1. How can we use spring to create restfull webservices returning json response?

In spring boot we don't need to put dependency for converting pojo to json..

but in spring we need to put dependency.

[{"id":1,"firstname":indu,"lastname":mukhi,"username":indu,"password":indu…}]

Spring's annotation-based MVC framework simplifies the process of creating RESTful web services. The key difference between a traditional Spring MVC controller and the RESTful web service controller is the way the HTTP response body is created. While the traditional MVC controller relies on the View technology, the RESTful web service controller simply returns the object and the object data is written directly to the HTTP response as JSON/XML.

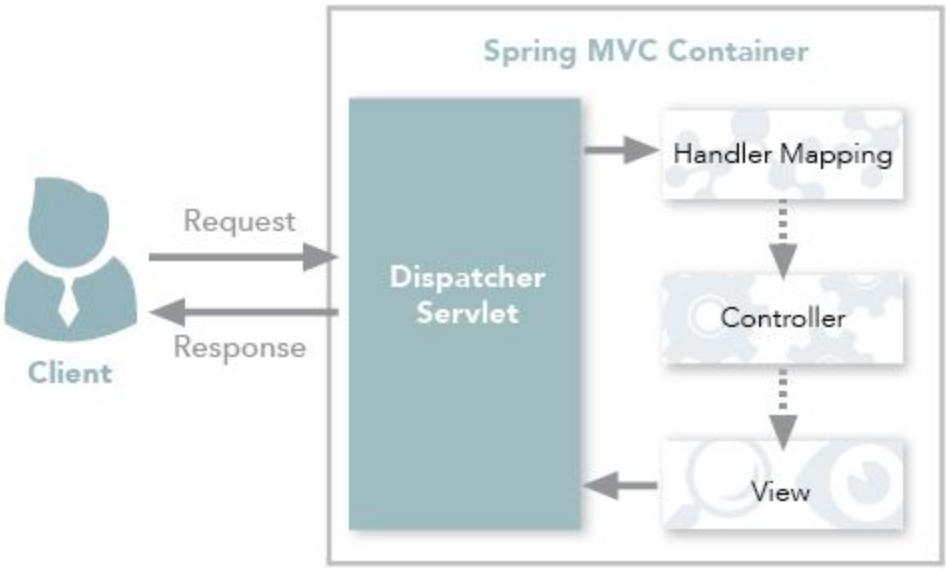


Figure 1: Spring MVC traditional workflow

Using the @ResponseBody Annotation

When you use the @ResponseBody annotation on a method, Spring converts the return value and writes it to the http response automatically. Each method in the Controller class must be annotated with @ResponseBody.

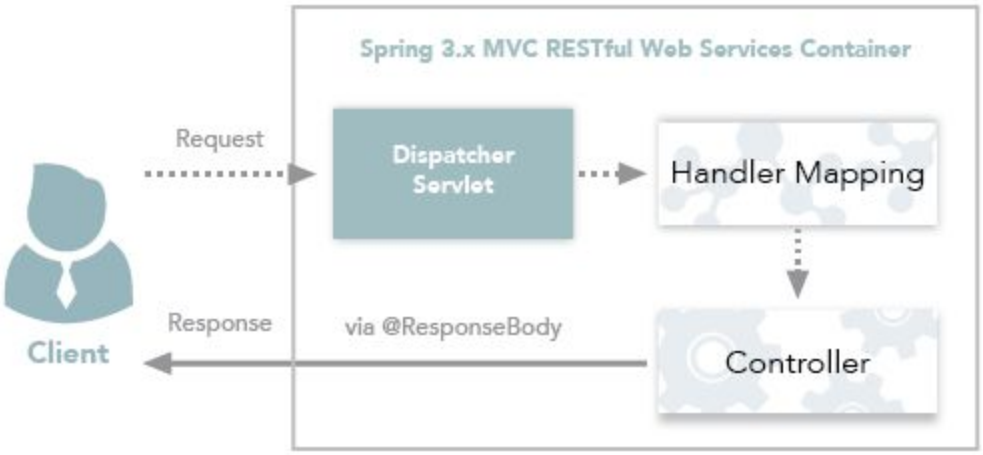


Figure 2: Spring 3.x MVC RESTful web services workflow

2. What is Rest Controller?

Spring MVC : is a **M**odel-**V**iew-**C**ontroller framework to build web application using different view technologies like JSP, JSTL, Tiles, etc

Spring Restful : is also a **M**odel-**V**iew-**C**ontroller framework but adding few annotations like @RestController, @RequestBody, @ResponseBody turns into REST web service

To **conclude**, Spring Restful service has both Spring MVC and Rest web service capabilities

1. The @Controller is a common annotation which is used to mark a class as Spring MVC Controller while @RestController is a special controller used in RESTFul web services and the equivalent of @Controller + @ResponseBody.

2. The **@RestController** is relatively new, added only on Spring 4.0 but @Controller is an old annotation, exists since Spring started supporting annotation, officially it was added on Spring 2.5 version.

3. The @Controller annotation indicates that the class is a "Controller" e.g. a web controller while @RestController annotation indicates that the class is a controller where @RequestMapping methods assume **@ResponseBody** semantics by default i.e. servicing REST API.

4.One of the key difference between @Controler and @RestCotroller in Spring MVC is that once you mark a class @RestController then every method is written a domain object instead of a view.

To return an java object in JSON form from an spring objects requires two simple configurations:
1) Adding 'jackson-mapper-asl' dependency to the classpath
2) Add @ResponseBody annotation to the controller's method

2. How form data bind into controller?

@ModelAttribute

This annotation can be used as the method arguments or before the method declaration. The primary objective of this annotation to bind the request parameters or form fields to an model object

The model object can be formed using the request parameters or already stored in the session object. Note that, this @ModelAttribute methods are invoked before the controller methods with @RequestMapping are invoked. The logic behind the sequence is that, the model object has to be created before any processing starts inside the controller methods.

3. What r d diff bet model attribute ,reqparam and pathvariable?

RequestParam annotation used for accessing the query parameter values from the request.

http://localhost:8080/springmvc/hello/101?param1=10¶m2=20

```
public String getDetails(
        @RequestParam(value="param1", required=true) String param1,
        @RequestParam(value="param2", required=false) String param2){
...
}
```

http://localhost:8080/springmvc/hello/101?param1=10&param2=20

The above URL request can be written in your Spring MVC as below:

```
@RequestMapping("/hello/{id}")
        public String getDetails(@PathVariable(value="id") String id,
        @RequestParam(value="param1", required=true) String param1,
        @RequestParam(value="param2", required=false) String param2){
.......
}
```

@PathVariable identifies the pattern that is used in the URI for the incoming request.

4. What is front controller and explain how does it internally works?or

explaining the flow of web request i.e. how an HTTP request is processed from start to end. In other words, explaining the **flow of request in Spring MVC**.
Refer notes

5. How to read data from external properties file?

Assign the property values to fields by using @Value with PropertySourcesPlaceholderConfigurer to resolve ${} in @Value:

Ex:

```
@Value(" ${linksharing.save.message}")

private String message;



@Value("${linksharing.save.error}")

private String error;
```

In Apllication.properties file:

```
linksharing.login.success=successfuly login

linksharing.login.error=Invalid Username and password

linksharing.login.message=User Already logined

linksharing.save.error=Record not Saved

linksharing.save.message=Record Saved Successfully
```

6. How to Define Association Mappings between Entities

The main advantage of putting relation ship between objects is, we can do operation on one object, and the same operation can transfer onto the other object in the database [remember, object means one row in hibernate terminology

Using hibernate we can put the following 4 types of relationships

- One-To-One
- Many-To-One
- Many-To-Many
- One-To-Many

7. **Hibernate Inheritance Mapping**

We can map the inheritance hierarchy classes with the table of the database. There are three inheritance mapping strategies defined in the hibernate:

1. Table Per Hierarchy

```
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)

@DiscriminatorColumn(discriminatorType = DiscriminatorType.STRING, name = "Type")

public abstract class Resource extends BaseEntity {}

@DiscriminatorValue("DocumentResource")

public class DocumentResource extends Resource {


@DiscriminatorValue("LinkResource")

public class LinkResource extends Resource {
```

It will add all columns in single table.

1. Table Per Concrete class
2. Table Per Subclass

8. How many Ways of creating Bean?

9. what is bean in spring?

10 . what does it mean @AutoWired?

11. Diff bet @AutoWired and @Qualifier?

12.what is Stereotype Annotations?

13. if we want to write Bussiness Logic where we will Write?
Service Layer

14. what is Repository?

15. To create Repository we need Interface r class?
Interface

16. what is @mappedSuperClass ?

17. how to differentiate column in tableper hirerchy?”
DiscriminatorColumn(name=””)

18.how many tables are created in one to many?
We can do 2 r 3 tables,3rd table is Mapping table.

19. Diff between @basic and @Column?

20. what does it mean by EAGER and LAZY loading?

21. How to use LAZY loading?

22. @OneToOne,@OneToMany relationships?

23. Diff bet Save ,save&flush?

24. Difference between @NotNull, @NotEmpty and @NotBlank.
@NotNull-----CharSequence, Collection, Map or Array object cannot be null, however can be empty.
@NotEmpty------The CharSequence, Collection, Map or Array object cannot be null and not empty (size > 0).
@NotBlank-------The string is not null and the length is greater than zero

25. diff between get,load and read method of hibernate?

session.load()

- It will always return a "**proxy**" (Hibernate term) without hitting the database. In Hibernate, proxy is an object with the given identifier value, its properties are not initialized yet, it just look like a temporary fake object.
- If no row found , it will throws an **ObjectNotFoundException**.

session.get()

- It always **hit the database** and return the real object, an object that represent the database row, not proxy.
- If no row found , it return **null**.

26. diff between save and saveorupdate methos in hibernate?
save() method saves records into database by INSERT SQL query, Generates a new identifier and return the Serializable identifier back.

On the other hand saveOrUpdate() method either INSERT or UPDATE based upon existence of object in database. If persistence object already exists in database then UPDATE SQL will execute and if there is no corresponding object in database than INSERT will run.

27.what are sessions in hibernate?

A Session is used to get a physical connection with a database.

Session is a common interface and a way of communication between java application and Hibernate ORM that can have one or many transactions. basically a transaction is a single atomic operation that may be insert, update, delete. in case of interupt the transaction is roll back.

28. how to get session factory object in hibernate?

29. there is one case in which u dnt want to persistent field of class in table so how to do?

30.what are the objects states in hibernate?

31. diff bet detach and persistent state of object?

32. how to write query in hibernate?

33. diff bet jpa repository and crud repository?

34. diff between merge and update method of hibernate?

35. diff bet merge save update methods?

36. how to set attribute and get attribute and remove attributes?

37.How to get http session obj in spring boot?

38. how to get httpsession in service layer with out passing httpsession from controller?

39. how to call init method of bean during creating bean?

40. what are diff scopes of bean?

41. diff bet request and prototype scope?

42.injecting a prototype bean into a singleton bean problem… why its not working as expected?

43.how to resolve cyclic bean dependency problem in spring?

44. what are the most common spring exception which u faced?

45.what is configuration annotation and import annotation?

46. What is autowire by name, type,no?

47.how to inject bean using name and type?

48. Diff bet qualifier and autowire annotation?

49.what are diff ways of injecting dependencies?
By constructor based
Setter based
Field based

50. What is lazy annotation?

51.How to interact two mocroservices?

52.what is @produces,@consumes?

53.what is proxy?
54. **What is DTO, DAO, and VO in Java?**
DTO: Data transfer objects are just data containers which are used to transport data between layers and tiers.

- It mainly contains attributes. You can even use public attributes without getters and setters.

- Data transfer objects do not contain any business logic.

- DTO was mainly used to get data transported across the network efficiently. It may be even from JVM to another JVM.

Data transfer objects " can travel between seperate layers in software architecture
It just encapsulates the data and transfer between layers (from persistence (DB) to Business) or network.

VO: "Value objects " hold a object such as Integer, City, etc.

DAO: The Data Access Object is basically an object or an interface that provides access to an underlying database or any other persistence storage.

55. what is Spring Ioc container== Spring Container?

The Spring IoC Container creates, configures and connects the objects (which are called Beans), then manage their life cycle. The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

to instantiate the application class to configure the object to assemble the dependencies between the objects There are two types of IoC containers. They are:

1)BeanFactory 2) ApplicationContext

Difference between Bean Factory and Application Context.

The ApplicationContext interface is built on top of the BeanFactory interface. It adds some extra functionality than BeanFactory such as simple integration with Spring's AOP, message resource handling (for I18N), event propagation, application layer specific context (e.g. WebApplicationContext) for web application. So it is better to use ApplicationContext than BeanFactory.

Using BeanFactory

The XmlBeanFactory is the implementation class for the BeanFactory interface. To use the BeanFactory, we need to create the instance of XmlBeanFactory class as given below:

```
1.  Resource resource=new ClassPathResource("applicationContext.xml");

2.  BeanFactory factory=new XmlBeanFactory(resource);
```

The constructor of XmlBeanFactory class receives the Resource object so we need to pass the resource object to create the object of BeanFactory.

Using ApplicationContext

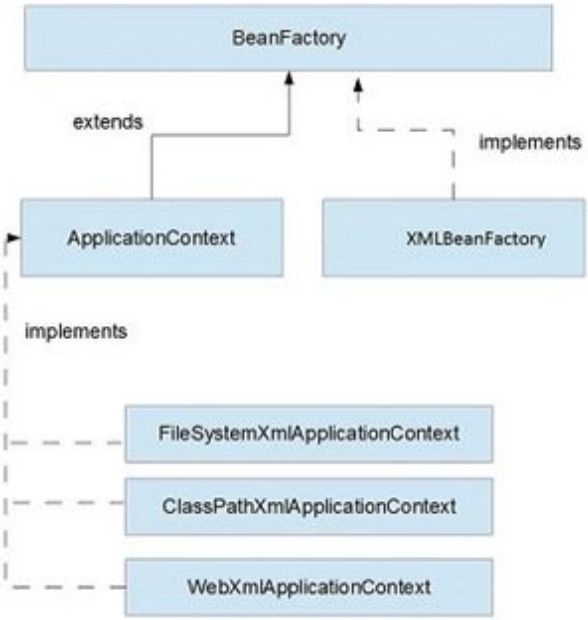
The ClassPathXmlApplicationContext class is the implementation class of ApplicationContext interface. We need to instantiate the ClassPathXmlApplicationContext class to use the ApplicationContext as given below:

```
ApplicationContext context =

   new ClassPathXmlApplicationContext("applicationContext.xml");
```

The constructor of ClassPathXmlApplicationContext class receives string, so we can pass the name of the xml file to create the instance of ApplicationContext.

	XMLBeanFactory	ApplicationContext
Annotation support	No	Yes
BeanPostProcessor Registration	Manual	Automatic
implimentation	XMLBeanFactory	ClassPath/FileSystem/WebXmlAppli
internationalization	No	Yes
Enterprise services	No	Yes
ApplicationEvent publication	No	Yes



56.how to get bean from application context?

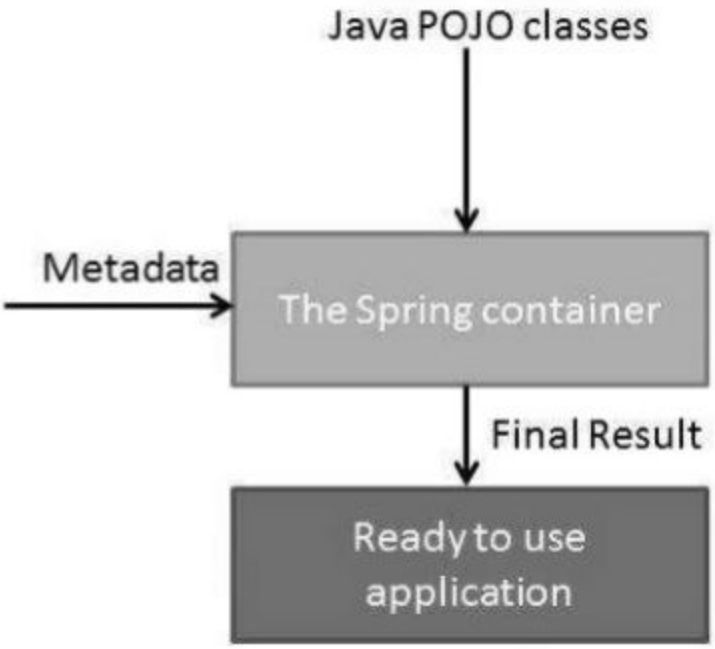
```
ApplicationContext context =
    new ClassPathXmlApplicationContext("applicationContext.xml");
```

57.what are implementation clas of application context?

FileSystemXmlApplicationContext
ClasspathXmlApplicationContext
WebXmlApplicationContext

58.How does ioc Container internally works?

The Spring container uses DI to manage the components that make up an application. These objects are called Spring Beans



The container gets its instructions on what objects to instantiate, configure, and assemble by reading the configuration metadata provided. The configuration metadata can be represented either by XML, Java annotations, or Java code. The following diagram represents a high-level view of how Spring works. The Spring IoC container makes use of Java POJO classes and configuration metadata to produce a fully configured and executable system or application.

59. How many ways to get bean from ioc container?
60. How to load Xml in spring container?

Bean.xml

```
<bean name="userService" class="com.sagarandcompany.IocContainer.UserService">
      <property name="name" value="Sagar"/>
      <property name="email" value="Sagarmal624@gmail.com"/>
      <property name="age" value="25"/>
  </bean>
</beans>
```

Java class

```
public class IocContainerApplication {

  public static void main(String[] args) {
```

```
//      SpringApplication.run(IocContainerApplication.class, args);

//                                  FileSystemXmlApplicationContext context = new
FileSystemXmlApplicationContext("E:/STS/WorkPlace/SpringIoC/src/main/resources/bean.xml")
;

//      Resource resource = new ClassPathResource("bean.xml");
//
//      BeanFactory factory = new XmlBeanFactory(resource);

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

    UserService userService = (UserService) context.getBean("userService");
    System.out.println(userService.toString());
  }
```

61. What is application context?

ApplicationContext is an interface for providing configuration information to an application. There are multiple classes provided by springframework that implements this interface and helps us use configuration information in applications. ApplicationContext provides standard bean factory lifecycle capabilities.

62. How do I copy properties from one bean to another?

copying properties from one bean to another.

```
public class BeanUtilsCopyPropertiesTest {

        public static void main(String[] args) {

                FromBean fromBean = new FromBean("fromBean", "fromBeanAProp",
"fromBeanBProp");
                ToBean toBean = new ToBean("toBean", "toBeanBProp", "toBeanCProp");
                System.out.println(ToStringBuilder.reflectionToString(fromBean));
                System.out.println(ToStringBuilder.reflectionToString(toBean));
                try {
                System.out.println("Copying properties from fromBean to toBean");
                BeanUtils.copyProperties(toBean, fromBean);
```

63. Diff between abstract and inheritance?

For inheritance class we cant define method;
For abstract class we can define implement in other classes,with same name and diff implementation.
In abstract class have abstract and non abstract methods..

In inheritance if can define new methods but we cant implement that in new class with same name..

64.diff between @ResponseBody and @RequestBody?