Spring

1. What is Loose Coupling?

In Spring Framework loose coupling is achieved by a core feature called Dependency Injection and IoC container. In dependency injection, components define their own dependencies when they need. This makes the applications easy to test and manage. The dependency injection can be done in two ways:

1. By using constructor (Constructor Args)
2. By using setter-methods (using property tag)
3. What is a Dependency?

Dependency injection is a pattern through which to implement IoC, where the control being inverted is the setting of object’s dependencies.

The act of connecting objects with other objects, or “injecting” objects into other objects, is done by an assembler rather than by the objects themselves.

Here’s how you would create an object dependency in traditional programming:

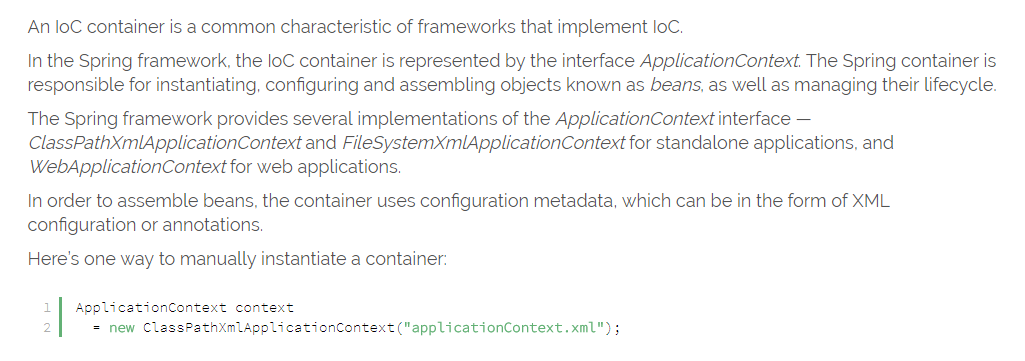
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public class Store {      private Item item;        public Store() {          item = new ItemImpl1();      }  } |

In the example above, we need to instantiate an implementation of the Item interface within the Store class itself.

By using DI, we can rewrite the example without specifying the implementation of Item that we want:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | public class Store {      private Item item;      public Store(Item item) {          this.item = item;      }  } |

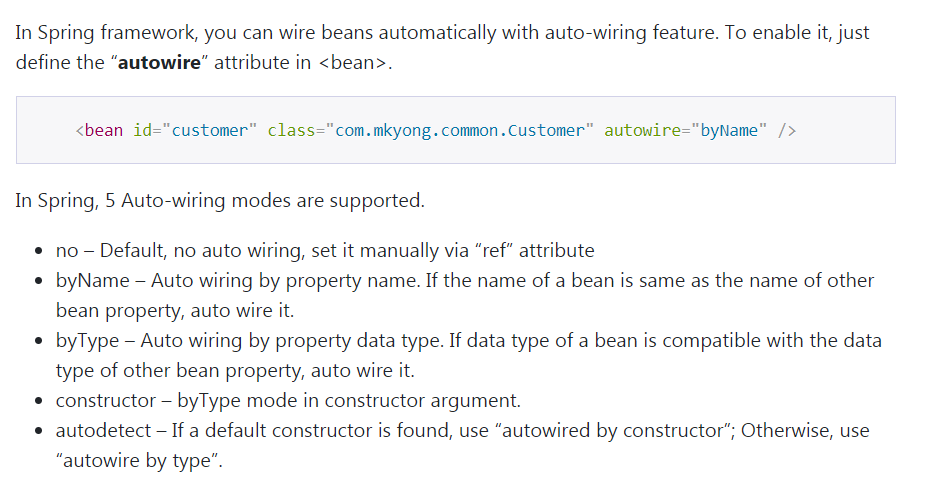
1. What is IOC (Inversion of Control) container?

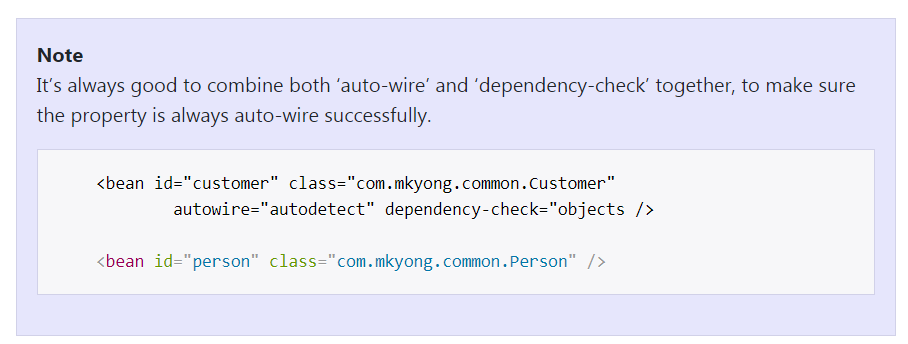


1. Can you give few examples of Dependency Injection?

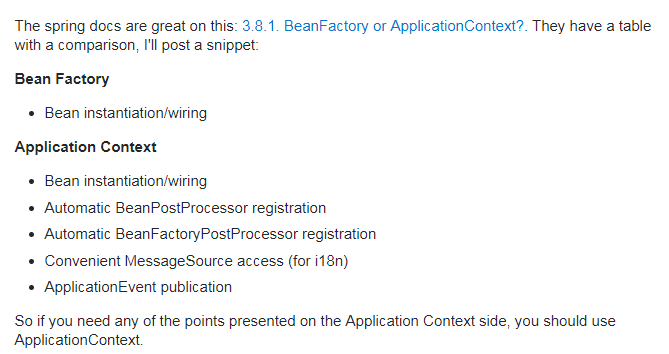


1. What is Auto Wiring?

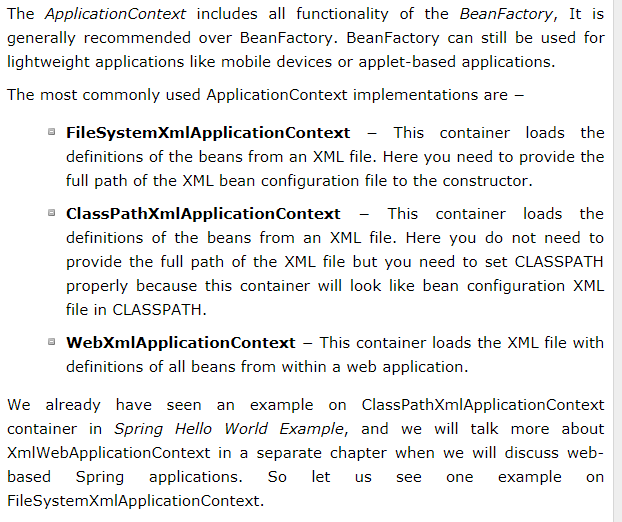


http://www.mkyong.com/spring/spring- auto-wiring-beans-in-xml/

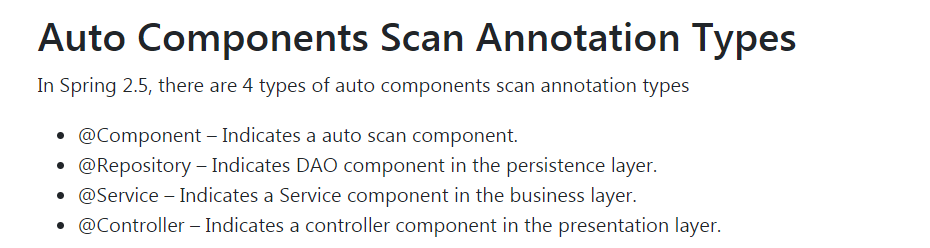
1. What are the important roles of an IOC Container?
2. What are Bean Factory and Application Context?



1. How do you create an application context with Spring?



1. How does Spring know where to search for Components or Beans?



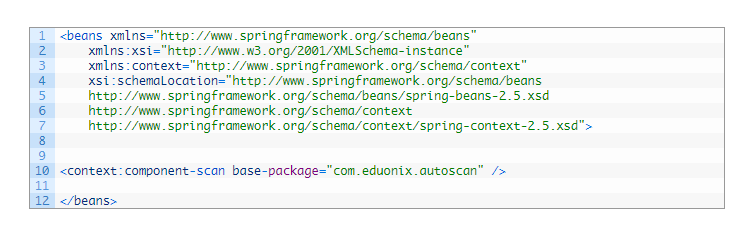
1. What is a Component Scan?

Automatic component scan is largely important since it helps reduce the amount of XML, as there is no need to declare bean definitions separately in the spring-config.xml file. Also, since you do not have to declare bean definitions separately, it does save a lot of time and makes the process simple.

1. How do you define a component scan in XML and Java Configurations?

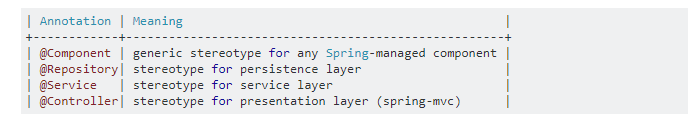


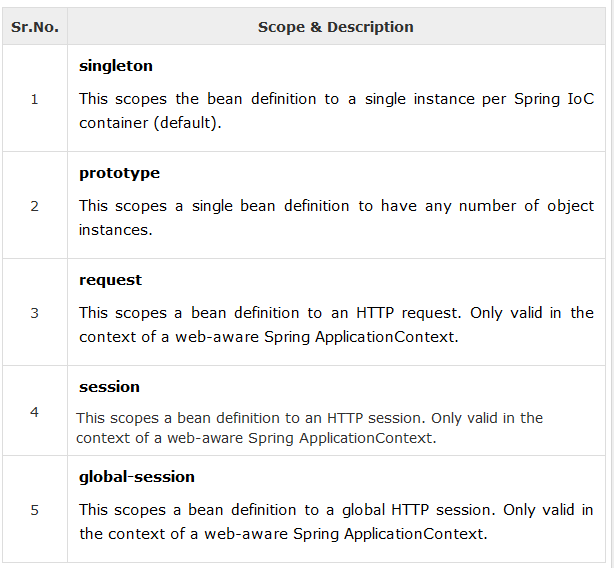
https://blog.eduonix.com/java-programming-2/learn-component-scanning-works-spring-framework/



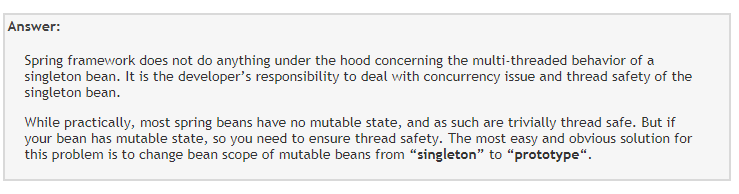
1. How is it done with Spring Boot?



1. What does @Component signify?
2. What does @Autowired signify?
3. What’s the difference Between @Controller, @Component, @Repository, and @Service Annotations in Spring?
4. 
5. What is the default scope of a bean?



1. Are Spring beans thread safe?



1. What are the other scopes available?
2. How is Spring’s singleton bean different from Gang of Four Singleton Pattern?
3. What are the different types of dependency injections?
4. What is setter injection?
5. What is constructor injection?
6. How do you choose between setter and constructor injections?
7. What are the different options available to create Application Contexts for Spring?
8. What is the difference between XML and Java Configurations for Spring?
9. How do you choose between XML and Java Configurations for Spring?
10. How does Spring do Autowiring?
11. What are the different kinds of matching used by Spring for Autowiring?
12. How do you debug problems with Spring Framework?
13. How do you solve NoUniqueBeanDefinitionException?
14. How do you solve NoSuchBeanDefinitionException?
15. What is @Primary?

**@Primary**

This Indicates that a particular bean should be given preference when multiple beans are candidates to be autowired to a single-valued dependency. If exactly one 'primary' bean exists among the candidates, it will be the autowired value.

When we have multiple dependency it will consider primary as default loader

1. What is @Qualifier?

When two beans has same name and you need mention which bean need to taken in qualifier

@Qualifier

If there are more than one instances available for an injection point then we can use @Qualifierannotation to resolve an ambiguity. As @Qualifier is used at the injection point, there might be two situations where we don't want to or cannot use @Qualifier.

1. When autowiring mode is Autowire.BY\_TYPE. Then, of course, we cannot use @Qualifierbecause we actually don't have user-defined injection point specified as @Autowired or @Inject
2. We want to do bean selection (i.e. resolve the ambiguity) at configuration time rather than during beans development time.

The solution to above problems is to use @Primary annotation.

1. What is CDI (Contexts and Dependency Injection)?
2. Does Spring Support CDI?
3. Would you recommed to use CDI or Spring Annotations?

This question has been around for many years since Spring started to move heavily towards Annotation based configurations (if I recall, it’s called configuration by convention). Annotations based configurations were like a jungle fire that spread across the industry and very soon it was the norm. But, this question “XML vs. Annotations” always existed.

I for one have been around the Spring world since version 1.1 when annotations weren’t a thing. I know how write XML and I know how to configure an application to suit my needs. Since then, whenever I write an application in Spring, I have asked myself "XML or Annotations" and I never really had a good answer until recently. While you will find tons of [posts around this topic in Google](https://www.google.co.in/search?q=Spring+XML+vs.+Annotations&oq=Spring+XML+vs.+Annotations&aqs=chrome..69i57.454j0j1&sourceid=chrome&es_sm=91&ie=UTF-8),  when you search for this, only a few really give you an unbiased opinion.

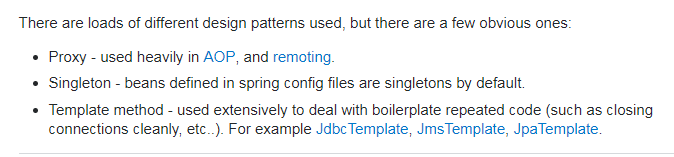
I started to work on an application that needs some very flexible configuration options and before I dive into that I had to make this decision yet again, and this time I wanted to keep things simple and my rationale was…

* Use Annotations in anything that is stable and defines the core structure of the application. Anything that would need a code change is okay to sit as an annotation.

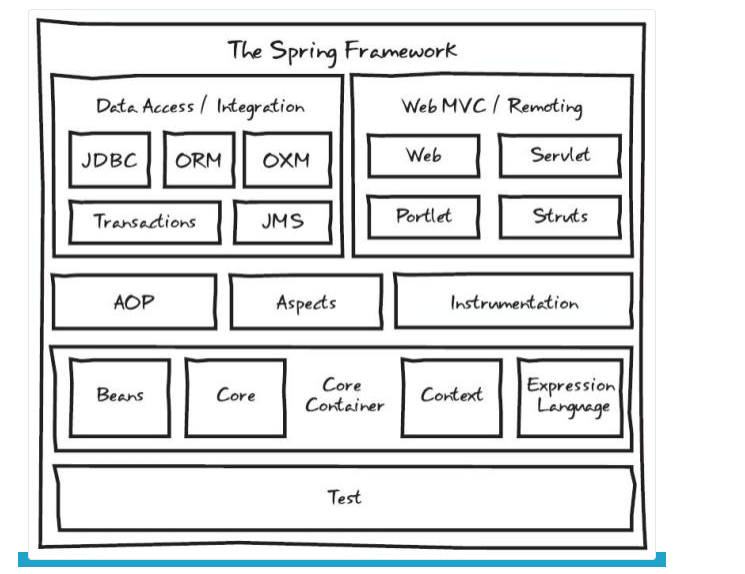
Use XML based configurations when you know you may have a need to alter the behavior of an application without the need of compiling and deploying the code all over again.

This is how simple I kept it for my team. Once this principle is defined, job is only halve done. But we will get there soon.

1. What are the major features in different versions of Spring?
2. What are new features in Spring Framework 4.0?
3. What are new features in Spring Framework 5.0?
4. What are important Spring Modules?
5. What are important Spring Projects?
6. What is the simplest way of ensuring that we are using single version of all Spring related dependencies?
7. Name some of the design patterns used in Spring Framework?

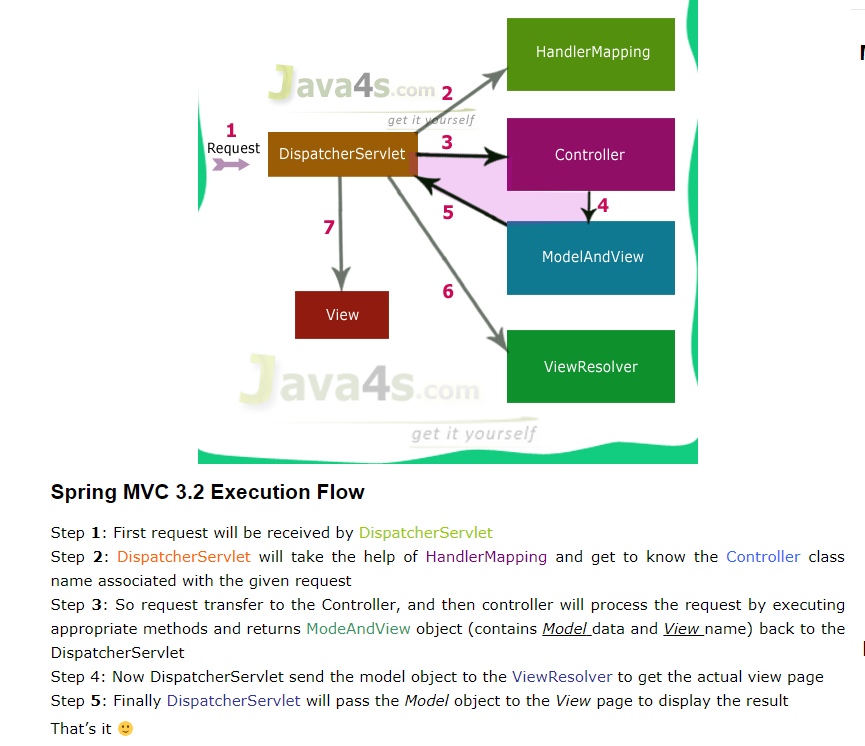


1. What do you think about Spring Framework?
2. Why is Spring Popular?
3. Can you give a big picture of the Spring Framework?

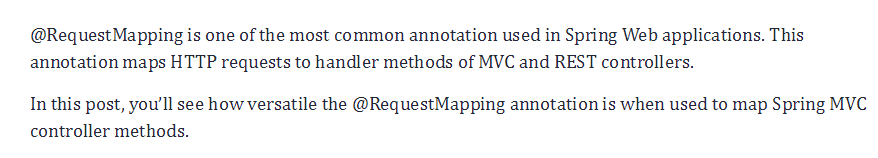


Spring MVC

1. What is Model 1 architecture?
2. What is Model 2 architecture?
3. What is Model 2 Front Controller architecture?
4. Can you show an example controller method in Spring MVC?
5. Can you explain a simple flow in Spring MVC?



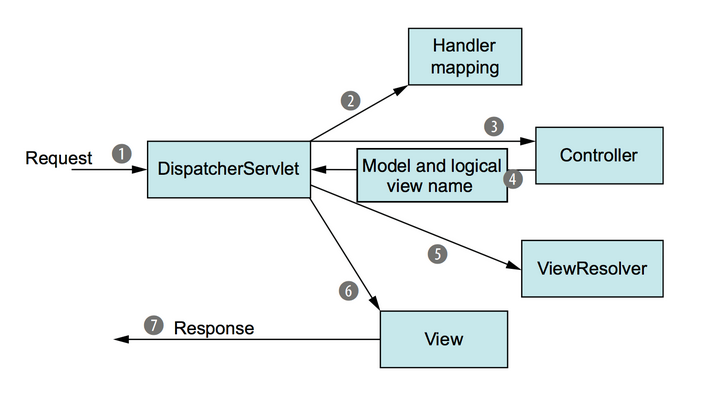
1. What is a ViewResolver?
2. What is Model?
3. What is ModelAndView?
4. What is a RequestMapping?



1. What is Dispatcher Servlet?

n Spring MVC all incoming requests go through a single servlet. This servlet - DispatcherServlet - is the front controller. Front controller is a typical design pattern in the web applications development. In this case, a single servlet receives all requests and transfers them to to all other components of the applicatio

1. How do you set up Dispatcher Servlet?

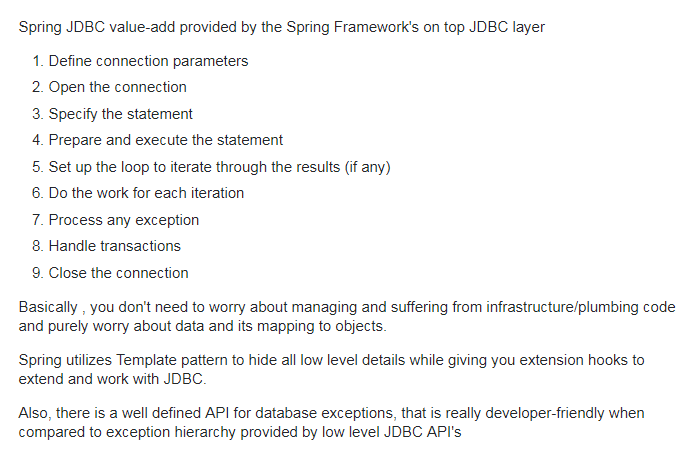
1. What is a form backing object?
2. How is validation done using Spring MVC?
3. What is BindingResult?
4. How do you map validation results to your view?
5. What are Spring Form Tags?
6. What is a Path Variable?
7. What is a Model Attribute?
8. What is a Session Attribute?
9. What is a init binder?
10. How do you set default date format with Spring?
11. Why is Spring MVC so popular?

Spring Boot

1. What is Spring Boot?
2. What are the important Goals of Spring Boot?
3. What are the important Features of Spring Boot?
4. Compare Spring Boot vs Spring?
5. Compare Spring Boot vs Spring MVC?
6. What is the importance of @SpringBootApplication?
7. What is Auto Configuration?
8. How can we find more information about Auto Configuration?
9. What is an embedded server? Why is it important?
10. What is the default embedded server with Spring Boot?
11. What are the other embedded servers supported by Spring Boot?
12. What are Starter Projects?
13. Can you give examples of important starter projects?
14. What is Starter Parent?
15. What are the different things that are defined in Starter Parent?
16. How does Spring Boot enforce common dependency management for all its Starter projects?
17. What is Spring Initializr?
18. What is application.properties?
19. What are some of the important things that can customized in application.properties?
20. How do you externalize configuration using Spring Boot?
21. How can you add custom application properties using Spring Boot?
22. What is @ConfigurationProperties?
23. What is a profile?
24. How do you define beans for a specific profile?
25. How do you create application configuration for a specific profile?
26. How do you have different configuration for different environments?
27. What is Spring Boot Actuator?
28. How do you monitor web services using Spring Boot Actuator?
29. How do you find more information about your application envrionment using Spring Boot?
30. What is a CommandLineRunner?

Database Connectivity - JDBC, Spring JDBC & JPA

1. What is Spring JDBC? How is different from JDBC?



1. What is a JdbcTemplate?
2. What is a RowMapper?
3. What is JPA?

Mapping Java objects to database tables and vice versa is called *Object-relational mapping*(ORM). The Java Persistence API (JPA) is one possible approach to ORM. Via JPA the developer can map, store, update and retrieve data from relational databases to Java objects and vice versa. JPA can be used in Java-EE and Java-SE applications.

1. What is Hibernate?
2. How do you define an entity in JPA?
3. What is an Entity Manager?
4. What is a Persistence Context?
5. How do you map relationships in JPA?
6. What are the different types of relationships in JPA?
7. How do you define One to One Mapping in JPA?
8. How do you define One to Many Mapping in JPA?
9. How do you define Many to Many Mapping in JPA?
10. How do you define a datasource in a Spring Context?
11. What is the use of persistence.xml
12. How do you configure Entity Manager Factory and Transaction Manager?
13. How do you define transaction management for Spring – Hibernate integration?

Spring Data

1. What is Spring Data?
2. What is the need for Spring Data?
3. What is Spring Data JPA?
4. What is a CrudRepository?
5. What is a PagingAndSortingRepository?

Unit Testing

1. How does Spring Framework Make Unit Testing Easy?
2. What is Mockito?
3. What is your favorite mocking framework?
4. How do you do mock data with Mockito?
5. What are the different mocking annotations that you worked with?
6. What is MockMvc?
7. What is @WebMvcTest?
8. What is @MockBean?
9. How do you write a unit test with MockMVC?
10. What is JSONAssert?
11. How do you write an integration test with Spring Boot?
12. What is @SpringBootTest?
13. What is @LocalServerPort?
14. What is TestRestTemplate?

AOP

1. What are cross cutting concerns?
2. How do you implement cross cutting concerns in a web application?
3. If you would want to log every request to a web application, what are the options you can think of?
4. If you would want to track performance of every request, what options can you think of?
5. What is an Aspect and Pointcut in AOP?
6. What are the different types of AOP advices?
7. What is weaving?
8. Compare Spring AOP vs AspectJ?

SOAP Web Services

1. What is a Web Service?
2. What is SOAP Web Service?
3. What is SOAP?
4. Waht is a SOAP Envelope?
5. What is SOAP Header and SOAP Body?
6. Can you give an example of SOAP Request and SOAP Response?
7. What is a SOAP Header? What kind of information is sent in a SOAP Header?
8. Can you give an example of a SOAP Header with Authentication information?
9. What is WSDL (Web Service Definition Language)?
10. What are the different parts of a WSDL?
11. What is Contract First Approach?
12. What is an XSD?
13. Can you give an example of an XSD?
14. What is JAXB?
15. How do you configure a JAXB Plugin?
16. What is an Endpoint?
17. Can you show an example endpoint written with Spring Web Services?
18. What is a MessageDispatcherServlet?
19. How do you configure a MessageDispatcherServlet?
20. How do you generate a WSDL using Spring Web Services?
21. How do you implement error handling for SOAP Web Services?
22. What is a SOAP Fault?

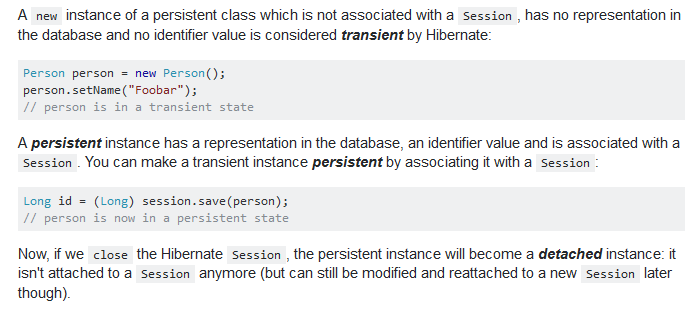
RESTful Web Services

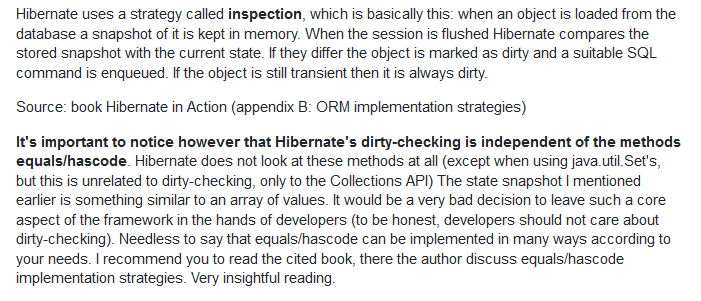
1. What is REST?
2. What are the key concepts in designing RESTful API?
3. What are the Best Practices of RESTful Services?
4. Can you show the code for an example Get Resource method with Spring REST?
5. What happens when we return a bean from a Request Mapping Method?
6. What is GetMapping and what are the related methods available in Spring MVC?
7. Can you show the code for an example Post Resource method with Spring REST?
8. What is the appropriate HTTP Response Status for successful execution of a Resource Creation?
9. Why do we use ResponseEntity in a RESTful Service?
10. What is HATEOAS?
11. Can you give an Example Response for HATEOAS?
12. How do we implement it using Spring?
13. How do you document RESTful web services?
14. Can you give a brief idea about Swagger Documentation?
15. How do you automate generation of Swagger Documentation from RESTful Web Services?
16. How do you add custom information to Swagger Documentation generated from RESTful Web Services?
17. What is Swagger-UI?
18. What is "Representation" of a Resource?
19. What is Content Negotiation?
20. Which HTTP Header is used for Content Negotiation?
21. How do we implement it using Spring Boot?
22. How do you add XML support to your RESTful Services built with Spring Boot?
23. How do you implement Exception Handling for RESTFul Web Services?
24. What are the best practices related to Exception Handling with respect to RESTful Web Services?
25. What are the different error status that you would return in RESTful Web Services?
26. How would you implement them using Spring Boot?
27. What HTTP Response Status do you return for validation errors?
28. How do you handle Validation Errors with RESTful Web Services?
29. Why do we need Versioning for RESTful Web Services?
30. What are the versioning options that are available?
31. How do you implement Versioning for RESTful Web Services?

What are the sample questions covered in this course?

Sample questions covered in this course are as follows:

1. What is Hibernate framework?
2. What is an Object Relational Mapping (ORM)?
3. What is the purpose of Configuration Interface in Hibernate?
4. What is Object Relational Impedance Mismatch?
5. What are the main problems of Object Relational Impedance Mismatch?
6. What are the key characteristics of Hibernate?
7. Can you tell us about the core interfaces of Hibernate framework?
8. How will you map the columns of a DB table to the properties of a Java class in Hibernate?
9. Does Hibernate make it mandatory for a mapping file to have .hbm.xml extension?
10. What are the steps for creating a SessionFactory in Hibernate?
11. Why do we use POJO in Hibernate?
12. What is Hibernate Query Language (HQL)?
13. How will you call a stored procedure in Hibernate?
14. What is Criteria API in Hibernate?
15. Why do we use HibernateTemplate?
16. How can you see SQL code generated by Hibernate on console?
17. What are the different types of collections supported by Hibernate?
18. What is the difference between session.save() and session.saveOrUpdate() methods in Hibernate?
19. What are the advantages of Hibernate framework over JDBC?
20. How can we get statistics of a SessionFactory in Hibernate?
21. What is the Transient state of an object in Hibernate?
22. What is the Detached state of an object in Hibernate?

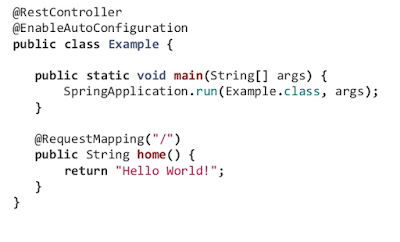
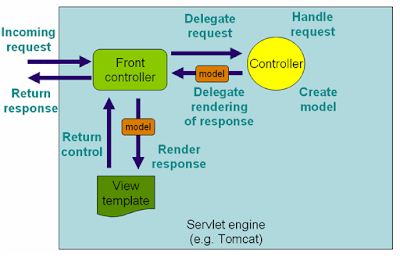
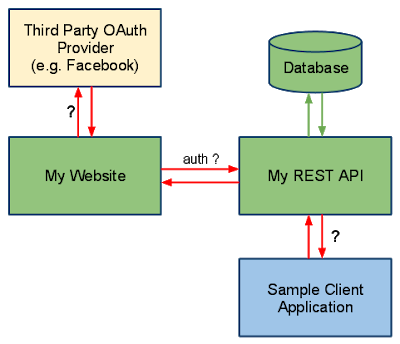


1. What is the use of Dirty Checking in Hibernate?
2. 
3. What is the purpose of Callback interface in Hibernate?
4. What are the different ORM levels in Hibernate?
5. What are the different ways to configure a Hibernate application?
6. What is Query Cache in Hibernate?

The Query Cache is disabled by default, and to activate it, we need to supply the following Hibernate property:

|  |  |
| --- | --- |
| 1  2 | properties.put("hibernate.cache.use\_query\_cache",      Boolean.TRUE.toString()); |

For Hibernate to cache a given query result, we need to explicitly set the [cachable query attribute](https://docs.jboss.org/hibernate/orm/4.3/javadocs/org/hibernate/Query.html#setCacheable%28boolean%29) when creating the Query.

1. What are the different types of Association mappings supported by Hibernate?
2. What are the different types of Unidirectional Association mappings in Hibernate?
3. What is Unit of Work design pattern?
4. In Hibernate, how can an object go in Detached state?
5. How will you order the results returned by a Criteria in Hibernate?
6. How does Example criterion work in Hibernate?
7. How does Transaction management work in Hibernate?
8. How can we mark an entity/collection as immutable in Hibernate?
9. What are the different options to retrieve an object from database in Hibernate?
10. How can we auto-generate primary key in Hibernate?
11. How will you re-attach an object in Detached state in Hibernate?
12. What is the first level of cache in Hibernate?
13. What are the different second level caches available in Hibernate?
14. Which is the default transaction factory in Hibernate?
15. What are the options to disable second level cache in Hibernate?
16. What are the different fetching strategies in Hibernate?
17. What is the difference between Immediate fetching and Lazy collection fetching?
18. What is ‘Extra lazy fetching’ in Hibernate?
19. How can we check is a collection is initialized or not under Lazy Initialization strategy?
20. What are the different strategies for cache mapping in Hibernate?
21. What is the difference between a Set and a Bag in Hibernate?
22. How can we monitor the performance of Hibernate in an application?
23. How can we check if an Object is in Persistent, Detached or Transient state in Hibernate?
24. What is ‘the inverse side of association’ in a mapping?
25. What is ORM metadata?
26. What is the difference between load() and get() method in Hibernate?
27. When should we use get() method or load() method in Hibernate?
28. What is a derived property in Hibernate?
29. How can we use Named Query in Hibernate?
30. What are the two locking strategies in Hibernate?
31. What is the use of version number in Hibernate?
32. What is the use of session.lock() method in Hibernate?
33. What inheritance mapping strategies are supported by Hibernate?
34. **What does REST stand for?**(answer)  
    REST stands for REpresentational State Transfer, which uses HTTP protocol to send data from client to server e.g. a book in the server can be delivered to the client using JSON or XML.  
      
      
    **What is a resource?**(answer)  
    A resource is how data is represented in REST architecture. By exposing entities as the resource it allows a client to read, write, modify, and create resources using HTTP methods e.g. [GET](http://javarevisited.blogspot.sg/2012/03/get-post-method-in-http-and-https.html), [POST](http://www.java67.com/2014/08/difference-between-post-and-get-request.html), [PUT](http://www.java67.com/2016/09/when-to-use-put-or-post-in-restful-web-services.html), DELETE etc.  
      
      
    **What are safe REST operations?**(answer)  
    REST API uses HTTP methods to perform operations. Some of the HTTP operations which doesn't modify the resource at the server is known as safe operations e.g. GET and HEAD. On the other hand, [PUT](http://javarevisited.blogspot.sg/2016/10/difference-between-put-and-post-in-restful-web-service.html), POST, and DELETE are unsafe because they modify the resource on the server.  
      
     **What are idempotent operations? Why is idempotency important?**([answer](http://javarevisited.blogspot.sg/2016/05/what-are-idempotent-and-safe-methods-of-HTTP-and-REST.html))  
    There are some HTTP methods e.g. GET which produce same response no matter how many times you use them e.g. sending multiple GET request to the same URI will result in same response without any side-effect hence it is known as idempotent.  
      
    On the other hand, the POST is not idempotent because if you send multiple POST request, it will result in multiple resource creation on the server, but again, PUT is idempotent if you are using it to update the resource.  
      
    Even, multiple PUT request to update a resource on a server will give same end result. You can further take [HTTP Fundamentals](http://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fxhttp-fund) course by Pluralsight to learn more about idempotent methods of HTTP protocol and HTTP in general.
35. **Is REST scalable and/or interoperable?**(answer)  
    Yes, REST is Scalable and interoperable. It doesn't mandate a specific choice of technology either at client or server end. You can use Java, C++, Python or JavaScript to create RESTful Web Services and Consume them at the client end. I suggest you read a good book on REST API e.g. [RESTful Web Services](http://javarevisited.blogspot.sg/2017/02/top-5-books-to-learn-rest-and-restful-web-services-in-java.html) to learn more about REST.  
      
      
    **What are the advantages of the RestTemplate?**([answer](http://javarevisited.blogspot.sg/2017/02/how-to-consume-json-from-restful-web-services-Spring-RESTTemplate-Example.html))  
    The RestTemplate class is an implementation of Template method pattern in Spring framework. Similar to other popular template classes e.g. JdbcTemplate or JmsTempalte, it also simplifies the interaction with RESTful Web Services on the client side. You can use it to consume a RESTful Web Servicer very easily as shown in this example.  
      
      
    **Which HTTP methods does REST use?**([answer](http://javarevisited.blogspot.sg/2016/04/what-is-purpose-of-http-request-types-in-RESTful-web-service.html#axzz56WGunSwy))  
    REST can use any HTTP methods but the most popular ones are GET for retrieving a resource, POST for creating a resource, PUt for updating resource and DELETE for removing a resource from the server.  
      
      
      
    **What is an HttpMessageConverter in Spring REST?**(answer)  
    An HttpMessageConverter is a [Strategy interface](http://www.java67.com/2014/12/strategy-pattern-in-java-with-example.html) that specifies a converter that can convert from and to HTTP requests and responses. Spring REST uses this interface to convert HTTP response to various formats e.g. JSON or XML.  
      
    Each HttpMessageConverter implementation has one or several MIME Types associated with it. Spring uses the "Accept" header to determine the content type client is expecting.  
      
    It will then try to find a registered HTTPMessageConverter that is capable of handling that specific content-type and use it to convert the response into that format before sending to the client.  
      
      
      
    **How to create a custom implementation of HttpMessageConverter to support a new type of request/responses?**(answer)  
    You just need to create an implementation of AbstractHttpMessageConverter and register it using the WebMvcConfigurerAdapter#extendMessageConverters() method with the classes which generate a new type of request/response.  
      
      
    **Is REST normally stateless?**([answer](http://javarevisited.blogspot.sg/2015/08/difference-between-soap-and-restfull-webservice-java.html))  
    Yes, REST API should be stateless because it is based on HTTP which is also stateless. A Request in REST API should contain all the details required it to process i.e. it should not rely on previous or next request or some data maintained at the server end e.g. Sessions. REST specification put a constraint to make it stateless and you should keep that in mind while designing your REST API.  
      
     **What does @RequestMapping annotation do?**([answer](http://javarevisited.blogspot.sg/2017/06/how-spring-mvc-framework-works-web-flow.html#axzz55vF5ugU8))  
    The @RequestMapping annotation is used to map web requests to Spring Controller methods. You can map request based upon HTTP methods  e.g. GET and POST and various other parameters. For examples, if you are developing RESTful Web Service using Spring then you can use produces and consumes property along with media type annotation to indicate that this method is only used to produce or consumers JSON as shown below:
36. @RequestMapping (**method** **=** RequestMethod.POST, consumes**=**"application/json")
37. **public** Book save(@RequestBody Book aBook) {
38. **return** bookRepository.save(aBook);
39. }
40. You can similarly create other handler methods to produce JSON or XML. If you are not familiar with these annotations then I suggest you join [**Spring MVC For Beginners**](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-mvc-tutorial-for-beginners-step-by-step%2F) course on Udemy to learn from scratch.
41. **Is @Controller a stereotype? Is @RestController a stereotype?**([answer](http://javarevisited.blogspot.sg/2017/08/difference-between-restcontroller-and-controller-annotations-spring-mvc-rest.html))  
    Yes, both @Controller and @RestController are stereotypes. The @Controller is actually a specialization of Spring's @Component stereotype annotation. This means that class annotated with @Controller will also be automatically be detected by Spring container as part of container's component scanning process.  
      
    And, @RestController is a specialization of @Controller for RESTful web service. It not only combines @ResponseBody and @Controller annotation but also gives more meaning to your controller class to clearly indicate that it deals with RESTful requests.  
      
    Spring Framework may also use this annotation to provide some more useful features related to REST API development in future.  
     **What is the difference between @Controller and @RestController?**([answer](http://javarevisited.blogspot.sg/2017/08/difference-between-restcontroller-and-controller-annotations-spring-mvc-rest.html))  
    There are many differences between @Controller and @RestController as discussed in my earlier article (see the answer) but the most important one is that with @RestControlleryou get the @ResponseBody annotation automatically, which means you don't need to separately annotate your handler methods with @ResponseBody annotation. This makes the development of RESTful web service easier using Spring. You can see here to learn
42. [](http://javarevisited.blogspot.sg/2016/12/top-5-spring-and-hibernate-training-courses-java-jee-programmers.html)
43. **When do you need @ResponseBody annotation in Spring MVC?**([answer](http://javarevisited.blogspot.sg/2018/01/7-reasons-for-using-spring-to-develop-RESTful-web-service.html#axzz55a8rTeu7))  
    The @ResponseBody annotation can be put on a method to indicates that the return type should be written directly to the HTTP response body (and not placed in a Model, or interpreted as a view name).  
      
    For example:  
      
    @RequestMapping(path = "/hello", method = RequestMethod.PUT)  
    @ResponseBody  
    public String helloWorld() {  
       return "Hello World";  
    }  
      
    Alternatively, you can also use @RestController annotation instead of @Controllerannotation. This will remove the need for using @ResponseBody because as discussed in the previous answer, it comes automatically with @RestController annotation.  
      
      
      
    **What does @PathVariable do in Spring MVC? Why it's useful in REST with Spring?**([answer](http://javarevisited.blogspot.sg/2017/10/differences-between-requestparam-and-pathvariable-annotations-spring-mvc.html))  
    It's one of the useful annotations from Spring MVC which allows you to read values from URI like query parameter. It's particularly useful in case of creating RESTful web service using Spring because in REST resource identifiers are part of URI.This questions is normally asked to experienced Spring MVC developers e.g. 4 to 6 years of experience.  
      
    For example, in the URL http://myapp.com/books/101 if you want to extract 101 the id, then you can use @PathVariable annotation of Spring MVC.  If you are not familiar with Spring MVC annotations then [Spring MVC For Beginners: Build Java Web App in 25 Steps](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-mvc-tutorial-for-beginners-step-by-step%2F) is a good place to start with.
44. [](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-mvc-tutorial-for-beginners-step-by-step%2F)
45. **What is the HTTP status return code for a successful DELETE statement?**([answer](http://www.java67.com/2015/09/top-10-restful-web-service-interview-questions-answers.html))  
    There is no strict rule with respect to what status code your REST API should return after a successful DELETE i.e it can return 200 Ok or 204 No Content. In general, if the DELETE operation is successful and the response body is empty return 204. If the DELETE request is successful and the response body is NOT empty, return 200  
      
      
    **What does CRUD mean?**(answer)  
    CRUD is a short form of Create, Read, Update and Delete. In REST API, the POST is used to create a resource, GET is used to read a resource, [PUT](http://javarevisited.blogspot.sg/2016/10/difference-between-put-and-post-in-restful-web-service.html) is used to updated a resource and DELETE is used to remove a resource from the server.This one is another beginner level Spring MVC questions for 1 to 3 years experienced programmers  
      
      
    **Where do you need @EnableWebMVC?**(answer)  
    The @EnableWebMvc annotation is required to enable Spring MVC when Java configuration is used to configure Spring MVC instead of XML. It is equivalent to <mvc: annotation-driven>  in XML configuration.  
      
    It enables support for @Controller-annotated classes that use @RequestMapping to map incoming requests to handler methods not already familiar with Spring's support for Java configuration, [Spring Master Class](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-tutorial-for-beginners%2F) on Udemy is a good place to start.
46. **When do you need @ResponseStatus annotation in Spring MVC?**([answer](http://javarevisited.blogspot.sg/2018/01/7-reasons-for-using-spring-to-develop-RESTful-web-service.html#axzz55a8rTeu7))
47. A good questions for 3 to 5 years experienced spring developers. The @ResponseStatus annotation is required during error handling in Spring MVC and REST. Normally when an error or exception is thrown at server side, web server return a blanket HTTP status code 500 - Internal server error.  
      
    This may work for a human user but not for REST clients. You need to send them proper status code e.g. 404 if the resource is not found. That's where you can use @ResponseStatusannotation, which allows you to send custom HTTP status code along with proper error message in case of Exception.  
      
    In order to use it, you can create custom exceptions and annotated them using @ResponseStatus annotation and proper HTTP status code and reason.  
      
    When such exceptions are thrown from controller's handler methods and not handled anywhere else, then appropriate HTTP response with the proper HTTP status code, which you have set is sent to the client.  
      
    For example, if you are writing a RESTful Web Service for a library which provides book information then you can use @ResponseStatus to create Exception which returns HTTP response code 404 when a book is not found instead of Internal Server Error (500), as shown below:  
      
     @ResponseStatus(value=HttpStatus.NOT\_FOUND, reason="No such Book")  // 404  
     public class BookNotFoundException extends RuntimeException {  
         // ...  
     }  
      
    If this Exception is thrown from any handler method then HTTP error code 404 with reason "No such Book" will be returned to the client.  
      
      
    **Is REST secure? What can you do to secure it?**([answer](http://www.java67.com/2017/04/3-great-books-to-learn-java-web-services-soap-and-restful.html))  
    This question is mostly asked with experienced Java programmers e.g. 2 to 5 years experience with both REST and Spring. Security is a broad term, it could mean security of message which is provided by encryption or access restriction which is provided using authentication and authorization. REST is normally not secure but you can secure it by using Spring security.  
      
    At the very least you can enable HTTP basic authentication by using HTTP in your Spring security configuration file. Similarly, you can expose your REST API using[HTTPS](http://javarevisited.blogspot.sg/2013/07/how-ssl-https-and-certificates-works-in-java-web-application.html) if the underlying server supports HTTPS.
48. [](http://javarevisited.blogspot.sg/2018/01/how-to-enable-http-basic-authentication-spring-security-java-xml-configuration.html)
49. **Does REST work with transport layer security (TLS)?**([answer](http://javarevisited.blogspot.sg/2012/01/rest-web-services-framework-interview.html))  
    TLS or Transport Layer Security is used for secure communication between client and server. It is the successor of SSL (Secure Socket Layer). Since HTTPS can work with both SSL and TLS, REST can also work with TLS.  
      
    Actually, REST says anything about Security, it's up to the server which implements that. Same RESTful Web Service can be accessed using HTTP and HTTPS if the server supports [SSL](http://javarevisited.blogspot.sg/2013/07/how-to-configure-https-ssl-in-tomcat-6-7-web-server-java.html#axzz56WXxxAC0).  
      
    If you are using Tomcat, you can see here to learn more about how to enable SSL in Tomcat.  
      
      
    **Do you need Spring MVC in your classpath for developing RESTful Web Service?**([answer](http://javarevisited.blogspot.sg/2017/01/where-and-how-to-download-spring-JAR-Files-Spring4-without-Maven-Gradle.html#axzz4pp42TeHu))  
    This question is often asked to Java programmers with 1 to 2 years of experience in Spring. Short answer is Yes, you need Spring MVC in your Java application's classpath to develop RESTful web services using Spring framework. It's actually Spring MVC which provides all useful annotations e.g. @RestController, @ResponseCode, @ResponseBody, @RequestBody, and @PathVariable, hence you must spring-mvc.jar or appropriate Maven entry in your pom.xml  
      
      
    That's all about some **frequently asked Spring REST Interview questions** for beginners and experienced Java JEE developers. These questions are also very useful to brush up your knowledge about Spring REST if you are going to take Spring Certification. If you need more questions from Spring certification perspective, you will find a lot of question on this topic on David Mayer's [Core Spring Simulator](https://www.certification-questions.com/spring-free-mock-exams/spring-core-v4.2-practice-test.html?affiliateCode=fcff36fd-557a-4713-abf6-973e9924770f&utm_source=Javin&utm_medium=affiliate&utm_campaign=affiliate), one of the best simulator to pass Spring certification at the moment.