**Springboot:**

**Why to use springboot rather than spring MVC?**

* Springboot has embedded Tomcat server, no need to add external server.
* It has starter form Maven Dependency. (Web Starter, JPA Starter, etc.)
* It automatically configure spring application by using @Autoconfigure based on the dependencies.
* It has annotations that’s why it skips XML configurations.

**Different ways to create springboot annotations?**

* Using Spring Initializer (<https://start.spring.io/>)
* Using Spring STS (Spring Tool Suite)
* Using Springboot CLI
* Using Spring IDE (Eclipse)

**How does springboot application works?**

* Create new project of springboot application.
* First specify dependency starter in pom.xml
* Then dependencies are added to project.
* Executes program from main ( ) where auto configuration done.
* Executes SpringApplication.run( ) that launches program.

**What does @SpringBootApplication do?**

* It marks class as a bean.
* It has 3 annotations working inbuilt are @Configuration , @EnableAutoConfiguration and @ComponentScan.
* @Configuration = It tells that class is spring bean.
* @EnableAutoConfiguration = It configures application as per jars added.
* @ComponentScan = Scans other packages in application.

**SpringApplication.run ( ) statement does?**

* It bootstraps and launches the application.
* How do we monitor & manages our application?
* To get production ready feature we use springboot actuator.
* It has inbuilt HTTP endpoints.
* We can create our endpoints.
* Enable this using springboot actuator.
* E.g. of endpoints /bean (name of beans in application), /health (up or down of application).

**What is springboot profile (@profile)?**

* It provide ways to divide application to spare them for particular environment.
* We specify @Profile to tell code is written for particular environment.
* Environments like Development environment, Production environment, Testing environment, etc.
* Each environment has different configurations.

**What is spring data JPA?**

* It has 3 repository interfaces in that-
* CrudRepository (It is parent interface, used for CRUD operations)
* PagingAndSortingRepository (It extends CrudRepository, used for Paging & Sorting purpose)
* JPARepository (It extends PagingAndSortingRepository, used for all above and its additional services)

**How to create custom queries?**

* Using @Query annotation.
* E.g. @Query("select c from City c where c.name like %?1") List<City> findByNameEndsWith(String chars);

**How to configure Hibernate in springboot?**

* Add JPA & MySQL database dependency.
* In application.properties, write hibernate configurations to connect database.

**How application.properties file added to container?**

* @EnableAutoConfiguration annotation is responsible for connection between database & application while configuring hibernate by adding application.properties file to container from its fixed path or specified path.

**RESTful Web Service:**

**What is web service?**

* It is medium or way of communicating application over the network.
* It is a client-server application or application component for communication.
* The method of communication between two devices over the network.
* It is a software system for the interoperable machine to machine communication.
* It is a collection of standards or protocols for exchanging information between two devices or application.

**What are the advantages of web service?**

* Interoperability: Web services are accessible over network and runs on HTTP/SOAP protocol and uses XML/JSON to transport data, hence it can be developed in any programming language. Web service can be written in java programming and client can be PHP and vice versa.
* Reusability: One web service can be used by many client applications at the same time.
* Loose Coupling: Web services client code is totally independent with server code, so we have achieved loose coupling in our application.
* Easy to deploy and integrate, just like web applications.
* Multiple service versions can be running at same time.

**What are the types of web service and difference between them?**

SOAP (Simple object Access Protocol) & REST (Representational State transfer) are the types of web services.

|  |  |
| --- | --- |
| SOAP | REST |
| It supports XML data format only. | It supports XML, HTML, Plain text, JSON,  etc. data formats. |
| It is protocol based. | It is architecture based. |
| SOAP cannot use REST. | REST can use SOAP. |
| It has its own security. | Have to add security. |
| It is less preferred. | It is more preferred. |
| Need JAX-WS API to implement SOAP. | Need JAX-RS API to implement REST. |

**What are the ways to implement Restful web service?**

* In JAX-RS, there are 3 types to implement Restful web service.
* Jersey.
* REST easy.
* Spring with REST.

.

**By using spring with REST, how do we create Restful web service?**

* Add REST dependency in pom.xml. (spring-boot-starter-data-rest)
* Add @RestController annotation for controller class.
* Map the methods written in controller class.
* What is REST? Why do you choose Restful web service?
* REST is the acronym for REpresentational State Transfer. REST is an architectural style for developing applications that can be accessed over the network.
* REST is a stateless client-server architecture where web services are resources and can be identified by their URIs (Uniform Resource Identifiers).
* We choose Restful web service because-
* It is architectural style not protocol.
* It is fast and has no restrictions like SOAP.
* It uses data formats like XML, HTML, Plain Text, JSON, etc.
* It also can use SOAP.
* It is programming language and platform (OS) independent.

**Annotations used in Restful web service?**

* @RestController
* @GetMapping
* @PostMapping
* @DeleteMapping
* @PutMapping
* @PathVariable
* @RequestBody
* @ResponseBody

**Difference between @Controller & @RestController?**

|  |  |
| --- | --- |
| @Controller | @RestController |
| It is used to mark a class as Spring MVC Controller. | It is used in RESTFul web services and the equivalent of @Controller +  @ResponseBody. |
| It returns view. | It returns data. |
| Added in Spring 2.5 version. | Added in Spring 4.0 version. |

**Difference between @PathVariable & @PathParam?**

|  |  |
| --- | --- |
| @PathVariable | @PathParam |
| It is from spring framework. | It is from JAX-RS. |
| It will work in spring MVC and REST | It will work in REST only. |
| It is annotation on a method argument to  bind it to the value of a URI template variable. | It is a parameter annotation which allows  you to map variable URI path fragments into your method call. |

**Have you Consumed or Produced Restful web service? Explain?**

Consume or Produce in Restful web service means specifying which type of data to be requested (it is just telling type of data accepted or send through application)

* E.g. While consuming data type we use syntax, consumes=”application/JSON”, it specify that certain service accepts only JSON data format.
* Produce is associated with @GetMapping annotation where we mention type of data while we are sending it when that method/service is called.
* Consume is associated with @PostMapping annotation where we mention type of data is to be accepted for saving it into the database when that method/service is called.
* E.g. Film Producer makes (Produce) Movies & Viewers watch (Consume) those movies.

**How to consume and produce data using xml format?**

* First use @XmlRootElement on POJO class or need to add Xml data supporting dependency.
* Then in Controller methods specify,
* (For Produce) produces=”application/xml”
* (For Consume) consumes=”application/xml”
* In Restful web service, how to convert JSON format data to Object format or vice versa?
* Object to JSON format:
* ObjectMapper mapper=new ObjectMapper( );
* String jsonResult=mapper.writeValueAsString(object);
* JSON to Object format:
* ObjectMapper mapper=new ObjectMapper( );
* Student s=mapper.readValue (jsonResult, Student.class);

**Difference between API & web services?**

|  |  |
| --- | --- |
| API | Web Services |
| It is an interface between two different  applications. | It is interaction between applications  over the network. |
| All APIs are not web services. | All web services are APIs. |
| API can be used for any style of  communication. | Web service uses styles like REST, SOAP  for communication. |
| It can be used by a client who  understands JSON or XML. | It can be used by any client who  understands XML. |
| API has a light-weight architecture. | Web Services does not have a light-  weight architecture. |

**How to test web service?**

* Using ARC & POSTMAN.
* Locally –
* Create war file.
* Deploy on local Tomcat server.
* How do you mock the web service?
* Mocking Web service is testing application using dummy data.

**What are HTTP methods?**

* GET, POST, DELETE, PUT, etc.
* GET - The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.
* POST - The POST method is used to submit an entity to the specified resource, often causing a change in state.
* PUT - The PUT method replaces all current representations of the target resource with the request payload.
* DELETE - The DELETE method deletes the specified resource.
* PATCH - The PATCH method is used to apply partial modifications to a resource.

**What are components in HTTP?**

Request, Body, Header, etc