Week 7 Agenda (Spring & Spring Boot Framework)

* Spring Framework (POJO/ IoC/DI)
* Spring AOP (Aspect Oriented Programming)
* Spring Web MVC (Creating MVC using Spring Web Framework)
* Spring Boot ( Creating Rest Web Services Using Spring Boot),
* Spring Actuators, API Documentations (OAS – Open Api Specification), Spring Data JPA

Week 6 Revisit ( Adv JAVA & JPA – EclipseLink & Hibernate)

* Functional Interface, Lambda, Optional Class
* Multi Threading, Thread Class & Runnable Interface, Thread States, Priority, Live/Dead Lock, Synchronization, Logging, Logging-Level, Logging Threshold
* JPA ( Java Persistence API) & ORM (Object Relational Mapping)
* JPA Providers (JPA Implementation) [Hibernate & EclipseLink]
* Factory Design Pattern [Entity/Session Manager]
* HQL, JPQL, ACID Properties, Criteria, Caching (First & Second Level of Caching)

If a operation takes long time to complete or takes more memory to get the result, it is called as costlier operation.

In Hibernate two levels of Caching available

First Level Caching is default in Hibernate, Can’t be disabled and shared by session. L1 caching is internal to session.

Second Level Caching (L2) is enabled separately. It’s shared by session. This caching is also valid till the session object is alive.

<property name="cache.use\_second\_level\_cache">true</property>

<property name="hibernate.cache.region.factory\_class">org.hibernate.cache.ehcache.EhCacheRegionFactory</property>

@Cacheable

@Cache(usage = CacheConcurrencyStrategy.READ\_ONLY)

@Entity

@Table(name="student")

public class Student

{

//code goes here...

}

DB Related Operations are costly operations.

User Input is also a costlier operation

Week7

Day 1

Spring Framework – Spring is a Java Based Framework.

* Used to create Enterprise Java Applications easily.
* Challenges faced by java EE developers are resolved in this framework
* Configuration (XML based)
* Stand-alone application (Core JAVA application) Adv-Java Applications (Web based – Servlet/JSP/JSF)
* JAR packaging – Stand-alone Java Application (Java Archive) pom.xml[optional]
* WAR Packaging – Adv-Java/Web Applications (Web Archive) – web.xml &/ pom.xml (Both are optional)
* EAR Packaging – Enterprise Java Application (Enterprise Archive) web.xml, pom.xml, persistence.xml ……
* XML is case and space sensitive (We need to restart the server when ever we update the xml configuration file)
* Creating and Configuring EJB (Enterprise Java Beans) is a challenging task.
* Mail configuration, messages, JNDI, JMS, (Adv Java)
* Spring framework is a powerful Java based framework which will simply use the POJO class to create enterprise application.

Spring uses two design pattern to achieve this.

1. IoC (Inversion of Control)
2. DI (Dependency Injection)

You are travelling to office everyday. (IoC Example)

1. You are taking your own car to office every day (You are the one & only controller here)
   1. You can decide the start time
   2. You can decide the route
   3. You can take n number of stops in between
   4. You can go in a route for a particular day
   5. You have the control to fill the fuel
2. You are booking ola/uber can to office everyday ( Control will be transferred to the cab driver)

IoC is a Design Pattern which can be applied in any programming language.

Spring Framework used this design pattern.

Spring Framework will automatically create the objects when ever required.

For Ex:

Employee emp = new Employee(); //Creating an object in a traditional way. [Tightly coupled code]

Employee emp1; // creating only the reference not actual object. [Loosely coupled code]

emp1 = (IoC container) Employee(); //Injecting the Object at run time [Dependency Injection]

Tight coupling vs Loose coupling

1. Spider man Toy (Tight Coupling)
2. Lego Spiderman Toy ( Loose Coupling)

Spring is also called as IoC Container. [It can change the way objects get created]

Created Objects will be injected to the references at the run time [Dependency Injection]

**Spring framework is mainly used to create Loosely coupled enterprise java applications. By using dependency injection**

Spring is also called as Framework of frameworks.

It supports other Java based frameworks like struts, Hibernate, Velocity, ….

Spring framework contains many different modules

1. Spring-core
2. Spring-Web
3. Spring-Data (JDBC, JPA)
4. Spring-Security

Spring uses POJO classes

POJO – Plain Old Java Object (A Class which is not extending any other class nor implementing any interfaces) POJO/Simple/Concrete class all are same.



<p align=”center”> </p>

Normally for the attributes we used to give static values only.

Using Expression Language (used to provide dynamic values to the attributes at run time)

<p align=”${alignment}” > </p>

Spring is developed by Rod Johnson in 2003. Why the name Spring?

Winter, Summer, Monsoon, Spring

Types of Dependency Injection

Constructor Injection

<beans>

<bean id = "order" class = "com.revature.models.Order">

<constructor-arg ref="account"/>

<constructor-arg ref="item"/>

</bean>

<bean id = "account" class = "com.revature.models.Account">

</bean>

<bean id = "item" class = "com.revature.models.Item">

</bean>

</beans>

Setter Injection (SI)

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

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

<bean id=*"studentbean"* class=*"com.revature.model.Student"*>

<property name=*"id"* value=*"100"*></property>

<property name=*"name"* value=*"Sample Name"*></property>

</bean>

</beans>