writes the class, then container will accept our class and creates an object to it, else
     container won't create the object.

SpringBean rules
++++++++++++++++
      a. class must have package statement.
      b. class must be public type.
      c. class can have a variable,if exists it should be private.
      d. class must have default constructor with setXXXX/getXXXX() for every variable.
      e. class can override Object class methods like toString(),hashCode(),equals()
      f. class can have annotations which are defined inside SpringAPI and alos it can have core java annotations
            like @Overide,....
      g. class can implement only SpringAPI interfaces and only one special interface is allowed(java.io.Serializable).

++++++++++++++++++++
Dependancy Injection
++++++++++++++++++++
  It is a theory concept followed by SpringFramework,this concept is used by spring container to create the objects and providing
  data to the variables.

Dependancy
  it is a variable defined in the class(SpringBean),Based on the datatype used to create the variable we have 3 types
      a. Primitive Type
      b. Collection Type
      c. Reference Type

Primitive Type Dependancy[PTD]
      a. If a variable is created using one of the below data type then in PTD/PT
            datatypes are : byte,short,int,char,long,float,double,boolean,String.

Collection Type Dependancy[CTD]
      a. If a variable is created using List(I),Set(I), Map(I),Properties(P) then it is called as "CTD/CD"
      b. All these interfaces are from java.util package.

Reference Type Dependancy[RTD]
      a. It refers to HAS-A relationship.
      b. Relationship can be between
            1. interface - class
            2. class - class

eg#1.
```
interface A{}
interface B{}
interface C{
      int a; //PTD
      String b;//PTD
      List c;//CTD
      Map d;//CTD
      A obj1;//RTD
      B obj2;//RTD
}
```

Injection(I)/Dependancy Injection(DI)

++++++++++++++++++++++++++++++++++
 Injection means "Provide data to variables(Dependancy)".
    It is of 4 types
      a. Setter injection
      b. Constructor injection
      c. LookUpMethod injection(comes during scope concept)
      d. Interface Injection(not supported by Spring Framework)

Softwares requried
      a. https://spring.io/tools(springtool suite)
      b. JDK Software
      c. Spring jars

ConfigurationFile
      a. XML
      b. Java
      c. Annotation

Note:
      a. After creating the configuration file and bean file, we need to start the container through Test class.

Test class
 a. This class is used to test our code,"Is this container created the object with the required data or not".
 b. Container is pre-defined code of Spring Framework(IOC).
 c. Container name is "BeanFactory(I)[Old container] and ApplicationContext(I)[New Container]".
            BeanFactory(I)
                a. XmlBeanFactory(C)
            ApplicationContext(I)
                a. ClassPathXmlApplicationContext(C)
                b. FileSystemXmlApplicationContext(C)
                c. AnnotationConfigWebApplicationContext(C)
                d. AnnotationConfigApplicationContext(C)

XML
  applicationContext.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.xsd">

      <!-- bean definitions here -->
      <bean id="student" class="in.pwskills.bean.Student">
            <property name="sid" value="10" />
            <property name="sname" value="sachin" />
            <property name="saddress" value="IND" />
            <property name="sage" value="49" />
      </bean>
</beans>
```


Java Configuration
++++++++++++
AppConfig.java

```
++++++++++++++
package in.pwskills.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import in.pwskills.bean.Student;

@Configuration
public class AppConfig {

	static {
		System.out.println("AppConfig.class file is loading...");
	}

	@Bean
	public Student studObj() {
		System.out.println("AppConfig.studObj()");

		Student student = new Student();
		student.setSid(7);
		student.setSname("dhoni");
		student.setSaddress("CSK");
		student.setSage(41);

		return student;
	}
}

++++++++++++++++++++++++++
Working with Collection Type
++++++++++++++++++++++++++
package in.pwskills.config;

import java.util.HashMap;
import java.util.LinkedHashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.Set;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import in.pwskills.bean.Product;

@Configuration
public class AppConfig {

	static {
		System.out.println("AppConfig.class file is loading...");
	}

	@Bean
	public Product getProdObj() {
		System.out.println("AppConfig.getProdObj()");

		Product product = new Product();
```

```
		product.setData(list());
		product.setModels(set());
		product.setModes(map());
		product.setContext(props());

		return product;
	}

	private Properties props() {
		Properties properties = new Properties();
		properties.put("GRM", "Fossil");
		properties.put("CHINA", "Tissot");
		properties.put("USA", "Omegha");
		return properties;
	}

	private Map<Integer, String> map() {
		Map<Integer, String> hm = new HashMap<>();
		hm.put(10000, "fossil");
		hm.put(20000, "tissot");
		hm.put(30000, "omegha");
		return hm;
	}

	private Set<String> set() {
		Set<String> hs = new LinkedHashSet<>();
		hs.add("chronography");
		hs.add("digital");
		hs.add("analog");
		return hs;
	}

	private List<String> list() {
		LinkedList<String> l1 = new LinkedList<>();
		l1.add("fossil");
		l1.add("tissot");
		l1.add("omegha");
		return l1;
	}

}
```

Product.java

```
++++++++++++
package in.pwskills.bean;

import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.Set;

public class Product {

	static {
		System.out.println("Product.class file is loading...");
	}

	//Collection Type
```

```
	private List<String> data;
	private Set<String> models;
	private Map<Integer, String> modes;
	private Properties context;

	public Product() {
		System.out.println("Product Constuctor used by SpringFramework...");
	}

	public List<String> getData() {
		return data;
	}

	public void setData(List<String> data) {
		this.data = data;
		System.out.println("Product.setData():: List<String> ");
	}

	public Set<String> getModels() {
		return models;
	}

	public void setModels(Set<String> models) {
		this.models = models;
		System.out.println("Product.setModels() :: Set<String> ");
	}

	public Map<Integer, String> getModes() {
		return modes;
	}

	public void setModes(Map<Integer, String> modes) {
		this.modes = modes;
		System.out.println("Product.setModes() :: Map<Integer,String>");
	}

	public Properties getContext() {
		return context;
	}

	public void setContext(Properties context) {
		this.context = context;
		System.out.println("Product.setContext():: Properties");
	}

	@Override
	public String toString() {
		return "Product [data=" + data + ", models=" + models + ", modes=" +
modes + ", context=" + context + "]";
	}

}
```