SPRING FRAMEWORK NOTES

**What is spring?**

The Spring Framework is an open-source application framework that provides infrastructure support for developing Java applications

**Spring Uses:**

* Provides to create high performing, easily testable and reusable code
* Spring is organized in a modular fashion simplifies java development
* Spring Framework is a well-designed web model-view-controller (MVC) framework
* The Inversion of Control (IoC) containers are lightweight.
* Testing is simple because environment-dependent code is moved into this framework.

**Beans:**

* Beans are created with the configuration metadata (XML file) that we supply to the container.
* Bean definition contains configuration metadata
* Spring supports given scope types for beans:
* Singleton
* Prototype
* Request
* Session
* Global-session

**Dependency Injection**

* Spring is most identified with Dependency Injection (DI) technology.
* DI is only one concrete example of Inversion of Control.
* In a complex Java application, classes should be loosely coupled. This feature provides code reuse and independently testing classes
* DI helps in gluing loosely coupled classes together and at the same time keeping them independent.
* DI makes testing easier
* DI will be accomplished by given two ways:
* passing parameters to the constructor
* using setter methods

**Spring framework Component**

**Core container**

* The Core module provides the fundamental parts of the framework, including the IoC and Dependency Injection features.
* The Bean module provides BeanFactory which is a sophisticated implementation of the factory pattern.
* The Context module builds on the solid base provided by the Core and Beans modules and it is a medium to access any objects defined and configured.
* The Expression Language module provides a powerful expression language for querying and manipulating an object graph at runtime.

**Data Access**

* The JDBC module provides a JDBC-abstraction layer that removes the need to do tedious JDBC related coding.
* The ORM module provides integration layers for popular object-relational mapping APIs, including JPA, JDO, Hibernate, and iBatis.
* The Transaction module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.

**Web**

* The Web-Servlet module contains Spring's model-view-controller (MVC) implementation for web applications.
* The Web-Portlet module provides the MVC implementation to be used in a portlet environment and mirrors the functionality of Web-Servlet module.
* The  Web-Struts  module contains the support classes for integrating a classic Struts web tier within a Spring application.

**Inversion of Control**

Inversion of control (IoC) is a programming technique in which object coupling is bound at run time by an assembler object and is typically not known at compile time using static analysis.

**Dependency Injection Contoller**

**Spring BeanFactory Container**

This is the simplest container providing basic support for DI. There are a number of implementations of the BeanFactory interface that come supplied straight out-of-the-box with Spring. The most commonly used BeanFactory implementation is the XmlBeanFactory class

**Spring ApplicationContext Container**

The ApplicationContext includes all functionality of the BeanFactory, it is generally recommended over the BeanFactory. It adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners.

**Annotation Based Configuration**

**@Autowired**

The @Autowired annotation can apply to bean property setter methods, non-setter methods, constructor and properties.

**@Qualifier**

The @Qualifier annotation along with @Autowired can be used to remove the confusion by specifiying which exact bean will be wired.

**@Required**

The @Required annotation applies to bean property setter methods.

**@Configuration**

It indicates that the class can be used by the Spring IoC container as a source of bean definitions

**@Bean**

Spring that a method annotated with @Bean will return an object that should be registered as a bean in the Spring application context.

**Spring Boot Annotations**

* @GetMapping

It is used to handle GET type of request method

* @PostMapping

It  is used to handle POST type of request method, etc.

* @PutMapping

This  annotation for mapping HTTP PUT requests onto specific handler methods.

* @DeleteMapping

The DELETE HTTP method is used to delete the resource and @DeleteMapping annotation for mapping HTTP DELETE requests onto specific handler methods.

**Spring Web MVC Framework**

* The Spring web MVC framework provides model-view-controller architecture and ready components that can be used to develop flexible and loosely coupled web applications.
* The **Model** encapsulates the application data and in general they will consist of POJO.
* The **View** is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.
* The **Controller** is responsible for processing user requests and building appropriate model and passes it to the view for rendering.

**Controller**

* DispatcherServlet delegates the request to the controllers to execute the functionality specific to it.
* The @Controller annotation indicates that a particular class serves the role of a controller.
* The @RequestMapping annotation is used to map a URL to either an entire class or a particular handler method.

**@RequestMapping**

URI templates can be used for convenient access to selected parts of a URL in a @RequestMapping method.

**HIBERNATE**

Hibernate is a high-performance Object/Relational persistence and query service. Hibernate ORM facilitated the storage and retrieval of Java domain objects via Object/Relational Mapping.

**Hibernate Architecture**

**CONFIGURATION OBJECT**

The Configuration object is the first Hibernate object you create in any Hibernate application and usually created only once during application initialization. It represents a configuration or properties file required by the Hibernate.

**SESSIONFACTORY OBJECT**

Configuration object is used to create a SessionFactory object which inturn configures Hibernate and allows for a Session object to be instantiated. The SessionFactory is a thread safe object and used by all the threads of an application.

**SESSION OBJECT**

A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database.

**TRANSACTION OBJECT**

A Transaction represents a unit of work with the database and most of the RDBMS supports transaction functionality. Transactions in Hibernate are handled by an underlying transaction manager and transaction (from JDBC or JTA)