# Web services

How to access SOAP web service?

There are two ways to access web service

**1. If Service provider knows client**

**If Service provider knows client,** then it will provide its wsdl to client and client will be able to access web service.



**2. If Service provider register its WSDL to UDDI and client can access it from UDDI**

**UDDI**:UDDI stands for Universal Description, Discovery and Integration. It is a directory service. Web services can register with a UDDI and make themselves available through it for discovery.So following steps are involved.

***1. Service provider registers with UDDI.***

***2. Client searches for service in UDDI.***

***3. UDDI returns all service providers offering that service.***

***4. Client chooses service provider***

***5. UDDI returns WSDL of chosen service provider.***

***6. Using WSDL of service provider, client accesses web service***



What are Rest components

It consists of two components

1. **REST server:** which provides access to the resources

2. **REST client** : which accesses and modify the REST resources.



What is Idempotent?

Idempotent means result of multiple successful request will not change state of resource after initial application

**For example:**

**GET is idempotent.** If Delete() is idempotent method because when you first time use delete, it will delete the resource (initial application) but after that, all other request will have no result because resource is already deleted.

**Post is not idempotent** method because when you use post to create resource, it will keep creating resource for each new request, so result of multiple successful request will not be same.

Webservices API in java?



JAX-WS Encoding Styles?

There are two encoding use models that are used to translate a WSDL binding to a SOAP message. They are: **literal, and encoded.**

The combination of the different style and use models give us four different ways to translate a WSDL binding to a SOAP message.

Document/literal

Document/encoded

RPC/literal

RPC/encoded

**When using a literal use model**, the body contents should conform to a user-defined **XML-schema (XSD) structure**. The advantage is two-fold.

* one, you can validate the message body with the user-defined XML-schema.
* Two, you can also transform the message using a transformation language like XSLT.

**With a (SOAP) encoded use model**, the message has to use XSD datatypes, but the structure of the message need not conform to any user-defined XML schema. This makes it difficult to validate the message body or use XSLT based transformations on the message body.

Diffrence between RPC-Style and Document Style

The way of generating SOAP message format is main difference between them.

**1. RPC Stlye**:

SOAP Body must conform to a structure that indicates the **method name & Parameters name**

<soap:envelope>

<soap:body>

<myMethod>

<x xsi:type="xsd:int">5</x>

<y xsi:type="xsd:float">5.0</y>

</myMethod>

</soap:body>

</soap:envelope>

**2. Document Style**

SOAP Body can be structured in any way you like. There is no TYPE attribute here

<soap:envelope>

<soap:body>

<xElement>5</xElement>

<yElement>5.0</yElement>

</soap:body>

</soap:envelope>

Steps to create JAX-WS Webservice

**1. JAX-WS Web Service End Point files**

1. Create a Web Service Endpoint Interface with **@SOAPBinding(style = Style.RPC)**
2. Create a Web Service Endpoint Implementation
3. Create an Endpoint Publisher
4. Test generated WSDL. Ex: **http://localhost:8080/ws/hello?wsdl**

**2. Web Service Client files**

1. Java Web Service Client

**1. JAX-WS Web Service End Point files**

**1. Create a Web Service Endpoint Interface**

package endpoint;

import javax.jws.WebMethod;

import javax.jws.WebService;

import javax.jws.soap.SOAPBinding;

import javax.jws.soap.SOAPBinding.Style;

//Service Endpoint Interface

@WebService

@SOAPBinding(style = Style.*RPC*)

public interface HelloWorld{

@WebMethod

String getHelloWorldMsg(String msg);

}

**2. Create a Web Service Endpoint Implementation**

package endpoint;

import javax.jws.WebService;

//Service Implementation

@WebService(endpointInterface = "endpoint.HelloWorld")

public class HelloWorldImpl implements HelloWorld{

@Override

public String getHelloWorldMsg(String msg) {

// TODO Auto-generated method stub

return "Your Message from WebService is : "+msg;

}

}

**3. Create an Endpoint Publisher**

package endpoint;

import javax.xml.ws.Endpoint;

//Endpoint publisher

public class HelloWorldPublisher{

public static void main(String[] args) {

Endpoint.*publish*("http://localhost:7777/ws/hello", new HelloWorldImpl());

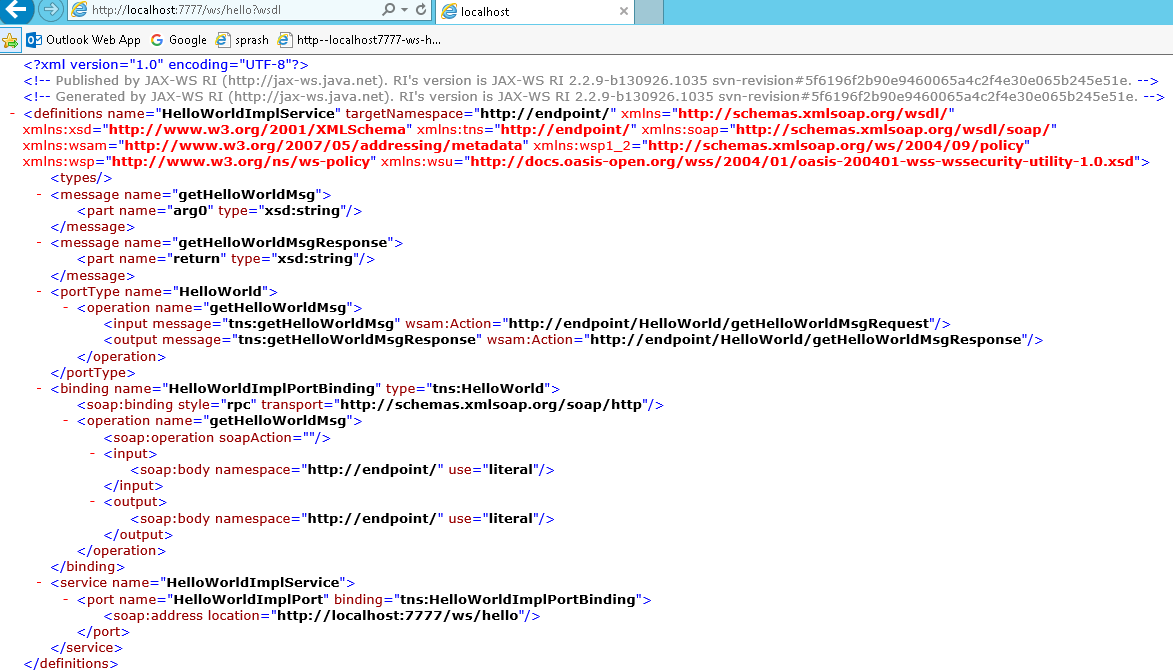
System.*out*.println("WSDL Published !!");

}

}

**4. Test generated WSDL**

Run HelloWorldPublisher as Java Application & access url: [***http://localhost:7777/ws/hello?wsdl***](http://localhost:7777/ws/hello?wsdl)

****

<http://endpoint/>" uses package name of Service endpoint publisher

wsimport tool VS wsgen

**1.wsimport –(WSDL Import)** tool is will import WSDL file and generates JAX-WS Web Service End Point files.

>wsimport -keep http://localhost:7777/ws/hello?wsdl

**2.wsGen –(WSDL Generator)**

It will read the JAX-WS Web Service End Point files & Generates WSDL Document & Webservice client for Testing . This wsgen tool is available in $JDK/bin folder

**>wsgen -verbose -keep -cp . endpoint.RandomNumber**

Difference between JAX-RS & RESTful

* RESTFul is a Generalized Web service Standard given by W3.ORG.
* JAX-RS is a specification for RESTful Web Services with Java and it is given by Sun.
* **Jersey** from Oracle, **Resteasy** from Jboss are the implementations of JAX-RS

majorly used annotations in RESTFul webservices

* **@Path(‘Path‘)**
* **@GET**
* **@POST**
* **@PUT**
* **@DELETE**
* **@Produces(MediaType.TEXT\_PLAIN [, more-types]) – Only for @GET**
* **@Consumes(type[, more-types]) - Only for @POST**
* **@PathParam()**
* **@QueryParam()**
* **@MatrixParam()**
* **@FormParam()**

Steps to creates to RestFul web-service in java?

1 Add Jersey jar files in pom.xml

2. Create RESTFul webservice at Server End.

@Path("/hellojersey")

public class HelloWorldWebService {

// This method is called if HTML and XML is not requested

@GET

@Produces(MediaType.*TEXT\_PLAIN*)

public String sayPlainTextHello() {

return "Hello Jersey Plain";

}

// This method is called if HTML is requested

@GET

@Produces(MediaType.*TEXT\_HTML*)

public String sayHtmlHello() {

return "<h1>" + "Hello Jersey HTML" + "</h1>";

}

}

**3.Configure web.xml**

In web.xml, register “com.sun.jersey.spi.container.servlet.ServletContainer“, and puts your Jersey service folder under “**init-param**“, “com.sun.jersey.config.property.packages

<web-app>

<servlet>

<servlet-name>jersey-serlvet</servlet-name>

<servlet-class>

com.sun.jersey.spi.container.servlet.ServletContainer

</servlet-class>

<init-param>

<param-name>com.sun.jersey.config.property.packages</param-name>

<param-value>service</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>jersey-serlvet</servlet-name>

<url-pattern>/rest/\*</url-pattern>

</servlet-mapping>

</web-app>

4.Test Service

[**http://localhost:8080/JAXRS-Jersey-HelloWorld/rest/hellojersey**](http://localhost:8080/JAXRS-Jersey-HelloWorld/rest/hellojersey)

or

System.*out*.println(target.path("rest").path("hellojersey").request().accept(MediaType.*TEXT\_PLAIN*).get(String.class));

System.*out*.println(target.path("rest").path("hellojersey").request().accept(MediaType.*TEXT\_HTML*).get(String.class));

}

Examples

[1.@Path](mailto:1.@Path) Annotation

@Path("/country")

public class PathMethodLevelService {

@GET

@Produces("text/html")

public Response selectCountry() {

String output = " Default Country : <h1>INDIA</h1>";

return Response.*status*(200).entity(output).build();

}

@GET

@Path("/usa")

@Produces("text/html")

public Response selectUSA() {

String output = "Selected Country : <h1>United States of America(USA)</h1>";

return Response.*status*(200).entity(output).build();

}

@GET

@Path("/uk")

@Produces("text/html")

public Response selectUK() {

String output = "Selected Country : <h1>UNITED KINGDOM(UK)</h1>";

return Response.*status*(200).entity(output).build();

}

}

Response Class in JAX-RS

**javax.ws.rs.core.Response** contains static methods to create a Response instance using a **ResponseBuilder**.

**http://localhost:8080/App/rest/students/101/Satya/Vijayawada**

@Path("/students")

public class PathParamService {

@GET

@Path("{rollno}/{name}/{address}")

@Produces("text/html")

public Response getResultByPassingValue(

@PathParam("rollno") String rollno,

@PathParam("name") String name,

@PathParam("address") String address) {

String output = "<h1>PathParamService Example</h1>";

output = output+"<br>Roll No : "+rollno;

output = output+"<br>Name : "+name;

output = output+"<br>Address : "+address;

return Response.*status*(200).entity(output).build();

}

}

How to set different status code in HTTP response?

For setting HTTP status code other than 200, we have to use javax.ws.rs.core.Response class for response. Below are some of the sample return statements showing it’s usage.

return Response.status(422).entity(exception).build();

return Response.ok(response).build(); //200



**http://localhost:8080/App/rest/students?rollno=1218&name=SATYA &address=VIJAYAWADA**

@Path("/students")

public class QueryParamwithDefaultvalueService {

@GET

@Produces("text/html")

public Response getResultByPassingValue(

@DefaultValue("1000") @QueryParam("rollno") String rollno,

@DefaultValue("XXXX") @QueryParam("name") String name,

@DefaultValue("XXXX") @QueryParam("address") String address) {

String output = "<h1>QueryParamwithDefaultvalueService Example</h1>";

output = output + "<br>Roll No : " + rollno;

output = output + "<br>Name : " + name;

output = output + "<br>Address : " + address;

return Response.*status*(200).entity(output).build();

}

}

**http://localhost:8080/App/rest/students;rollno=1118;name=SATYA;address=VIJAYAWADA**

@Path("/students")

public class MatrixParamService{

@GET

@Produces("text/html")

public Response getResultByPassingValue(

@MatrixParam("rollno") String rollno,

@MatrixParam("name") String name,

@MatrixParam("address") String address) {

String output = "<h1>@MatrixParam Example</h1>";

output = output+"<br>Roll No : "+rollno;

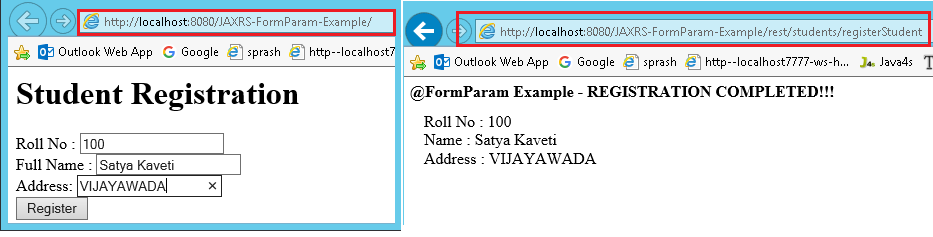
output = output+"<br>Name : "+name;

output = output+"<br>Address : "+address;

return Response.*status*(200).entity(output).build();

}

}



@Path("/students")

public class FormParamService {

@POST

@Path("/registerStudent")

@Produces("text/html")

public Response getResultByPassingValue(

@FormParam("rollno") String rollno,

@FormParam("name") String name,

@FormParam("address") String address) {

String output = "<h1>@FormParam Example - REGISTRATION COMPLETED!!!</h1>";

output = output+"<br>Roll No : "+rollno;

output = output+"<br>Name : "+name;

output = output+"<br>Address : "+address;

return Response.*status*(200).entity(output).build();

}

}

JAX-RS Download files (text/image/pdf/execel) Example

We can download any type of files from the RESTful web services, **@produces** annotation

We should annotate our method with

* **@Produces(“text/plain“)** If you are expecting Text file as response
* **@Produces(“image/your image type[.jpg/.png/.gif]”)** for downloading any Image files
* **@Produces(“application/pdf“)** for downloading PDF files
* @Produces**(MediaType.APPLICATION\_JSON**) -- JSON
* @Produces(MediaType.APPLICATION\_XML). -XML

How to Test (JAX-RS) RESTful Web Services

in real time projects we will use different tools to test RESTful web services

**1.Browser Addons**

* Postman [ Chrome Extension ]
* REST Client [ Chrome Extension ]
* Advanced REST Client [ Chrome Extension ]
* Rest Client [ Firefox Add-On ]

**2.JAX-RS Local System Tools**

* [RESTClient UI](https://code.google.com/archive/p/rest-client/downloads)
* [SoupUi](https://www.soapui.org/)

What are advantages of SOAP Web Services?

SOAP web services have all the advantages that web services has, some of the additional advantages are:

* WSDL document provides contract and technical details of the web services for client applications without exposing the underlying implementation technologies.
* SOAP uses XML data for payload as well as contract, so it can be easily read by any technology.
* SOAP protocol is universally accepted, so it’s an industry standard approach with many easily available open source implementations.

What are different components of WSDL?

Some of the different tags in WSDL xml are:

* xsd:import namespace and schemaLocation: provides WSDL URL and unique namespace for web service.
* message: for method arguments
* part: for method argument name and type
* portType: service name, there can be multiple services in a wsdl document.
* operation: contains method name
* soap:address for endpoint URL.

What is difference between Top Down and Bottom Up approach in SOAP Web Services?

**In Top Down approach first WSDL document is created to establish the contract between web service** and client and then code is written, it’s also termed as contract first approach. This is hard to implement because classes need to be written to confirm the contract established in WSDL. Benefit of this approach is that both client and server code can be written in parallel.

**In Bottom Up approach, first web service code is written and then WSDL is generated**. It’s also termed as contract last approach. This approach is easy to implement because WSDL is generated based on code. In this approach client code have to wait for WSDL from server side to start their work.

Can we maintain user session in web services?

Web services are stateless so we can’t maintain user sessions in web services.

What is difference between SOA and Web Services?

Service Oriented Architecture (SOA) is an architectural pattern where applications are designed in terms of services that can be accessed through communication protocol over network. SOA is a [design pattern](https://www.journaldev.com/1827/java-design-patterns-example-tutorial) and doesn’t go into implementation.

Web Services can be thought of as Services in SOAP architecture and providing means to implement SOA pattern.

Name some frameworks in Java to implement SOAP web services?

We can create SOAP web services using JAX-WS API, however some of the other frameworks that can be used are Apache Axis and Apache CXF.

What is use of javax.xml.ws.Endpoint class?

Endpoint class provides useful methods to create endpoint and publish existing implementation as web service. This comes handy in testing web services before making further changes to deploy it on actual server.

What is sun-jaxws.xml file?

This file is used to provide endpoints details when JAX-WS web services are deployed in servlet container such as Tomcat. This file is present in WEB-INF directory and contains endpoint name, implementation class and URL pattern. For example;

**sun-jaxws.xml**

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

<endpoints xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime" version="2.0">

<endpoint

name="PersonServiceImpl"

implementation="com.journaldev.jaxws.service.PersonServiceImpl"

url-pattern="/personWS"/>

</endpoints>

Name important annotations used in JAX-RS API?

Some of the important JAX-RS annotations are:

* @Path: used to specify the relative path of class and methods. We can get the URI of a webservice by scanning the Path annotation value.
* @GET, @PUT, @POST, @DELETE and @HEAD: used to specify the HTTP request type for a method.
* @Produces, @Consumes: used to specify the request and response types.
* @PathParam: used to bind the method parameter to path value by parsing it.

What is purpose of different HTTP Request Types in RESTful Web Service?

* **GET** request on /employee/101, you can retrieve details of that user.
* **POST** on employe/102 would create a new user with employee id 102,
* **PUT** request type on /employee/101 can be used to update details of employee with id 101.
* **DELETE** method on /employee/101 can be used to remove data for that id.

By the way, in the case of PUT and POST method representation would be in the **request body**

# Spring

## Spring Core

 List out the new features available in Spring 4.0 and Spring 5.0?

* Spring 2.5 **Annotations/Autowire**
* Spring 3.0 **Java Configuration**
* Spring 4.0 is the first to support **Java 8 features**.
* Spring 5.0 has the support for **Reactive Programming**

What is the default scope of bean in Spring framework? ([answer](http://javarevisited.blogspot.sg/2012/05/what-is-bean-scope-in-spring-mvc.html))

The default scope of a **Spring bean** is the Singleton scope and in the **WebApplication** default scope of a spring bean is request scope.

Singleton bean means the same instance of a bean is shared with all other beans, while request scope means a bean is alive only for a request.

Does Spring Bean provide thread safety?

The default scope of Spring bean is singleton, so there will be only one instance per context. If two threads calls factory.getBean(“s”), it returns same object. if any threads changes bean property by setName() , then other thread may get inconsistence results.

That means that all the having a class level variable that any thread can update will lead to inconsistent data. **Hence in default mode spring beans are not thread-safe**.

However, we can change spring bean scope to request, prototype or session to achieve thread-safety at the cost of performance. It’s a design decision and based on the project requirements.

What is Inversion of Control concept, how does Spring support IOC? ([answer](http://javarevisited.blogspot.sg/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html))

**Removing bean creation things from developer End.** The simple meaning of inversion of the control means that now the framework, Spring is responsible for creating objects, wiring dependencies and managing their life-cycle instead of a developer, which was the case before. That's where control is inverted from developer to framework.

What is the difference between @Autowired and @Inject annotation in Spring?

The **@Inject** annotation also serves the same purpose as **@Autowired,** but the main difference between them is that

* **@Inject** is a **standard annotation(JRS -330)** for dependency injection
* **@Autowired** is **spring specific.**

How to create ApplicationContext in a Java Program?

* **AnnotationConfigApplicationContext**: If we are using Spring in standalone java applications and using annotations for Configuration, then we can use this to initialize the container and get the bean objects.
* **ClassPathXmlApplicationContext**: If we use SpringConfig.xml file in standalone application, then we can use this class to load the file and get the container object.
* **FileSystemXmlApplicationContext**: This is similar to ClassPathXmlApplicationContext except that the xml configuration file can be loaded from **anywhere in the file system**.

Name some of the design patterns used in Spring Framework?

Spring Framework is using a lot of design patterns, some of the common ones are:

1. Singleton Pattern: Creating beans with default scope.
2. [Factory Pattern](https://www.journaldev.com/1392/factory-design-pattern-in-java): Bean Factory classes
3. [Prototype Pattern](https://www.journaldev.com/1440/prototype-design-pattern-in-java): Bean scopes
4. [Adapter Pattern](https://www.journaldev.com/1487/adapter-design-pattern-java): Spring Web and Spring MVC
5. [Proxy Pattern](https://www.journaldev.com/1572/proxy-design-pattern): Spring Aspect Oriented Programming support
6. [Template Method Pattern](https://www.journaldev.com/1763/template-method-design-pattern-in-java): JdbcTemplate, HibernateTemplate etc
7. Front Controller: Spring MVC DispatcherServlet
8. Data Access Object: Spring DAO support
9. Dependency Injection
10. Aspect Oriented Programming

How to inject a java.util.Properties into a Spring Bean?

By writing in Spring bean xml file with <property> tag

<bean id="configBean" class="com.boraji.tutorial.spring.xml.DatabaseConfig">

<property name="properties">

<props>

<prop key="driverClassName">com.mysql.jdbc.Driver</prop>

<prop key="url">jdbc:mysql://localhost:3306/mydb</prop>

<prop key="username">root</prop>

<prop key="password">abcdxyz</prop>

</props>

</property>

</bean>

util:properties in Spring reads configuration file from a location

<beans>

<util:properties id=*"props"*>

<prop key=*"driverClassName"*>com.mysql.jdbc.Driver</prop>

<prop key=*"url"*>jdbc:mysql://localhost:3306/mydb</prop>

<prop key=*"username"*>root</prop>

<prop key=*"password"*>abcdxyz</prop>

</util:properties>

<bean id=*"configBean"* class=*"com.boraji.tutorial.spring.xml.DatabaseConfig"*>

<property name=*"properties"* ref=*"props"* />

</bean>

</beans>

public class DBConnection {

@Value("${DB\_DRIVER\_CLASS}")

private String driverClass;

@Value("${DB\_URL}")

private String dbURL;

@Value("${DB\_USERNAME}")

private String userName;

@Value("${DB\_PASSWORD}")

private char[] password;

public DBConnection() {

}

public void printDBConfigs() {

System.*out*.println("Driver Class = " + driverClass);

System.*out*.println("DB URL = " + dbURL);

System.*out*.println("User Name = " + userName);

// Never do below in production environment :D

System.*out*.println("Password = " + String.*valueOf*(password));

}

}

How do you turn on annotation based autowiring?

* Include <context:annotation-config > in bean configuration file.
* Use **AnnotationConfigApplicationContext** to get Context Object.

Differentiate between BeanFactory and ApplicationContext.

|  |  |
| --- | --- |
| **BeanFactory** | **ApplicationContext** |
| It uses **Lazy initialization** | It uses **Eager/ Aggressive initialization** |
| It explicitly provides a resource object using the syntax | It creates and manages resource objects on its own |
| It **doesn’t supports internationalization** | It supports internationalization |
| It doesn’t supports annotation based dependency | It supports annotation based dependency |

Can we have multiple Spring configuration files in one project?

You can load multiple Java-based configuration files:

@Configuration

@Import({MainConfig.class, SchedulerConfig.class})

public class AppConfig {

Or load one XML file that will contain all other configs:

ApplicationContext context = new ClassPathXmlApplicationContext("spring-all.xml");

And inside **this** XML file you’ll have:

<import resource="main.xml"/>

<import resource="scheduler.xml"/>

## Spring MVC

What’s the difference between @Component, @Controller, @Repository & @Service annotations in Spring?

| **ANNOTATION** | **USE** | **DESCRIPTION** |
| --- | --- | --- |
| **@Component** | Type | Generic stereotype annotation for any Spring-managed component. |
| **@Controller** | Type | Stereotypes a component as a Spring MVC controller. |
| **@Repository** | Type | Stereotypes a component as a repository. Also indicates that SQLExceptions thrown from the component's methods should be translated into Spring DataAccessExceptions. |
| **@Service** | Type | Stereotypes a component as a service. |

What is ViewResolver in Spring?

ViewResolver implementations are used to resolve the view pages by name. Usually we configure it in the spring bean configuration file. For example:

<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<beans:property name="prefix" value="/WEB-INF/views/" />

<beans:property name="suffix" value=".jsp" />

</beans:bean>

What is View Resolver pattern? how it works in Spring MVC

View Resolver pattern is a J2EE pattern which allows a web application to dynamically choose it's view technology e.g. HTML, JSP, Tapestry, JSF, XSLT or any other view technology.  
  
In this pattern, View resolver holds mapping of different views, controller return name of the view, which is then passed to View Resolver for selecting an appropriate view.

What is the difference between @Controller and @RestController?

@RestController is better when you are developing RESTful web services using Spring MVC framework. It's a combination of **@Controller + @ResponseBody** annotation which allows the controller to directly write the response and bypassing the view resolution process, which is not required for RESTful web service.   
  
It also instructs DispatcherServlet to use different HttpMessageConverters to represent the response in the format client is expecting e.g. HttpMessageJackson2Convert to represent response in JSON format and JAXB based message converts to generate XML response

What does @RequestMapping annotation do? ([answer](http://javarevisited.blogspot.sg/2017/06/how-spring-mvc-framework-works-web-flow.html" \l "axzz55vF5ugU8" \t "_blank))

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:

@RequestMapping (method = RequestMethod.POST, consumes="application/json")

public Book save(@RequestBody Book aBook) {

return bookRepository.save(aBook);

}

When do you need @ResponseBody annotation in Spring MVC?

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).

@RequestMapping(path = "/hello", method = RequestMethod.PUT)

@ResponseBody

public String helloWorld() {

return "Hello World";

}

Alternatively, you can also use @RestController annotation instead of @Controller annotation. 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?

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

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.

How to Call Stored procedure in Spring Framework?

To call a Stored procedure in Spring framework you need to create Class which will should extends **StoredProcedure** class. You just need to call the execute method from the DAO layer.

public class EmployeeInfo extends StoredProcedure

{

private static final String EMP\_ID = "EMP\_ID";

private static final String EMP\_NAME = "EMP\_NAME";

private static final String JOIN\_DATE = "JOIN\_DATE";

public SnapshotSearchStoredProcedure(DataSource dataSource, String procedureName)

{

super(dataSource, procedureName);

declareParameter(new SqlParameter(EMP\_ID, Types.NUMERIC));

declareParameter(new SqlOutParameter(EMP\_NAME, Types.VARCHAR));

declareParameter(new SqlOutParameter(JOIN\_DATE, Types.VARCHAR));

compile ();

}

public Map execute(Integer empId)

{

Map<String, Object> inputs = new HashMap<String, Object>();

inputs.put(P\_CLD\_IDR, empId);

Map<String, Object> result = execute (inputs);

return result;

}

}

How to get ServletContext and ServletConfig object in a Spring Bean?

There are two ways to get Container specific objects in the spring bean.

**Using @Autowired** annotation with bean variable of type ServletContext and ServletConfig. They will work only in servlet container specific environment only though.They alreay comes with Server jar

@Autowired

ServletContext servletContext;

How to upload file in Spring MVC Application?

Spring provides built-in support for uploading files through **MultipartResolver** interface implementations.

## Spring Data

How to use Tomcat JNDI DataSource in Spring Web Application?

For using servlet container configured JNDI DataSource, we need to configure it in the spring bean configuration file and then inject it to spring beans as dependencies. Then we can use it with **JdbcTemplate** to perform database operations

<beans:bean id="dbDataSource" class="org.springframework.jndi.JndiObjectFactoryBean">

<beans:property name="jndiName" value="java:comp/env/jdbc/MyLocalDB"/>

</beans:bean>

## Spring Security

If we think about the meaning of authentication, it seems that it is all about a client identifying itself to the server. After client identification is done, the server can remember the client each time the request comes from the client. There are two common approaches to authentication mechanisms: one of them is called "Session Cookie Based" and the other one is "Token Based".

Session Cookie based

The most common approach we probably all know is to use a server generated secret token (Session key) in the form of a JSESSIONID cookie. Initial setup for this is near nothing these days perhaps making you forget you have a choice to make here in the first place. Even without further using this “Session key” to store any other state “in the session”, the key itself is in fact *state* as well.  I.e. without a shared and persistent storage of these keys, no successful authentication will survive a server reboot or requests being load balanced to another server.

OAuth2 / API keys

Whenever talking about REST APIs and Security; OAuth2 and other types of API keys are mentioned. Basically they involve sending custom tokens/keys within the HTTP Authorization header. When used properly both relieve clients from dealing with Cookies using the header instead. This solves CSRF vulnerabilities and other Cookie related issues. One thing they do not solve however is the need for the server to check the presented authentication keys, pretty much demanding some persistent and maintainable shared storage for linking the keys to users/authorizations.

private HttpHeaders createHeaders(final String userId, final String password) {

String auth = userId + ":" + password;

byte[] encodedAuth = Base64.*encodeBase64*(auth.getBytes(StandardCharsets.*US\_ASCII*));

String authHeader = "Basic " + new String(encodedAuth);

HttpHeaders headers = new HttpHeaders();

headers.set("Authorization", authHeader);

return headers;

}

private ResponseEntity<String> makeRestCall(String url, String userId,

String password) {

// Basic Auth only.

if (!"".equals(userId) && !"".equals(password)) {

return restOperations.exchange(url, HttpMethod.*GET*,

new HttpEntity<>(createHeaders(userId, password)),

String.class);

} else {

return restOperations.exchange(url, HttpMethod.*GET*, null,

String.class);

}

}

# Spring MVC

Do you need spring-mvc.jar in your classpath or is it part of spring-core? (answer)

The spring-mvc.jar is not part of spring-core, which means if you want to use Spring MVC framework in your Java project, you must include spring-mvc.jar in your application's classpath. In Java web application, spring-mvc.jar is usually placed inside /WEB-INF/lib folder.

What is the DispatcherServlet and what is it used for? ([answer](http://www.java67.com/2017/06/what-is-use-of-dispatcherservlet-in-spring-mvc.html))

The DispatcherServlet is an implementation of Front Controller design pattern which handles all incoming web request to a Spring MVC application. A Front Controller pattern is a common pattern in web applications whose job is to receive all request and route it to different components of application for actual processing.  
  
In case of Spring MVC, DispatcherServlet route web requests to Spring MVC controllers.  
  
In Spring MVC, DispatcherServlet is used for finding the correct Controler to process a request, which it does with the help of handler mapping e.g. @RequestMapping annotation.  
  
It is also responsible for delegating logical view name to ViewResolver and then sending the rendered response to the client.

Is the DispatcherServlet instantiated via an application context? ([answer](http://javarevisited.blogspot.sg/2017/09/dispatcherservlet-of-spring-mvc-10-points-to-remember.html))  
No, DispatcherServlet is instantiated by Servlet containers like Tomcat or Jetty. You must define DispatcherServlet into the web.xml file

What is the root application context in Spring MVC? How is it loaded? ([answer](https://javarevisited.blogspot.com/2012/11/difference-between-beanfactory-vs-applicationcontext-spring-framework.html#axzz5N1cdCqrn))

In Spring MVC, the context loaded using ContextLoaderListener is called the "root" application context which belongs to the whole application while the one initialized using DispatcherServlet is actually specific to that servlet.  
  
Technically, Spring MVC allows multiple DispatcherServlet in a Spring MVC web application and so multiple such contexts each specific for respective servlet but having same root context may exist.

The ContextLoaderListener is configured in web.xml as listener and you put that inside a tag as shwon below:  
<listener>  
<listener-class>  
org.springframework.web.context.ContextLoaderListener  
</listener-class>  
</listener>  
  
When the Spring MVC web application is deployed, Servlet container created an instance of ContextLoaderListener class which loads the Spring's WebApplicationContext

**What is the @Controller annotation used for? How can you create a controller without an annotation? (**[**answer**](https://javarevisited.blogspot.com/2017/08/difference-between-restcontroller-and-controller-annotations-spring-mvc-rest.html)**)**

The @Controller is a Spring MVC annotation to define Controller but in reality, it's just a stereotype annotation.

You can even create a controller without @Controller by annotating the Spring MVC Controller classes using @Component annotation.

The real job of request mapping to handler method is done using @RequestMapping annotation.

How is an incoming request mapped to a controller and mapped to a method? ([answer](http://javarevisited.blogspot.com/2017/06/how-spring-mvc-framework-works-web-flow.html))

Sometimes this question is also asked How does DispatcherServlet knows which Controller should process the request? Well, the answer lies in something called handler mappings.  
  
Spring uses handler mappings to associate controllers with requests, two of the commonly used handler mappings are BeanNameUrlHandlerMapping and SimpleUrlHandlerMapping.  
  
In BeanNameUrlHandlerMapping, when the request url matches the name of the bean, the class in the bean definition is the controller that will handle the request.  
  
On the other hand, In SimpleUrlHandlerMapping, the mapping is more explicit. You can specify the number of URLs and each URL can be explicitly associated with a controller.  
  
Btw, if you are using annotations to configure Spring MVC, which you should then @RequestMapping annotations is used to map an incoming request to a controller and a handler method.

What are some of the valid return types of a controller method? (answer)

There are many return types are available for a controller method in Spring MVC which is annotated by @RequestMapping inside the controller. Some of the popular ones are:

1. **String**
2. **void**
3. **View**
4. **ModelAndView (Class)**
5. **Model (Interface)**
6. **Map**
7. **HttpEntity<?> or ResponseEntity<?>**
8. **HttpHeaders**

What is the Model? (answer)

Model is a reference to encapsulate data or output for rendering. Model is always created and passed to the view in Spring MVC. If a mapped controller method has Model as a method parameter, then a model instance is automatically injected by Spring framework to that method.  
  
Any attributes set on the injected model are preserved and passed to the View. Here is an example of using Model in Spring MVC:

public String personDetail(Model model) {

...

model.addAttribute("name", "Joe");

...

}

What is the purpose of the session scope? ([answer](https://javarevisited.blogspot.com/2012/05/what-is-bean-scope-in-spring-mvc.html#axzz5IZi1jCsQ))

The purpose of the session scope is to create an instance of the bean for an HTTP Session. This means the same bean can serve multiple requests if it is scoped in session.

You can define the scope of a Spring bean using scope attribute or @Scope annotation in Spring MVC application.

What is the default scope in the web context? ([answer](http://javarevisited.blogspot.sg/2012/05/what-is-bean-scope-in-spring-mvc.html))

The **singleton** scope is the default scope for a Spring bean even in the web context.

The other three Web context-aware scopes are a **request, session, and global-session,** which are only available in a web application aware ApplicationContext object.

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 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.i mean we can create client class to test webservice manulay using java.

You can use it to consume a RESTful Web Servicer very easily as shown in this example.

public class App implements CommandLineRunner {

private static final Logger *log* = LoggerFactory.getLogger(App.class);

public static void main(String args[]) {

SpringApplication.run(App.class);

}

public void run(String... args) throws Exception {

RestTemplate restTemplate = new RestTemplate();

Response response = restTemplate.getForObject("localhost:9090/student/getall",

Response.class);

*log*.info("==== RESTful API Response using Spring RESTTemplate START =======");

*log*.info(response.toString());

*log*.info("==== RESTful API Response using Spring RESTTemplate END =======");

}

}

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.

**to create a custom implementation of HttpMessageConverter to support a new type of request/responses,**   
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 @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.

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.

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))

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 @ResponseStatus annotation, which allows you to send custom HTTP status code along with proper error message in case of Exception.

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 {

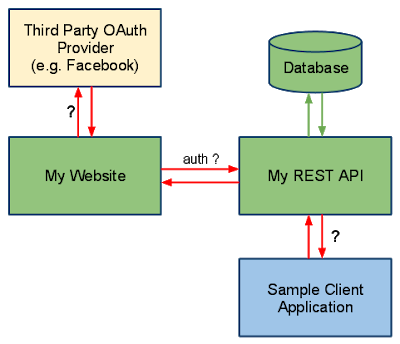
// ...

}

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?

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.

[](http://javarevisited.blogspot.sg/2018/01/how-to-enable-http-basic-authentication-spring-security-java-xml-configuration.html)

# SpringBoot IQ’s

What is the latest version of spring boot and its System requirement?

As per July, 2018 Spring boot latest version is **2.1.**

Spring boot needs **Java 8+ version** and Spring 5 framework as minimum version.

How does Spring enable create production ready applications in quick time?

Spring Boot aims to enable production ready applications in quick time. Spring Boot provides a few non functional features out of the box like caching, logging, monitoring and embedded servers.

* spring-boot-starter-actuator - To use advanced features like monitoring & tracing to your application out of the box
* spring-boot-starter-undertow, spring-boot-starter-jetty, spring-boot-starter-tomcat - To pick your specific choice of Embedded Servlet Container
* spring-boot-starter-logging - For Logging using logback
* spring-boot-starter-cache - Enabling Spring Framework’s caching support

What is the minimum baseline Java Version for Spring Boot 2 and Spring 5?

Spring Boot 2.0 requires Java 8 or later. Java 6 and 7 are no longer supported.

What do Dev Tools in Spring boot mean?

By using devtools, You don’t have to redeploy your application each time you made the changes.

The developer can reload the progressions without restart of the server.

It maintains a strategic distance from the agony of redeploying application each time when you roll out any improvement. This module will can’t be utilized in a production environment.

Spring DevTools & uses

Applications that use spring-boot-devtools will **automatically restart whenever files on the classpath change**. This can be a useful feature when working in an IDE as it gives a very fast feedback loop for code changes.

Would we be able to Use Spring Boot with Applications Which Are Not Using Spring?

No, it isn’t conceivable starting at now. Spring boot is restricted to Spring applications only.

How to connect to an external database like MSSQL or oracle with Spring boot?

**Step 1** -The first step to connect the database like Oracle or MySql is adding the dependency for your database connector to pom.xml.

**Step 2** -The next step is the elimination of H2 Dependency from pom.xml

**Step 3** -Step 3 includes the schema and table to establish your database.

**Step 4** -The next step is configuring of the database by using Configure application.properties to connect to your database.

**Step 5**-And the last step is to restart your device and your connection is ready to use.

How to disable a specific auto-configuration?

If we want to disable a specific auto-configuration, we can indicate it using the **exclude attribute** of the **@EnableAutoConfiguration** annotation. For instance, this code snippet neutralizes DataSourceAutoConfiguration:

// other annotations

@EnableAutoConfiguration(exclude = DataSourceAutoConfiguration.class)

public class MyConfiguration { }

If we enabled auto-configuration with the @SpringBootApplication annotation — which has @EnableAutoConfiguration as a meta-annotation — we could disable auto-configuration with an attribute of the same name:

@SpringBootApplication(exclude = DataSourceAutoConfiguration.class)

public class MyConfiguration { }

We can also disable an auto-configuration with the spring.autoconfigure.exclude environment property in the application.properties

spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration

How to deploy Spring Boot web applications to WebSphere as a WAR?

spring-boot-maven-plugin handles the Packing things.

<plugin>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

If we mentioned in our POM.xml packaging as jar it will generate Jar by including Tomcat server as well.

To Build WAR with out Tomact we need to below 2 Changes

**1.In POM.xml – package type as ‘war’**

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.3.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>demo</name>

<description>Demo project for Spring Boot</description>

**2.Remove tomcat starter Dependency**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

What is Spring Boot DevTools used for?

Spring Boot Developer Tools, or DevTools, is a set of tools making the development process easier.

The *spring-boot-devtools* module is automatically disabled if the application runs in production

By default, DevTools applies properties suitable to a development environment. These properties disable template caching, enable debug logging for the web group, and so on. As a result, we have this sensible development-time configuration without setting any properties.

Applications using DevTools restart whenever a file on the classpath changes. This is a very helpful feature in development, as it gives quick feedback for modifications.

By default, static resources, including view templates, don’t set off a restart. Instead, a resource change triggers a browser refresh. Notice this can only happen if the LiveReload extension is installed in the browser to interact with the embedded LiveReload server that DevTools contains.

What is LiveReload?

The spring-boot-devtools module includes an **embedded LiveReload server** **that can be used to trigger a browser refresh when a resource is changed**. LiveReload browser extensions are freely available for Chrome, Firefox and Safari from livereload.com.

How to exclude auto restart for static files?

By default changing resources in**/META-INF/maven**, **/META-INF/resources**, **/resources, /static**, **/public or /templates** will not trigger a restart.

But If you want to customize these exclusions you can use the **spring.devtools.restart.exclude** property.

If you want to keep those defaults and add additional exclusions, use the **spring.devtools.restart.additional-exclude** property instead.

What is Hot swapping in spring boot?

**Reloading the changes without restarting the server is called hot swapping**, Modern IDEs (Eclipse, IDEA, etc.) all support hot swapping of bytecode, so if you make a change that doesn’t affect the class or method signatures it should reload cleanly with no side effects.

How to write integration tests?

When you create Project using Spring.io, by default test class for Application class also will created. It is annotated with **@SpringBootTest**

@RunWith(SpringRunner.class)

@SpringBootTest

public class DemoApplicationTests {

@Test

public void contextLoads() {

}

}

 What is Spring Boot Actuator?

Spring Boot provides actuator to monitor and manage our application. **Actuator is a tool which has HTTP endpoints**. when application is pushed to production, you can choose to manage and monitor your application using HTTP endpoints.

**The actuator provides features like auditing, health, metrics, environment information, thread dump etc.** using HTTP or JMX endpoints. Here are some of the most common built-in actuator endpoints:

* *beans* – **Displays a complete list of all the Spring beans in your application.**
* auditevents – Exposes audit events information for the current application.
* *caches* – Exposes available caches.
* *configprops* – Displays a collated list of all @ConfigurationProperties.
* *health* – **Shows application health information**.
* *info* – Displays arbitrary application info.
* *metrics* – Shows ‘metrics’ information for the current application.
* *mappings* – **Displays a collated list of all @RequestMapping paths.**
* *sessions* – Allows retrieval and deletion of user sessions from a Spring Session-backed session store.
* threaddump – Performs a thread dump.

How do you Change tomcat HTTP port & Context URL?

You can change the Tomcat HTTP port by changing default HTTP property in the **application.properties** file.

Server.port = 8080

Server.context.path = /

Can you control logging with Spring Boot? How?

Yes, we can control logging with Spring Boot by specifying log levels on application.properties file.

Spring Boot loads this file when it exists in the [classpath](http://www.java67.com/2012/08/what-is-path-and-classpath-in-java-difference.html) and it can be used to configure both Spring Boot and application code.  
  
Spring Boot uses Commons Logging for all internal logging and you can change log levels by adding following lines in the **application.properties** file:  
  
logging.level.org.springframework=DEBUG  
logging.level.com.demo=INFO

What is YAML ?

YAML is a human-readable data serialization language. It is commonly used for configuration files.

Compared to properties file, YAML file is much more structured and less confusing in case we want to add complex properties in the configuration file. As can be seen YAML has hierarchical configuration data

How to set the active profile in Spring Boot?

There are two ways to set the active profile in Spring Boot.

Pass in the active profile as an argument while launching the application.

java -jar -Dspring.profiles.active=production application-1.0.0-RELEASE.jar //pass as command line argument

Use the application.properties file to set the active profile.

spring.profiles.active=production

# JUnit

How to create Parameterized tests?

For this java class

public class EvenNumbers {

public Boolean checkEven(final Integer num) {

for (int i = 1; i <= num; i++) {

if (i % 2 == 0) {

return true;

}

}

return false;

}

}

There are five steps to create Parameterized tests−

* First, test class is annotated with **@RunWith** which is a **Parameterized.class.**
* Then create a **public static method** which is annotated with **@Parameters**. it returns a Collection of Objects as test data set.
* Next, create a public constructor**, which takes in one row of test data**.
* Create an instance variable that is for each column of the test data row.
* Create tests case(s) using the instance variables as a source of the test data.
* The test case invokes once per each row of data.

Example

package parameterizedtest;

import java.util.\*;

import org.junit.\*;

import static org.junit.Assert.*assertEquals*;

@RunWith(Parameterized.class)

public class PrimeNumberCheckerTest {

private Integer inum;

private Boolean res;

private EvenNumbers evenObj;

@Before

public void initialize() {

evenObj = new EvenNumbers();

}

public PrimeNumberCheckerTest(Integer inum, Boolean res) {

this.inum = inum;

this.res = res;

}

@Parameterized.Parameters

public static Collection evenNumbers() {

return Arrays.*asList*(new Object[][] {

{ 2, true },

{ 6, true },

{ 18, true },

{ 19, false },

{ 48, true }

});

}

@Test

public void testPrimeNumberChecker() {

System.*out*.println("Parameterized Number is : " + inum);

*assertEquals*(res, evenObj.checkEven(inum));

}

}

Parameterized Number is : 2

Parameterized Number is : 6

Parameterized Number is : 18

Parameterized Number is : 19

Parameterized Number is : 48

What are JUnit classes? List some of them?

JUnit classes are important classes which are used in writing and testing JUnits. Some of the important classes are:

* Assert – A set of assert methods.
* Test Case – It defines the fixture to run multiple tests.
* Test Result – It collects the results of executing a test case.
* Test Suite – It is a Composite of Tests.

# MongoDB vs SQL

| **SQL Terms/Concepts** | **MongoDB Terms/Concepts** |
| --- | --- |
| database | [database](https://docs.mongodb.com/manual/reference/glossary/#term-database) |
| table | [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection) |
| row | [document](https://docs.mongodb.com/manual/reference/glossary/#term-document) or [BSON](https://docs.mongodb.com/manual/reference/glossary/#term-bson) document |
| column | [field](https://docs.mongodb.com/manual/reference/glossary/#term-field) |
| index | [index](https://docs.mongodb.com/manual/reference/glossary/#term-index) |
| table joins | [$lookup](https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/#pipe._S_lookup), embedded documents |
| primary key  Specify any unique column or column combination as primary key. | [primary key](https://docs.mongodb.com/manual/reference/glossary/#term-primary-key)  In MongoDB, the primary key is automatically set to the [\_id](https://docs.mongodb.com/manual/reference/glossary/#term-id)field. |

<https://docs.mongodb.com/manual/reference/sql-comparison/>

**Read this, Don’t forget important**

# Real

Is Hibernate is slow? Did you face any memory issues?

Better use *FetchType.LAZY* instead.

The N+1 problem

Supplier with a one-to-many relationship with Product. One Supplier has (supplies) many Products.

\*\*\*\*\* Table: Supplier \*\*\*\*\*

+-----+-------------------+

| ID | NAME |

+-----+-------------------+

| 1 | Supplier Name 1 |

| 2 | Supplier Name 2 |

| 3 | Supplier Name 3 |

| 4 | Supplier Name 4 |

+-----+-------------------+

\*\*\*\*\* Table: Product \*\*\*\*\*

+-----+-----------+--------------------+-------+------------+

| ID | NAME | DESCRIPTION | PRICE | SUPPLIERID |

+-----+-----------+--------------------+-------+------------+

|1 | Product 1 | Name for Product 1 | 2.0 | 1 |

|2 | Product 2 | Name for Product 2 | 22.0 | 1 |

|3 | Product 3 | Name for Product 3 | 30.0 | 2 |

|4 | Product 4 | Name for Product 4 | 7.0 | 3 |

+-----+-----------+--------------------+-------+------------+

Factors:

* Lazy mode for Supplier set to “true” (default)
* Fetch mode used for querying on Product is Select
* Fetch mode (default): Supplier information is accessed
* Caching does not play a role for the first time the

Supplier is accessed

Fetch mode is Select Fetch.Eager (default)

// It takes Select fetch mode as a default

Query query = session.createQuery( "from Product p");

List list = query.list();

// Supplier is being accessed

displayProductsListWithSupplierName(results);

select ... various field names ... from PRODUCT

select ... various field names ... from SUPPLIER where SUPPLIER.id=?

select ... various field names ... from SUPPLIER where SUPPLIER.id=?

select ... various field names ... from SUPPLIER where SUPPLIER.id=?

Result:

* 1 select statement for Product
* N select statements for Supplier

This is N+1 select problem!

The Solution

* Avoiding Eager Fetching
* Only fetching the data that are actually needed

jpaQuery = entityManager.createQuery("SELECT P FROM PurchaseOrder P WHERE P.customerId = :customerId", PurchaseOrder.class);

jpaQuery.setParameter("customerId", "Sayem")

* We can use [JOIN FETCH](https://www.objectdb.com/java/jpa/query/jpql/from#LEFT_OUTER__INNER_JOIN_FETCH_) in our queries whenever we need to fetch an entity with all of its children at the same time. This results in a much less database traffic resulting in an improved performance.

Clone() will create new Object of Singleton Class?

[Cloning](https://www.geeksforgeeks.org/clone-method-in-java-2/) is a concept to create duplicate objects. **Using clone we can create copy of object**. Suppose, we ceate clone of a singleton object, then it wil create a copy that is there are two instances of a singleton class, **hence the class is no more singleton.**

public static void main(String args[]) throws CloneNotSupportedException {

Student s1 = Student.*getObject*();

Student s2 = Student.*getObject*();

Student s3 = (Student) s1.clone();

System.*out*.println(s1);

System.*out*.println(s1);

System.*out*.println(s3);

}

Student@15db9742

Student@15db9742

Student@6d06d69c // Creates new Object, our singleton failed.

To overcome this, we should override clone() method, it should throw Exception, anyone tries to do clone

class Student implements Cloneable{

……….

@Override

protected Object clone() throws CloneNotSupportedException {

throw new CloneNotSupportedException();

}

public static void main(String args[]) throws CloneNotSupportedException {

Student s1 = Student.*getObject*();

Student s2 = Student.*getObject*();

Student s3 = (Student) s1.clone();

System.*out*.println(s1);

System.*out*.println(s1);

System.*out*.println(s3);

}

}

Exception in thread "main" java.lang.CloneNotSupportedException //We are GOOD now

at Student.clone(Student.java:25)

at Student.main(Student.java:33)

**Reflection:** You can make the new instance of the Singleton class by changing the **constructor visibility as public** in run-time and create new instance using that constructor .

public static void main(String args[]) throws InstantiationException, IllegalAccessException, IllegalArgumentException, InvocationTargetException {

Student s1 = Student.*getObject*();

Student s2 = null;

//1.Making Construtor visible

Constructor<Student>[] constructors = (Constructor<Student>[]) Student.class.getDeclaredConstructors();

for (Constructor constructor : constructors)

{

// Below code will destroy the singleton pattern

constructor.setAccessible(true);

s2 = (Student) constructor.newInstance();

}

System.*out*.println(s1);

System.*out*.println(s2);

//Using Class of newInstance()

Class c = Student.class;

Student s1 = Student.*getObject*();

Student s2 = (Student) c.newInstance();

System.*out*.println(s1); // Student@15db9742

System.*out*.println(s2); // Student@6d06d69c -Failed again ☹

}

Student@15db9742

Student@6d06d69c //Failed again ☹

To prevent Singleton failure while due to reflection you have to **throw a run-time exception in constructor**, if the constructor is already initialized.

class Student implements Cloneable{

private static Student *st*;

private Student() {

if(*st*!=null)

throw new RuntimeException("Go Fucker.....");

}

Exception in thread "main" java.lang.RuntimeException: Go Fucker.....

at Student.<init>(Student.java:15)

at sun.reflect.NativeConstructorAccessor

Sum of mutilple combinations of Integer array?

public class findSubsetsThatSumToATarget {

private static HashSet<String> *allSubsets* = new HashSet<>();

private static final String *token* = " ";

/

public static void findTargetSumSubsets(int[] input, int target, String ramp, int index) {

if (index > (input.length - 1)) {

if (*getSum*(ramp) == target) {

*allSubsets*.add(ramp);

}

return;

}

// First recursive call going ahead selecting the int at the currenct index

// value

*findTargetSumSubsets*(input, target, ramp + input[index] + *token*, index + 1);

// Second recursive call going ahead WITHOUT selecting the int at the currenct

// index value

*findTargetSumSubsets*(input, target, ramp, index + 1);

}

private static int getSum(String intString) {

int sum = 0;

StringTokenizer sTokens = new StringTokenizer(intString, *token*);

while (sTokens.hasMoreElements()) {

sum += Integer.*parseInt*((String) sTokens.nextElement());

}

return sum;

}

public static void main(String[] args) {

int[] n = { 24, 1, 15, 3, 4, 15, 3 };

int counter = 1;

FindSubsetsThatSumToATarget.findTargetSumSubsets(n, 25, "", 0);

for (String str : *allSubsets*) {

System.*out*.println(counter + ") " + str);

counter++;

}

}

}

# H R Mapping

Reason for Job Change"

I am looking for better opportunities! Both Technically & Finacially. I want to the part of Product based things, innovating / implementing some thing new instead of working with existing code.

I think this job would be a great opportunity in my career.

**Keep in Mind:**

* Tell the truth, yes! because big shots like google or microsoft sees honesty in employees.
* Be positive. Tell them what's your work there and how it's affecting you blah blah
* Tell them, I'm good at 'x' but I"m doing 'y' there.
* Put your points in polite manner.

Technology Analyst – Roles & Responcibilites

As a Technology Analyst, I have to take part in client calls,

understand the various requirements, implement solutions, suggest work improvements,

mentoring of junior resources for technical assistance, report the work status in timely manner.

Software Engineer

Understanding Requirements

Implementing Business Logic

Application Development

hands-on with many different technologies

worked in different layers of Product development

Learning the discipline to manage myself

# References

**1.**[**https://javarevisited.blogspot.com/2017/01/how-to-prepare-for-java-interviews.html**](https://javarevisited.blogspot.com/2017/01/how-to-prepare-for-java-interviews.html)

* 133 Java Interview Questions from last 5 years ([list](https://javarevisited.blogspot.sg/2015/10/133-java-interview-questions-answers-from-last-5-years.html)) **(Done)**
* 50 Java Concurrency Interview Questions ([list](http://javarevisited.blogspot.com/2014/07/top-50-java-multithreading-interview-questions-answers.html)) **(Done)**
* 25 Java Collection Interview Questions ([list](http://javarevisited.blogspot.com/2011/11/collection-interview-questions-answers.html)) **(Done)**
* 10 Spring Framework Interview Questions ([list](http://javarevisited.blogspot.com/2011/09/spring-interview-questions-answers-j2ee.html)) **(Done)**
* 20 Hibernate Framework Interview Questions with Answers ([list](http://java67.blogspot.com/2016/02/top-20-hibernate-interview-questions.html)) ) **(Done)**
* 10 RESTful Web Service interview Questions for Java developers ([list](http://www.java67.com/2015/09/top-10-restful-web-service-interview-questions-answers.html)) **(Done)**
  + Top 10 Spring Framework Interview Questions with Answers ([see here](http://javarevisited.blogspot.sg/2011/09/spring-interview-questions-answers-j2ee.html))
  + 20 Great Java Design Pattern Questions asked on Interviews ([see here](http://java67.blogspot.com/2012/09/top-10-java-design-pattern-interview-question-answer.html))
  + 10 popular Struts Interview Questions for Java developers ([list](http://javarevisited.blogspot.sg/2011/11/struts-interview-questions-answer-j2ee.html))
  + 10 frequently asked Servlet Interview Questions with Answers ([see here](http://javarevisited.blogspot.sg/2011/09/servlet-interview-questions-answers.html))
  + 20 jQuery Interview Questions for Java Web Developers ([list](http://javarevisited.blogspot.sg/2015/02/top-16-jquery-interview-questions.html))
  + 10 Great Oracle Interview Questions for Java developers ([see here](http://javarevisited.blogspot.sg/2012/12/top-10-oracle-interview-questions-and-answers-database-sql.html))
  + Top 10 JSP Questions  from J2EE Interviews ([read here](http://javarevisited.blogspot.sg/2011/10/jsp-interview-questions-answers-for.html))
  + 12 Good RESTful Web Services Questions from Interviews ([read here](http://javarevisited.blogspot.sg/2012/01/rest-web-services-framework-interview.html))
  + Top 10 EJB Interview Questions and Answers ([see here](http://javarevisited.blogspot.sg/2012/03/top-10-ejb-interview-question-and.html))
  + Top 10 JMS and MQ Series Interview Questions and Answers ([list](http://javarevisited.blogspot.sg/2014/03/top-10-websphere-mq-series-interview-questions-answers-active-rabbit.html))
  + 10 Great Hibernate Interview Questions for Java EE developers ([see here](http://javarevisited.blogspot.sg/2013/05/10-hibernate-interview-questions-answers-java-j2ee-senior.html))
  + 10 Great JDBC Interview Questions for Java Programmers ([questions](http://javarevisited.blogspot.sg/2012/12/top-10-jdbc-interview-questions-answers.html))
  + 15 Java NIO and Networking Interview Questions with Answers ([see here](http://javarevisited.blogspot.sg/2014/08/socket-programming-networking-interview-questions-answers-Java.html))
  + Top 10 XSLT Interview Questions with Answers ([read more](http://javarevisited.blogspot.sg/2013/05/10-xslt-or-xml-xsl-transformation-interview-questions-answers-java.html))
  + 15 Data Structure and Algorithm Questions from Java Interviews ([read here](http://javarevisited.blogspot.com/2013/03/top-15-data-structures-algorithm-interview-questions-answers-java-programming.html))
  + Top 10 Trick Java Interview Questions and Answers ([see here](http://java67.blogspot.com/2012/09/top-10-tricky-java-interview-questions-answers.html))
  + Top 40 Core Java Phone Interview Questions with answers ([list](http://java67.blogspot.com/2015/03/top-40-core-java-interview-questions-answers-telephonic-round.html))

**2.**[**https://www.pearsonfrank.com/blog/java-interview-questions/**](https://www.pearsonfrank.com/blog/java-interview-questions/)

**3.**[**https://howtodoinjava.com/java-interview-questions/**](https://howtodoinjava.com/java-interview-questions/)

**4.**[**https://snowdream.github.io/**](https://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/exception.html)

<https://javaconceptoftheday.com/>

<http://www.thejavageek.com/core-java/>