**Index**

* **What is spring framework and Explain Spring feature and drawback?**
* **Explain module of spring?**
* **difference b/w different version of spring (2.0,2.5,3 and 3.1 and 4.0)**

**http://spring.io/blog/2011/02/15/spring-3-1-m1-unified-property-management**

* **Instantiating an Application Context and Instantiating IOC container**
* **Different type to instantiate application context**
* **Different controller in spring**
* **what is bean factory**
* **Difference B/w bean factory and application context**
* **Difference b/w context-annotation-config and context-component-scan**
* **Configuring Beans in the Spring IOC Container**
* **what is IOC and Dependency Injection and different type?**
* **Difference b/w setter injection and constructor injection?**
* **Can we override bean name?**
* **How define bean property as attribute of bean**
* **Anonymous Bean**
* **Configuring Collections for Your Beans**
* **Merging the Collection of the Parent Bean**
* **Resolving Constructor Ambiguity**
* **Specifying Bean References(In setter injection and constructor) and use of local**
* **Declaring Inner Beans**
* **Specifying the Data Type for Collection Elements**
* **Creating Beans Using Spring’s Factory Bean**
* **Defining Collections Using Factory Beans and the Utility Schema**
* **Checking property with dependency checking mode**
* **Checking dependency with @required annotation**
* **Auto wiring and different mode.**
* **Where we can apply autowire and autowire all bean of compatible type**
* **Dependency check**
* **@Qualifier annotation and custom annotation.**
* **Abstract bean**
* **Inheriting and overriding bean property.**
* **Component scanning**
* **Creating bean by invoking static factory bean**
* **Bean with static field**
* **Declaring bean from object property(Need to do)**
* **Expression Language(Need to do)**
* **Bean scope**
* **Customizing Bean Initialization and Destruction**
* **Different Bean Aware Interfaces and execution order in bean life cycle**
* **Loading External Resources(Context:property-placeholder location=)**
* **Bean Post Processor and what is the use of it ?**
* **Difference b/w bean post processor and bean factory post processor.**
* **Message Localization(ResourceBundleMessageResource)**
* **Event handling**
* **Property Editor**
* **What is AOP,Advice,Aspect,JoinPoint and pointcut**
* **AOP Advice by implementing advice interface**
* **AOP Advice with custom annotation in xml**
* **Accesing joinPoing information**
* **AOP Advice with operator(and) and args**
* **AOP Advice with Args**
* **AOP Advice by expression in xml**
* **AOP Advice with target**
* **AOP Advice with @within**
* **AOP Advice with “this”**
* **AOP Advice with Bean**
* **AOP Ordering**
* **Reuse pointcut**
* **Difference b/w “this” and “target”**
* **Method Signature Patterns**
* **Bean name patterns**
* **Weaving**
* **Aspect of factory method**
* **Injecting Spring Beans into Domain Objects**

**---------------------------------------------Spring security-------------------------**

* **How Spring handle security**
* **Different default services and how to set automatically**
* **Spring security configuration(Filter,listener,DeletingFilterProxy)**
* **How to make web url secure**
* **Spring security with db based authentication**
* **Session management and login and logut element attribute**
* **How to control concurrent session in spring security?**
* **Authentication by memory data source**
* **Authentication by property file**
* **Anonymous Login**
* **Spring Remember me support**
* **LDAP Authentication and ldap attributes and cofiguration**
* **Decission Handler and configuration**
* **Ecache ( bean configuration,attributes)**
* **Method invocation secure(normal element, pointcut,annoation)**
* **Displaying authentication infromation**
* **Handling domain object securly**
* **ACL for domain object**
* **Spring integration with other Framework(get context)**
* **WebApplicationContext context WebApplicationContextUtils.getRequiredWebApplicationContext(getServletContex))**
* **HttpRequestHandler**
* **How web application load springContext in other frameork**
* **--------------------- Spring webflow -------------------------------**
* **Spring webflow state**
* **web flow configuration and explain different attribute**
* **event id convention in web flow**

**--------------------------------------Spring MVC -------------------------------**

* **What is mvc and Explain mvc flow?**
* **What is front controller?**
* **What is controller and handler method?**
* **Type of parameter handler method can accept in controller**
* **How view represented and how to pass data from controller to view?**
* **What is view resolver and how view populated?**
* **MVC configuration file name convention**
* **what is context loader**
* **Do each dispatcher servlet instance loads its own file in its context?**
* **How to activate spring mvc annotation**
* **Bean require to work @Requestmapping**
* **Which default method Controller redirect each request? What will happen if default method is not there?**
* **How does the handler mapping work?**
* **Can one handler map to multiple URL?**
* **How to pass input to handler method from request?**
* **Can a handler method have return type void then how does it work?**
* **Why do we need to specify other http method in @request Mapping?**
* **Different way to intercept a request? Bean require registering interceptor.**
* **Difference b/w filter and interceptor?**
* **What is selectedAnnotationHandlerMapping and how it is different b/w defaultannotaionahandelermapping?**
* **Different Interface use to resolve locale and for http?**
* **How to change user locale?**
* **Bean required to get message resource from different resource bundle**
* **Different way to resolve view name(bean name)**
* **Property required to set to resolve view using multiple resolver Bean**
* **How contentNegotatingviewresolver works?**
* **Exception handling using simpleMappingExceptionResolver**
* **Data binding with view**
* **Data Validation**
* **Multiple form handing using wizard**
* **Data validation using JSR-303**

**--------------------------------------Spring Rest -------------------------------**

* **Annotation required to publish data in spring rest**
* **how to make controller handler method as rest service end point**
* **Different marshaller supported by spring**
* **Class required to access third party services and method**
* **What is abstractatomfeedview and abstractrssfeedview**
* **Accessing REST Services with Elaborate XML Responses**

**--------------------------------------Spring Flex -------------------------------**

**--------------------------------------Spring Data Access -------------------------------**

**https://www.youtube.com/watch?v=YCC-CpTE2LU**

* **Different ORM tool supported by spring**
* **Bean is required to configure data source**
* **different bean to configure data source**
* **Bean require for JNDI**
* **Explain** PreparedStatementSetter and **preparestatementcreater**
* **explain Batchpreparedstatementsetter**
* **what is row callback handler**

**Links :**

[**http://springcert.sourceforge.net/2.5/2-study-aop.html**](http://springcert.sourceforge.net/2.5/2-study-aop.html)

[**http://www.tkhts.com/spring/spring\_component\_scanning.jsp**](http://www.tkhts.com/spring/spring_component_scanning.jsp)

[**http://www.knowledgewalls.com/j2ee/books/spring-30-examples/spring-aops-pointcut-expression-using-within-aspectj-designator-with-example(aop**](http://www.knowledgewalls.com/j2ee/books/spring-30-examples/spring-aops-pointcut-expression-using-within-aspectj-designator-with-example(aop)**)**

<http://www.byteslounge.com/tutorials/spring-aop-pointcut-advice-example(aop)see>

Index

* **What is IOC (or Dependency Injection)?**
* **What are features of spring?**
* **What is the difference between spring 3.2,3.1 3.0 and Spring 2.5,2.0?**
* **What is Bean Factory?**
* **What are the different types of IOC (dependency injection) ?**
* **What is Application Context?**
* **What is the Bean?**
* **What are the common implementations of the Application Context?**
* **Explain lifecycle of beans?**
* **What is auto wiring and difference mode of auto ?**
* **What are Bean scopes in Spring Framework?**
* **What are the types of the transaction management Spring supports ?**
* **What is the difference b/w <Context-component-scan/> and <Context-annotation-config/>?**

**Q1. What is IOC (or Dependency Injection)?**

Ans. The basic concept of IOC pattern is that ‘NO need to create the object, but describe how they should be created and no need to connect services and components together in code but describe which service are needed by which component.

- Giving control to container to get the instance of object is called IOC. Means instead of creating the instance using new container does for us.

- Way of injecting the properties to an object is called DI.

**Q. 2.  What are features of spring?**

Ans:

* **Lightweight**: spring is lightweight when it comes to size and transparency. The basic version of spring framework is around 1MB. And the processing overhead is also very negligible.
* **Inversion of control (IOC):** Loose coupling is achieved in spring using the technique Inversion of Control. The objects give their dependencies instead of creating or looking for dependent objects.
* **Aspect oriented (AOP**): Spring supports Aspect oriented programming and enables cohesive development by separating application business logic from system services.
* **Container**: Spring contains and manages the life cycle and configuration of application objects.
* **MVC Framework**: Spring comes with MVC web application framework, built on core Spring functionality. This framework is highly configurable via strategy interfaces, and accommodates multiple view technologies like JSP, Velocity, Tiles, iText, and POI. But other frameworks can be easily used instead of Spring MVC Framework.
* **Transaction Management**: Spring framework provides a generic abstraction layer for transaction management.
* **JDBC Exception Handling**: The JDBC abstraction layer of the Spring offers a meaningful exception hierarchy, which simplifies the error handling strategy. Integration with Hibernate, JDO, and iBATIS: Spring provides best Integration services with Hibernate, JDO and iBATIS

**Q What is the difference between spring 3.0 and Spring 2.5?**

Ans:

|  |  |
| --- | --- |
| **Spring Framework 3.0** | **Spring Framework 2.5** |
| Spring framework 3.0 is compatible with Java 5 and higher versions. | Spring framework 2.5 is compatible with Java 1.4 and higher versions. |
| Spring framework 3.0 introduces Spring Expression Language which defines bean definitions based XML and Annotation. | In Spring framework 2.5, native expression language exists which is less powerful than spring expression language of spring 3.0 and it has different parsing rules. |
| Spring framework 3.0 has type converting system and field formatting. | Spring framework 2.5 does not support type conversions and field formatting. |
| Spring framework 3.0 fully supports the JSR303 bean validation API. | Spring framework 2.5 does not offer complete support for JSR303 bean validation API. |
| Spring framework 3.0 offers support for various embedded database engines like HSQL, Derby and H2. | Spring framework 2.5 does not support the embedded database engines. |
| Comprehensive REST support is available in Spring framework 3.0. | Spring framework 2.5 does not support Comprehensive REST. |
| Spring framework 3.0 automatically validates the @Controller inputs. | Spring framework 2.0 does not validate the @Controller inputs automatically. |



**Q. 3.  What is the new feature in spring 2.5?**

**Ans :** In spring 2.5 following new features are added:

* Added new bean scope
* Spring xml configuration is easy than spring 2.5. : Doc-Type declaration removed and xml schema based configuration added.
* In Spring 2.5 xml configuration is extensible.(developer can create custom tag)
* Support for @aspecjJ
* New annotation is addd (preDestory,postConstruct) and component scanning
* New classes added in jdbc(simpleJdbcTemplate)
* Support for jax-ws and JSF 1.2
* Support for annotation based controller(RequestParam,RequestBody)
* New form tag library added for spring mvc
* Auto detecting component in classpath
* Support for dynamic language.

**Q. 3.  What is the new feature in spring 3.1?**

**Ans :**

* Cache abstraction
* @profile
* Environment abstraction
* Property source abstraction
* C: namespace for concise constructor injection
* Flash and redirect Attribute
* Uri template variable enhancement
* Consume and produce condition in request mapping
* Servlet 3 multipart Resolver support
* Code equivalent to spring’s xml namespace
* @ valid on @Request Body controller method arguments
* @Request Part annotation on controller method arguments

**Q. 3.  What is the new feature in spring 3.2?**

**Ans :**

* Support for Servlet 3 based asynchronous request processing
* @ControllerAdvice annotation
* Matrix annotation
* Support for generic types in the RestTemplate and in @RequestBody arguments
* @DateTimeFormat
* Initial support for JCache 0.5

**Q. 3.  What are the different types of IOC (dependency injection) ?**

Ans:

* **Constructor Injection** (e.g. Pico container, Spring etc): Dependencies are provided as constructor parameters.
* **Setter Injection** (e.g. Spring): Dependencies are assigned through JavaBeans properties (ex: setter methods).
* **Interface Injection** (e.g. Avalon): Injection is done through an interface.

**Q4.  What is Bean Factory?**

Ans :

* *BeanFactory* is represented by *org.springframework.beans.factory.BeanFactory.*
* It provides DI / IOC mechanism for the spring.
* It loads the beans definitions and their property descriptions from some configuration source.
* It is built upon Factory Design Pattern.
* Instantiates the beans when they are requested.
* Wire dependencies and properties for the beans according to their configuration defined in configuration source.

*Ex.*

BeanFactory bfObj = new XmlBeanFactory(new FileSystemResource ("c:/beansconfig.xml"));

MyBean beanObj= (MyBean) bfObj.getBean("mybean");

**Q What is Application Context?**

Ans :

* A means for resolving text messages, including support for internationalization.
* A generic way to load file resources.
* Events to beans that are registered as listeners.

**Implementaion of ApplicationContext :**

ApplicationContext context = new FileSystemXmlApplicationContext("c:/myconfig.xml");

ApplicationContext context = new ClassPathXmlApplicationContext("myconfig.xml");

**Q What is the difference between Bean Factory and Application Context ?**

Ans.

* Application contexts provide a means for resolving text messages, including support for i18n of those messages.
* Application contexts provide a generic way to load file resources, such as images.
* Application contexts can publish events to beans that are registered as listeners.
* Certain operations on the container or beans in the container, which have to be handled in a programmatic fashion with a bean factory, can be handled declaratively in an application context.
* ResourceLoader support: Spring’s Resource interface us a flexible generic abstraction for handling low-level resources. An application context itself is a ResourceLoader, Hence provides an application with access to deployment-specific Resource instances.
* MessageSource support: The application context implements MessageSource, an interface used to obtain localized messages, with the actual implementation being pluggable
* Application context support annotation based dependency injection but bean factory doesn’t support.
* Bean factory by default support lazy loading and application context support aggressive loading.
* When we use Bean factory as IOC Container bean Bean-Postprocessor should be registered through addBeanPostProcessor(). But in case of application Context the Bean-Postprocessor registered automatically.

**Q What is the Bean?**

**Ans :** Bean is a object which is instantiate and managed by IOC container. Beans are the objects which form the backbone of our application.

**Q . What are the common implementations of the Application Context?**

   The three commonly used implementation of 'Application Context' are

* **ClassPathXmlApplicationContext :** It Loads context definition from an XML file located in the classpath, treating context definitions as classpath resources. The application context is loaded from the application's classpath by using the code .  
  ApplicationContext context = new ClassPathXmlApplicationContext("bean.xml");
* **FileSystemXmlApplicationContext :** It loads context definition from an XML file in the filesystem. The application context is loaded from the file system by using the code .  
  ApplicationContext context = new FileSystemXmlApplicationContext("bean.xml");
* **XmlWebApplicationContext :** It loads context definition from an XML file contained within a web application.

**Q.** **Explain lifecycle of beans?**

Ans :

- IOC container looks for configuration metadata for given bean.

- Instantiate the bean

- Populate the bean injecting DI

- Call setBeanName()method of BeanNameAware class if bean implements BeanNameAware .

- Call setFactoryName() method of BeanFactoryAware class if bean implements BeanFactoryAware.

* The Bean Container finds the definition of the Spring Bean in the Configuration file.
* The Bean Container creates an instance of the Bean using Java Reflection API.
* If any properties are mentioned, then they are also applied. If the property itself is a Bean, then it is resolved and set.
* If the Bean class implements the BeanNameAware interface, then the setBeanName() method will be called by passing the name of the Bean.
* If the Bean class implements the BeanClassLoaderAware interface, then the method setBeanClassLoader() method will be called by passing an instance of the ClassLoader object that loaded this bean.
* If the Bean class implements the BeanFactoryAware interface, then the method setBeanFactory() will be called by passing an instance of BeanFactory object.
* If there are any BeanPostProcessors object associated with the BeanFactory that loaded the Bean, then the method postProcessBeforeInitialization() will be called even before the properties for the Bean are set.
* If the Bean class implements the InitializingBean interface, then the method afterPropertiesSet() will be called once all the Bean properties defined in the Configuration file are set.
* If the Bean definition in the Configuration file contains a 'init-method' attribute, then the value for the attribute will be resolved to a method name in the Bean class and that method will be called.
* The postProcessAfterInitialization() method will be called if there are any Bean Post Processors attached for the Bean Factory object.
* If the Bean class implements the DisposableBean interface, then the method destroy() will be called when the Application no longer needs the bean reference.
* If the Bean definition in the Configuration file contains a 'destroy-method' attribute, then the corresponding method definition in the Bean class will be called.

Q. **What is auto wiring and difference mode of auto** wiring?(http://howtodoinjava.com/2013/05/08/spring-bean-autowire-bytype)

Ans :

* no
* byName
* byType
* constructor
* autodirect

**Q. What are Bean scopes in Spring Framework?**

|  |  |
| --- | --- |
| singleton | Single object instance per Spring IoC container. |
| prototype | Scopes a single bean definition to any number of object instances. |
| request | Each and every HTTP request will have its own instance of a bean created off the back of a single bean definition. |
| session | Scopes a single bean definition to the lifecycle of a HTTP Session. Only valid in the context of a web-aware Spring ApplicationContext. |
| global session | Scopes a single bean definition to the lifecycle of a global HTTP Session. Typically only valid when used in a portlet context. Only valid in the context of a web-aware Spring ApplicationContext. |

**Q. What are the types of the transaction management Spring supports ?**

Ans :

* Programmatic transaction management.
* Declarative transaction management.

**Q. What is the difference b/w <Context-component-scan/> and <Context-annotation-config/>?**

Ans :

* <Context-annotation-config/> activate applied annotation in already registered bean in application context.
* <Context-component-scan/> does what <context-annotation-config> does ,additionally it scan package and registered the bean in application context.

**Q. Bean life cycle**

1. BeanNameAware's setBeanName  
2. BeanClassLoaderAware's setBeanClassLoader  
3. BeanFactoryAware's setBeanFactory  
4. ResourceLoaderAware's setResourceLoader (only applicable when running in an application context)  
5. ApplicationEventPublisherAware's setApplicationEventPublisher (only applicable when running in an application context)  
6. MessageSourceAware's setMessageSource (only applicable when running in an application context)  
7. ApplicationContextAware's setApplicationContext (only applicable when running in an application context)  
8. ServletContextAware's setServletContext (only applicable when running in a web application context)  
9. postProcessBeforeInitialization methods of BeanPostProcessors  
10. InitializingBean's afterPropertiesSet  
11. a custom init-method definition  
12. postProcessAfterInitialization methods of BeanPostProcessors

**--------------------------------------AOP-----------------------------------------------------**

* What is aop and advantage of AOP.

Ans :

1.Aspect Oriented Programming helps overcome system level coding i.e. Logging, Transaction or Security management problem by centralizing these cross-cutting concerns.

2. Aspect Oriented Programming addresses each aspect separately in a modular fashion with minimal coupling and duplication of code. This modular approach also promotes code reuse by using a business logic concern with a separate logger aspect

**Index**

netstat -nao | findstr 8080 mvn jetty:run -Djetty.http.port=9999