1. What is Loose Coupling?

Thight Coupling -

Example-1:

```
public class TodoBusinessService{
        TodoDataServiceImpl dataService = new TodoDataServiceImpl();
        ...
```

Now here TodoBusinessService is dependent on TodoDataServiceImpl. Its directly creating a instance of TodoDataServiceImpl using new.

Example-2:

```
public class ComplexAlgorithmImpl{
    BubbleSortAlgorithm bubbleSortAlgorithm = new BubbleSortAlgorithm();
    ...
```

Same way ComplexAlgorithmImpl is dependent on BubbleSortAlgorithm. This is tight coupling. Suppose tommorrow we need to switch from bubble sort to quick sort, we will have to make this change in here.

Loose Coupling -

Example-1:

```
@Component
public class TodoBusinessService{

    @Autowired
    TodoDataService dataService;        // = new TodoDataService

    public TodoBusinessService(TodoDataService dataService){
        this.dataService = dataService;
    }

    .....

public interface TodoDataService{
    List<String> retrieveTodos(String user);
}
```

Now here to make it loose coupling, we are telling that whoever wants to use TodoBusinessService needs to pass TodoDataService implementation. Now TodoBusinessService is no longer dependent on TodoDataServiceImpl class. TodoDataServiceImpl class can implement TodoDataService and provide implementation for retrieveTodos(). Then if anyone wants to use TodoBusinessService he will then pass TodoDataServiceImpl while creating TodoBusinessService

(**doubt - what if we are Autowiring TodoBusinessService as well, how can we pass TodoDataService implementation then? - I think, Due to the @Component annotation on TodoBusinessService we don't have to instantiate it. If we autowire TodoBusinessService, Spring will search for it, then it will find new dependency of TodoDataService, it will then find the implementation class for this, once found it will autowire it in TodoBusinessService and now we have TodoBusinessService complete and ready to be used.)

Example-2:

```
@Component
public class ComplexAlgorithmImpl {

    @Autowired
    private SortAlgorithm sortAlgorithm;

    public ComplexAlgorithmImpl(SortAlgorithm sortAlgorithm){
```

```
                    this.sortAlgorithm = sortAlgorithm;
                }
                ...


            public interface SortAlgorithm {
                public int[] sort(int[] numbers);
            }


            public class QuickSortAlgorithm implements SortAlgorithm {


            public class BubbleSortAlgorithm implements SortAlgorithm {


            Now here at the time of creating ComplexAlgorithmImpl, user can inject
            either QuickSortAlgorithm or BubbleSortAlgorithm according to his need.
            ComplexAlgorithmImpl is no longer dependent on specific implementation.

    This Loose Coupling.


2. What is dependency?
    In the above code TodoDataService is dependency for TodoBusinessService and
    SortAlgorithm is dependency for ComplexAlgorithmImpl

3. What is Dependency Injection?

    ComplexAlgorithmImpl binarySearch = new ComplexAlgorithmImpl(new
    QuickSortAlgorithm());     //We are doing this explicitly

    We use Spring Framework to instantiate beans and wire dependencies

    @Component
    public class ComplexAlgorithmImpl {

      @Autowired
      private SortAlgorithm sortAlgorithm;
        ...

    In this case when spring is creating bean for ComplexAlgorithmImpl, it will detect
    the SortAlgorithm dependency because of Autowired annotation, and it will start
    finding possible matches to resolve that dependency. And as soon as it finds
    QuickSortAlgorithm, it will instantiate it and inject in ComplexAlgorithmImpl. This
    is dependency injection.
    So spring searches for beans and once it has found the appropriate bean it will
    autowire it.

    Dependency Injection is process where spring identifies the dependencies, starts
    searching then and once found it will instantiate them and autowire them.

4. What is Inversion Of Control?

    public class ComplexAlgorithmImpl {

      BubbleSortAlgorithm bubbleSortAlgorithm = new BubbleSortAlgorithm();
      ...

    In above case, the ComplexAlgorithmImpl is responsible to create the dependency
    i.e. BubbleSortAlgorithm. So the control is with ComplexAlgorithmImpl.
    This tightly coupled.

    @Component
    public class ComplexAlgorithmImpl {

      @Autowired
      private SortAlgorithm sortAlgorithm;
        ...
```

Here we are not instantiating the SortAlgorithm or dependency. Here the control or responsibility of creating the dependency goes to the user(who will be using ComplexAlgorithmImpl class) or to framework (Spring). So Spring will take control, search the dependencies and inject.

So now the control has been inversed as it was in earlier case. This is inversion of control.

5. What is Autowiring?

```
@Component
public class ComplexAlgorithmImpl {

  @Autowired
  private SortAlgorithm sortAlgorithm;
    ...
```

When we write @Autowired, Spring framework needs to find the bean which would match this dependency and then it would have to populate the dependency in there. This process is called Autowiring.

6. What are the important roles of an IOC Container? What are Bean Factory and Application Context? Can you compare Bean Factory with Application Context?

- Find the beans
- Wire Dependencies
- Manage Lifecycle of the bean

Example-1:

```
@Component
public class ComplexAlgorithmImpl {

  @Autowired
  private SortAlgorithm sortAlgorithm;
  ...


@Component
public class QuickSortAlgorithm implements SortAlgorithm {


public interface SortAlgorithm {
  public int[] sort(int[] numbers);
}
```

In the above code IOC container will first identify the class with annotation @Component. This will idicate it that these are the beans it needs to create. This is step one.
Second is to autowire dependencies.

The two terms that are related to IOC container are Application Context and Bean Factory.

Bean Factory is the most basic version of IOC container that spring provides.

Bean Factory, the basic things of IOC container it will be able to that - find beans, wire dependencies, manage life cycle of bean.

Application Context is much more advance kind of IOC Container.
Application Context = Bean Factory + Spring's AOP features + I18n capabilities + WebApplicationContext for web applications, etc

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

We can define it using an XML or we can define it using an annotation

@Configuration.

Using XML -

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                      http://www.springframework.org/schema/tx
                      http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
                      http://www.springframework.org/schema/context
                      http://www.springframework.org/schema/context/spring-context-3.0.xsd
                      http://www.springframework.org/schema/aop
                      http://www.springframework.org/schema/aop/spring-aop.xsd
                      ">
</beans>
ApplicationContext context =
    new ClassPathXmlApplicationContext(
    new String[] {"BusinessApplicationContext.xml",
        "Other-Configuration.xml"});
```

Using Annotation -

```
@Configuration                          //Just by this annotation we have defined application context.
class SpringContext {
}

ApplicationContext ctx =
     new AnnotationConfigApplicationContext(
          SpringContext.class);
```

We can also create ApplicationContext in out Unit tests -

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = JavaTestContext.class)
//JavaTestContext is our java annotation configuration
public class DependencyInjectionJavaContextExamples {
...



@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "/TestContext.xml" })
//TestContext.xml is our xml configuration
public class TodoBusinessTest {
...
```

8. How does Spring know where to search for Components or Beans?
   What is a component scan?
   How do you define a component scan in XML and Java Configuration?
   How is it done with Spring Boot?

   Java Configuration -

```
@Configuration
@ComponentScan(basePackages = {
    "com.in28minutes.spring.example1.businessservice",
    "com.in28minutes.spring.example1.dataservice.stub" })
class SpringContext {
}
```

   XML Configuration -

```
229
230         <?xml version="1.0" encoding="UTF-8" standalone="no"?>
231         <beans xmlns="http://www.springframework.org/schema/beans"
232           xmlns:aop="http://www.springframework.org/schema/aop"
233           xmlns:context="http://www.springframework.org/schema/context"
234           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
235           xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
236                                 http://www.springframework.org/schema/tx
                                    http://www.springframework.org/schema/tx/spring-tx-3.0.x
                                    sd
237                                 http://www.springframework.org/schema/context
                                    http://www.springframework.org/schema/context/spring-con
                                    text-3.0.xsd
238                                 http://www.springframework.org/schema/aop
                                    http://www.springframework.org/schema/aop/spring-aop.xsd
                                    ">
239
240               <context:component-scan base-package="com.in28minutes.example"/>
241
242         </beans>
243
244     Spring Boot -
245
246         package com.in28minutes.spring.basics.springin5steps;
247
248         @SpringBootApplication
249         public class SpringIn5StepsApplication {
250         ...
251
252         ------------
253
254         package com.in28minutes.spring.basics.springin5steps;
255
256         @RunWith(SpringRunner.class)
257         @SpringBootTest
258         public class SpringIn5StepsApplicationTests {
259
260
261         Annotations @SpringBootApplication and @SpringBootTest will automatically
            intiate a scan on the package where they are in and also the sub-packages.
262
263 9.  What does @Component signify?
264     What does @Autowired signify?
265     What's the difference between @Controller, @Component, @Repository and @Service
      annotations in Spring?
266
267     @Component - Generic Component. signifies that this is a bean and it needs to be
      managed by Spring.
268
269     @Autowired - Spring should find the matching bean and wire the dependency in.
270
271     @Service - Business Service Facade
272
273     @Controller - Controller in MVC pattern
274
275     @Repository - encapsulating storage, retrieval, and search behavior typically from
      a relational database
276
277     We can use @Component on any layer i.e. Web, Business, or Data. So @Component is
      generic.
278     @Controller is specific to Web layer
279     @Repository is specific to Data layer
280     @Service is specific to Business layer
281
282     There also some specific features which Spring has attached with these annotations,
      like when every a JDBC exception occurs, due to @Repository annotation it will be
      translated to Spring speciofic exception.
283
```

10. What is the default scope of a bean?
    Are Spring beans thread safe?
    What are the other scopes available?
    How is Spring's singleton bean different from Gang of Four Singleton Pattern?

    When we create a bean we can specify any of the below scopes -
        singleton - One instance per Spring Context. When we create a ApplicationContext then there is only one instance of that bean.
        prototype - New bean whenever requested. In a single ApplicationContext if we have 100 requests, then we will create 100 different beans.
        request - One bean per HTTP request. Web-aware Spring ApplicationContext. Applicable only in case of WebApplicationContext.
        session - One bean per HTTP session. Web-aware Spring ApplicationContext. Applicable only in case of WebApplicationContext. Eg. - store user-specific details.

    (Maintain details of user ascross multiple request, then we create a session)

    As by default scope of bean is singleton i.e. only one instance, so by default it is not thread safe as multiple threads can act on single bean at the same time.

    Notes:
        - The singleton scope is the default scope in Spring.
        - The Gang of Four defines Singleton as having one and only one instance per ClassLoader.
        - However, Spring singleton is defined as one instance of bean definition per container (per ApplicationContext, if we have multiple ApplicationContext inside same JVM then we will have multiple instance of that bean).

    Examples:

    ```
    @RequestScope
    @Component
    public class RequestScopedBean {
    ...

    ------

    @SessionScope
    @Component
    public class SessionScopedBean {
    ...

    ------

    <bean id="someBean" class="com.in28minutes.SomeBean"
                scope="prototype"/>
    ```

11. What are the different types of dependency injections?
    What is setter injection?
    What is constructor injection?
    How do you choose between setter and constructor injections?

    There are two types ofdependency injections - Setter and Constructor injections.

    Setter Injection:

    ```
        @Component
        public class TodoBusinessService {

          TodoDataService dataService;

          @Autowired
          public void setDataService(TodoDataService dataService) {
            this.dataService = dataService;
          }
    ```

```
  ...

------

//Through Reflection
@Component
public class TodoBusinessService {

  @Autowired
  TodoDataService dataService;
...
```

Setter inject happens through a setter.

Constructor Injection:

```
@Component
public class TodoBusinessService {

  TodoDataService dataService;

  @Autowired
  public TodoBusinessService(TodoDataService dataService) {
    super();
    this.dataService = dataService;
  }
```

Here Spring will constructor to do wiring, it will call constructor.

Constructor vs Setter Injection
https://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/
The Spring team generally advocates constructor injection as it enables one to implement application components as immutable objects and to ensure that required dependencies are not null. (means Spring recomments to use constructor injection)

Constructor Injection for Mandatory Dependencies (mandatory dependencies i.e. that that need for sure)
Setter Injection for Optional Dependencies.

Furthermore constructor-injected components are always returned to client (calling) code in a fully initialized state.
As a side note, a large number of constructor arguments is a bad code smell.

12. What are the different options available to create Application Contexts for Spring?
    What is the difference between XML and Java Configurations for Spring?
    How do you choose between XML and Java Configurations for Spring?

    XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                      http://www.springframework.org/schema/tx
                      http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
                      http://www.springframework.org/schema/context
                      http://www.springframework.org/schema/context/spring-context-3.0.xsd
                      http://www.springframework.org/schema/aop
                      http://www.springframework.org/schema/aop/spring-aop.xsd
                      ">
</beans>
```

In between <beans> tag we will define all the beans we want and we can lauch it

using below code:

```
ApplicationContext context = new ClassPathXmlApplicationContext(new String[]
{"BusinessApplicationContext.xml", "Other-Configuration.xml"});
```

Java:

```
@Configuration
class SpringContext {
}
```

We need to put @Configuration, and then we can define beans using @Bean annotation and we can launch them using below code:

```
ApplicationContext ctx =  new
AnnotationConfigApplicationContext(SpringContext.class);
```

With Spring Boot, we are slowly moving towards complete Java Configuration.

13. How does Spring do Autowiring?
What are the different kinds of matching used by Spring for Autowiring?

Autowiring:
- byType
- byName
- constructor - similar to byType, but through constuctor. It finds matching constructor by type.

by Type - Class or Interface:

```
@Component
public class ComplexAlgorithmImpl {

  @Autowired
  private SortAlgorithm sortAlgorithm;
  ...

------

public interface SortAlgorithm {
  public int[] sort(int[] numbers);
}

------

@Component
public class QuickSortAlgorithm implements SortAlgorithm {

------
```

by Name:

```
@Component
public class ComplexAlgorithmImpl {

  @Autowired
  private SortAlgorithm quickSortAlgorithm;
  ...

------

public interface SortAlgorithm {
  public int[] sort(int[] numbers);
}

------
```

```
        @Component
        public class QuickSortAlgorithm implements SortAlgorithm {

        ------

        @Component
        public class BubbleSortAlgorithm implements SortAlgorithm {

        ------

    Constructor Injection:

        @Component
        public class TodoBusinessService {

          TodoDataService dataService;

          @Autowired
          public TodoBusinessService(TodoDataService dataService) {
            super();
            this.dataService = dataService;
          }


14. How do you debug problems with Spring Framework?
        - NoUniqueBeanDefinitionException
        - NoSuchBeanDefinitionException
    What is @Primary?
    What is @Qualifier?


    No matching Components:

        @Component
        public class ComplexAlgorithmImpl {

          @Autowired
          private SortAlgorithm sortAlgorithm;
          ...

         ------

        public interface SortAlgorithm {
          public int[] sort(int[] numbers);
        }

        ------

        public class QuickSortAlgorithm implements SortAlgorithm {

        ------

        public class BubbleSortAlgorithm implements SortAlgorithm {

        In above code we don't have @Component defined to QuickSortAlgorithm and
        BubbleSortAlgorithm. So spring won't find any beans. In such cases it will
        throw a NoSuchBeanDefinitionException.
        In some cases, it might happen that you have difined @Component to those
        classes, but have not defined @ComponentScan well. This might also result in
        same exception - NoSuchBeanDefinitionException.

    Typically problems:
        - @Component missing
        - or @ComponentScan not defined properly


    Exception - Two matching Components:

        @Component
```

```
public class ComplexAlgorithmImpl {

  @Autowired
  private SortAlgorithm sortAlgorithm;
  ...

 ------

public interface SortAlgorithm {
  public int[] sort(int[] numbers);
}

------
@Component
public class QuickSortAlgorithm implements SortAlgorithm {

------
@Component
public class BubbleSortAlgorithm implements SortAlgorithm {
```

Now here, spring will find ambiguity, first it will look by type, it will find two class of type SortAlgorithm - QuickSortAlgorithm and BubbleSortAlgorithm. So it will next look by name - sortAlgorithm, but again it won't find any bean with name SortAlgorithm of type SortAlgorithm. It will throw exception - NoUniqueBeanDefinitionException.
We can solve this by naming it properly - by using autowire by name.
Other option is to use @Primary.
Other option is to use @Qualifier.

Primary:

```
@Component
@Primary
public class BubbleSortAlgorithm implements SortAlgorithm {
    ...
```

We can say BubbleSortAlgorithm is our primary class. So now if spring finds ambiguity, it will use BubbleSortAlgorithm first.

Qualifier

```
@Component
public class ComplexAlgorithmImpl {

  @Autowired
  @Qualifier("mainAlgorithm")
  private SortAlgorithm sortAlgorithm;
  ...

 ------

@Component
@Qualifier("mainAlgorithm")
public class BubbleSortAlgorithm implements SortAlgorithm {
    ...
```

So need to specify @Qualifier on the dependency and give it a name. And we also need to specify same Qualifier name on the Component which we want to autowire in case of ambiguity.

15. What is CDI (Contexts and Dependency Injection)?
    Does Spring Support CDI?
    Would you recommend to use CDI or Spring Annotations?

    CDI:
    CDI is Java EE Dependency Injection Standard (JSR-330), just like JPA is a standard. CDI defines some annotation/APIs for Dependency Injection. This uses

some different annotation then Spring. Like below for Spring's @Autowired it uses @Inject annotation an so on.
Spring Supports most annotations that CDI provides -
- @Inject (@Autowired)
- @Named (@Component & @Qualifier) - @Named means we want CDI framework to manage this bean
- @Singleton (Defines a scope of Singleton)

This is just like JPA and Hibernate. JPA is a standard and Hibernate follows JPA. JPA has different annotations which Hibernate supports with some different annotations.
Same way CDI is standard which is supported by Spring.

We can use either of these.

16. What are the major features in different versions of Spring?
What are new features in Spring Framework 4.0?
What are new features in Spring Framework 5.0?

Notes
Spring 2.5 made annotation-driven configuration possible.
Spring 3.0 made great use of the Java 5 improvements in language.

Spring 4.0
First version to fully support Java 8 features.
Minimum version of Java to use Spring 4 is Java SE 6.
Introduced @RestController annotation
Spring 4.1 supports JCache (JSR-107) annotations

Spring 5.0
Functional web framework
Support for Jigsaw (Java Modularity)
Support for reactive programming
Support for Kotlin

17. What are important Spring Modules?

file:///C:/Users/inarajp/Desktop/temp/spring-interview-guide-master/spring-interview-guide-master/1.presentation/images/SpringModules.png

Core Container Modules - Beans, Core, Context, SpEL - Bean management, Autowiring, Dependency injection, ApplicationContext, Spring core, etc.

SpEL - Spring Expression Language

Data Access/Integration module - JDBC, ORM, OXM, JMS, Transactions

Spring has its own JDBC framework called Spring-JDBC

ORM - Object Relation Mapping, thus has good intregration with framework like Hibernate

OXM - Object XML Mapping module, thus has good intregration with framework like JAXB

JMS - Java Messaging Services

Transactions - Build in support for Transactions, can manage transactions for JPA, JDBC, Hibernate, etc.

Web module - WebSocket, Servlet, Portlet, Web

Spring has web MVC framework of its own called - Spring MVC - Can develop web applications easily, support for REST

Supoorts - Servlets, Portlets, Web Socket

Cross cutting concerns - Logging, exception handling, etc. These are concerns for all the layers present in the application.
AOP can be used to implement cross cutting concerns. Spring provides a basic aspect oriented programming module of its own, also spring has good support for AOP frameworks like Aspectj.

Spring-Test module - it helps use to write unit test and integration test.

So when we say there a new relese of spring with new version, there will be a new release for all these modules with that version. Modules will have the same versions as the base spring framework.

18. What are important Spring Projects?

All the Spring Modules have the same release version as the base Spring framework. So they are like part of Spring framework.

Spring Projects provide solutions to different problems.

Spring Projects:
- Spring Boot
- Spring Cloud
- Spring Data
- Spring Integration
- Spring Batch
- Spring Security
- Spring HATEOAS
- Spring Web Services
- Spring Session

Spring Boot - Popular framework to develop microservices. Very helpful to develop the applications quickly. They have feature like - Startup projects, auto configurations, accuators, etc.

Spring Cloud - Cloud navtive application, dynamically connect them, deploy them, etc.

Spring Data - Can connect to variety of databases.

Spring Integration - Addresses problems related to application integration.

Spring HATEOAS - In rest it might not be sufficient to just return the data. You would also want to return the related links, that would help the consumer to understand where to go from here. Spring-HATEOAS will help us do that easily.

There are many other Spring projects as well.

19. What is the simplest way of ensuring that we are using single version of all Spring related dependencies?

If we are making use of different modules, then we don't want to specify same versions to all.

BOM (Bill Of Material dependency) - Declares dependency version for all the spring modules.

Use a BOM (Bill Of Material dependency):

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-framework-bom</artifactId>
            <version>5.0.0.RELEASE</version>
            <type>pom</type>
            <scope>import</scope>
```

```
            </dependency>
        </dependencies>
    </dependencyManagement>

    -------

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
        </dependency>
    <dependencies>

    https://www.baeldung.com/spring-maven-bom


20. Name some of the design patterns used in Spring Framework?

    Design Patterns in Spring:
        - Front Controller pattern - Dispatcher Servlet - all browser request will
        first go to Dispatcher Servlet.
        - Prototype - Beans
        - Dependency Injection
        - Factory Pattern - Bean Factory & Application Context
        - Template Method
            org.springframework.web.servlet.mvc.AbstractController, JDBCTemplate, etc


21. What are some of the important Spring annotations you have used?

    Annotations:
        - @Component, @Service, @Repository, @Controller
        - @Autowired
        - @Primary
        - @Qualifier
        - @Configuration


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

    - Architecture - Flexible & No Restrictions
    - Design - Loosely Coupled
    - Code - Easy Unit Testing
    - Features - Dependency Injection, IOC Container(Bean Factory & Application
    Context), Auto wiring, Component Scan
    - Spring Modules
    - Spring Projects

    Spring has good support for struts, jsps, AOP, Freemarker, etc. It integrates well
    with all of these. This is architectural flexibility. It doesn't restrict our
    choices.
    Whenever we develop a loosely coupled application with Spring its easy to unit test
    it.


23. MVC - What is Model 1 architecture?
    What is Model 2 architecture?
    What is Model 2 Front Controller architecture?

    Model 1 is most basic. In this when request comes from browser the request directly
    goes to JSPs. (in html action we will put .jsp page). All logic will be in JSP page
    - like calling the DB, populating the model, creating the view, etc everything
    happens in JSPs. There we no Servlets. Therefore this resulted into very complex
```

JSPs, difficult to maintain.

Hence we migrated to Model 2 architecture, where request first goes to a Servlet and Servlet makes sure that it gets all the data from db and model is ready and then it redirects it to view. And view makes use of model and display it in the view. So Servlets actted as a Controller here.

Future enhancement of MVC-pattern was Model 2 Front Controller. Here request instead of going to Servlet, will first go to Front Controller. Front controller gets all the request, it makes sure that the response is ready and sends it back to the browser. Now we have one place to control everything that comes from the outside world. So if we want to make our application secure then we will implement security at the Front Controller level. Thus if our Front controller is secure our entire application below it will also be secure.
Front Controller based on the request will identify the right Servlet and then when Servlet will execute (make calls to db, populate the models) and return the models to Front Controller. Front Controller will then identify which view to render. And once the view is renderred it will send that information back to browser.

Check diagram at -
C:\Users\inarajp\Desktop\temp\spring-interview-guide-master\spring-interview-guide-master\1.presentation\images

24. Can you show an example controller method in Spring MVC?
Can you explain a simple flow in Spring MVC?
What is a ViewResolver?
What is Model?
What is ModelAndView?
What is a RequestMapping?

Controller Method:

```
@RequestMapping(value = "/list-todos", method = RequestMethod.GET)
public String listTodos(ModelMap model) {
  model.addAttribute("todos", service.retrieveTodos(retrieveLoggedinUserName()));
  return "list-todos";
}
```

ViewResolver:

```
<bean class =
"org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="prefix">
    <value>/WEB-INF/views/</value>
  </property>
  <property name="suffix">
    <value>.jsp</value>
  </property>
</bean>
```

Model vs ModelAndView:

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public String showLoginPage(ModelMap model) {
  model.put("name", "in28Minutes");
  return "welcome";
}

@RequestMapping(value = "/", method = RequestMethod.GET)
public ModelAndView showLoginPage() {
  ModelAndView mv = new ModelAndView();
  mv.addObject("name", "in28Minutes");
  mv.setViewName("welcome");
}
```

Check diagram.

DispatcherServlet knows about all the Controllers, all the urls that are mappedin those controllers, etc. It know all the handler mapping. Controller then will populate the model. It wil then return name of the view back to DispatcherServlet. Then DispatcherServlet asks ViewResolver to map the view name to the physical view page. ViewResolver will then add a prefix and suffix and map return to DispatcherServlet. DispatcherServlet will then pass the model to view, which will render and this renderred view will be send as response by DispatcherServlet.

25. What is Dispatcher Servlet?
How do you set up Dispatcher Servlet?

```
<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/todo-servlet.xml</param-value>
      //location of spring context (beans).
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>                                          //all request will go to dispatcher
</servlet-mapping>
```

In Spring boot we don't have to do this configuration at all. Because spring boot has a feature called Auto-Configuration, where in as soon as we add spring-web-mvc-jar in a classpath, it would know that you are developing a web application and it will auto configure a DispatcherServlet for us.

26. What is a form backing object?
How is validation done using Spring MVC?
What is BindingResult?
How do you map validation results to your view?
What are Spring Form Tags?

form backing form is basically the server side representation of the form.

Show New Todo Page:

```
    // This mapping method is called for first time display of the form. Thus form fields will be filled with default values or will be empty.

    @RequestMapping(value = "/add-todo", method = RequestMethod.GET)
    public String showTodoPage(ModelMap model) {

      model.addAttribute("todo", new Todo(0, retrieveLoggedinUserName(), "", new Date(), false));                    //todo => form backing object
      return "todo";
    }
```

todo.jsp:

```
    <form:form method="post" commandName="todo">
    //commandName="todo" => mapping the form backing object.
      <fieldset>
        <form:label path="desc">Description</form:label>
        <form:input path="desc" type="text"/>
        <form:errors path="desc"/>
      </fieldset>
      <fieldset>
        <form:label path="targetDate">Target Date</form:label>
        <form:input path="targetDate" type="text" />
        <form:errors path="targetDate"/>
```

```
            </fieldset>
            <input type="submit" value="Submit" />
        </form:form>

    Add Todo:

        @RequestMapping(value = "/add-todo", method =
        RequestMethod.POST)                    //value = "/add-todo", is for POST, above
        was for GET. When submit is pressed POST will happen.
        public String addTodo(ModelMap model, @Valid Todo todo, BindingResult result) {

          if (result.hasErrors()) {
            return "todo";
          }

          service.addTodo(retrieveLoggedinUserName(), todo.getDesc(), new Date(), false);

          model.clear();
          return "redirect:list-todos";
        }

    Todo.java:

        public class Todo {

          private int id;

          private String user;

          @Size(min = 6, message = "Enter atleast 6 characters")
          private String desc;

        ...

    For validation we need to use Java validation API, hibernate validator is
    implementation if Java validation API.  @Size and @Valid. We add validation on
    beans. @Valid will validate whatever validations we have added on the bean.
    Validation gets invoked and the result of validation gets stored into BindingResult
    We display validation error on jsp using spring's <form:errors tag.


27. What is a Path Variable?
    What is a Model Attribute?
    What is a Session Attribute?

    Path Variable:

        URI - http://localhost:8080/todos/1

        @RequestMapping(value = "/todos/{id}")
        public Todo retrieveTodo(@PathVariable int id) {
          return service.retrieveTodo(id);
        }

    Model Attribute:

        @ModelAttribute
        public void addAttributes(Model model) {
            model.addAttribute("options", Arrays.asList("Option 1","Option 2","Option
            3" ));
        }

        - Indicates the purpose of that method is to add one or more model attributes.
        - Invoked before @RequestMapping methods.
        - Used to fill the model with commonly needed attributes
            Drop down values for form
        - It will be available to all the methods in the controller

    @SessionAttributes:
```

List the names of model attributes which should be transparently stored in the session or some conversational storage.

```
@SessionAttributes("name")
public class TodoController {
    ...
```

Example name of logged in user.

When developing web applications, we often need to refer to the same attributes in several views. A good location to store those attributes is in the user's session.

@SessionAttributes or @SessionAttribute example:

@SessionAttributes annotation is used to store the model attribute in the session. This annotation is used at controller class level.

```
@SessionAttributes("user")
public class LoginController {

    @ModelAttribute("user")
    public User setUpUserForm() {
        return new User();
    }
}
```

In the above code snippet, the model attribute 'user' will be added to the session if the name attribute of the @ModelAttribute and @SessionAttributes annotations is same.

@SessionAttribute annotation is used to retrieve the existing attribute from session that is managed globally and it is used at method parameter as shown follows.

```
@GetMapping("/info")
public String userInfo(@SessionAttribute("user") User user) {
    //...
    //...
    return "user";
}
```

https://www.boraji.com/spring-mvc-4-sessionattributes-example

28. What is a init binder?
How do you set default date format with Spring?

Setting default format of date. We want to set a format for the entire application. We don't want to keep setting date format for all the fields in the application.

```
@InitBinder
protected void initBinder(WebDataBinder binder) {
    SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
    binder.registerCustomEditor(Date.class, new CustomDateEditor(dateFormat, false));
}
```

Suppose we have a form and on submission of the form we don't want spring to bind a specific field in an object, then we use initBinder.

```
@InitBinder
protected void initBinder(WebDataBinder binder) {  //name of function can be anything, by its parameter should be WebDataBinder and with annotation @InitBinder

    binder.setDisabllowedFields(new String[] {"studentMobile"});
    //studentMobile will not be binded by spring
}
```

```
@RequestMapping(value...)
public ModelAndView submit(@ModelAttribute("student") Student stud){
//studentMobile won't be binded here..
    ...
}
```

Above is basic level example.

The concept of property editor -

Suppose we have -

```
class Student{
    ....

    Date studentDOB;
    ....
}
```

if user inputs date in format 2010/10/10 Spring will be able to bind it in Student object. If user enters date in format 2010****10****10 then spring won't bind it, it will give error.
So question is how to allow user to provide date in some customized format of your choice, say --> yyyy****MM****dd?
And answere is using property editor concept provided by spring.

```
@InitBinder
protected void initBinder(WebDataBinder binder) {

    binder.setDisabllowedFields(new String[] {"studentMobile"});

    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy****MM****dd");
    binder.registerCustomEditor(Date.class, "studentDOB", new
    CustomDateEditor(dateFormat, false));

    //For above line - // We are registering our own format with WebDataBinder. We
    are telling Spring MVC that, hey when whenever you are performing binding for
    Date.class and you will be performing it for field studentDOB then you simply
    for this - dateFormat format for performing that task.
}
```

Spring mvc internally uses property editors to perform data binding tasks. What this means is Spring mvc has many classes which it calls as a property editor class and CustomDateEditor is one of them. Other examples are FileEditor, ClassEditor, CustomNumberEditor, etc. Spring MVC uses these classes in order to perform type conversion while data binding. We can use a particular property editor class in order to customize data binding for a particular data type.

29. How do you implement common logic for Controllers in Spring MVC?
    What is a Controller Advice?
    What is @ExceptionHandler?
    How to handle exceptions for web applications?
    What are the important things to think about when implementing Exception Handling?
    How do you implement Specific Error Handling for a Spring MVC Controller?

    ExceptionController:

```
@ControllerAdvice                       //For all the controllers present in
application
public class ExceptionController {

 private Log logger = LogFactory.getLog(ExceptionController.class);

 @ExceptionHandler(value = Exception.class) //@ExceptionHandler is the advice which
 we are handling in @ControllerAdvice. We can directly put this in a controller to
 make it controller specific.
 public String handleException(HttpServletRequest request, Exception ex) {
```

```
            logger.error("Request " + request.getRequestURL() + " Threw an Exception", ex);
            return "error";
         }
        }


30. Why is Spring MVC so popular?

    Spring MVC:
        - Clear Seperation of Concerns
            - Dispatcher Servlet
            - View Resolver
            - View
            - Model

        All these are completely independent of one another.
        Helps use to develop the web application easily
        As well as Unit test it easily
        It is not only used to develop web applications but it is also used to develop
        RESTfull webservices.


31. What is Spring Boot?
    What are the important Goals of Spring Boot?
    What are the important Features of Spring Boot?

    Why Spring Boot?
        Spring based applications have a lot of configuration.
        When we use Spring MVC, we need to configure component scan, dispatcher
        servlet, a view resolver, web jars(for delivering static content) among other
        things.
        World is moving towards Microservices.
        We do not have a lot of time to set up 100 microservices.

    Spring Boot Goals:
        Quick Start to Spring
        Be opinionated
        Non functional features
        No code generation

    Spring Boot Features:
        - Auto Configuration -
            eg - if spring boot find spring mvc jar in classpath then it will assume
            that we are developing a web application and it will automatically
            configure all the required things like dispatcher servlet, view resolver,
            etc. Same way if we add a JPA jar, then we get auto configuration related
            to connecting the database.

        - Spring Boot Starter Projects
        - Spring Boot Actuator  - helps in monitoring our application
        - Embedded Server   - with spring boot application we can embed a server inside
        our deployable jar. Servers like tomcat, jetty can be included into our jar
        file and when we run application on the server then we don't need server
        installed separately.


32. Compare Spring Boot vs Spring?
    Compare Spring Boot vs Spring MVC?

    Spring:
        Most important feature of Spring Framework is Dependency Injection. At the core
        of all Spring Modules is Dependency Injection or IOC Inversion of Control.

    @RestController
    public class WelcomeController {

        private WelcomeService service = new WelcomeService();

        @RequestMapping("/welcome")
```

```
        public String welcome() {
            return service.retrieveWelcomeMessage();
        }
    }

    With Spring:

        @Component
        public class WelcomeService {
            //Bla Bla Bla
        }

        @RestController
        public class WelcomeController {

            @Autowired
            private WelcomeService service;

            @RequestMapping("/welcome")
            public String welcome() {
                return service.retrieveWelcomeMessage();
            }
        }

    Problems Spring Solves:
        Problem 1 : Duplication/Plumbing Code   --like very less code for jdbc or db
        Problem 2 : Good Integration with Other Frameworks.     --like if we want to
        integrate struts, hibernate, JMS, etc

    Spring MVC:
        Spring MVC Framework provides decoupled way of developing web applications.
        With simple concepts like Dispatcher Servlet, ModelAndView and View Resolver,
        it makes it easy to develop web applications.

    Spring Boot:
        - Eliminates all configuration needed by Spring and Spring MVC and auto
        configures it
            - No need for @ComponentScan. Default Component Scan.
            - No need to configure DispatcherServlet
            - No need to configure a Data Source, Entity Manager Factory or Transaction
            Manager.

    Spring Boot Thinking:
        Can we bring more intelligence into this? When a spring mvc jar is added into
        an application, can we auto configure some beans automatically?

        Spring Boot looks at -
            Frameworks available on the CLASSPATH
            Existing configuration for the application.
            Based on these, Spring Boot provides Auto Configuration.


33. What is the importance of @SpringBootApplication?

    @SpringBootApplication is replaced with -

        @SpringBootConfiguration
        @EnableAutoConfiguration
        @ComponentScan
        public @interface SpringBootApplication {
            ......


34. What is Auto Configuration?
    How can we find more information about Auto Configuration?


    Spring Boot looks at :
        a) Frameworks available on the CLASSPATH
```

b) Existing configuration for the application. Based on these, Spring Boot provides basic configuration needed to configure the application with these frameworks. This is called Auto Configuration.

Application Startup Log:
Mapping servlet: 'dispatcherServlet' to [/]

Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity <java.util.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web. BasicErrorController.error(javax.servlet.http.HttpServletRequest)

Mapped URL path [/webjars/ * *] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]

All these logs are part of spring boot autoconfiguration.

What does auto configuration works, where is it implementation:
It is implemented in spring-boot-autoconfigure.jar

To get more details -
- Turn on Debug logging logging.level.org.springframework: DEBUG - in your application.properties file
- Use Spring Boot Actuator

35. What is an embedded server? Why is it important?
What is the default embedded server with Spring Boot?
What are the other embedded servers supported by Spring Boot?

We are moving towards microservices and we are developing lot of small applications. So to deploy these small applications on server we would like to have as less pre-requiesite as possible.

Embedded Server:
- Server is embedded as part of the deployable - jar or application. We can directly run this jar.
- Removes the need to have the server pre-installed on the deployment environment.
- Default is Tomcat.
- Spring Boot also supports Jetty and UnderTow.

Switching to Jetty:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

36. What are Starter Projects?
Can you give examples of important starter projects?

Spring Boot Documentation:
- Starters are a set of convenient dependency descriptors that you can include in your application. You get a one-stop-shop for all the Spring and related technology that you need, without having to hunt through sample code and copy paste loads of dependency descriptors.

- For example, if you want to get started using Spring and JPA for database access, just include the spring-boot-starter-data-jpa dependency in your project, and you are good to go.

Starters:
- spring-boot-starter-web-services - SOAP WebServices
- spring-boot-starter-web - Web & RESTful applications
- spring-boot-starter-test - Unit, Integration Testing
- spring-boot-starter-jdbc - Traditional JDBC
- spring-boot-starter-hateoas - HATEOAS features
- spring-boot-starter-security - Authentication and Authorization using Spring Security
- spring-boot-starter-data-jpa - Spring Data JPA with Hibernate
- spring-boot-starter-cache - Enabling Spring Framework's caching support
- spring-boot-starter-data-rest - Expose Simple REST Services using Spring Data REST - if we want to expose a JPA entity to outside world using RESTful web service. It helps us to take the spring repositories and make them available as a RESTful web service.
- 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-log4j2 - Logging using Log4j2

37. What is Starter Parent?
What are the different things that are defined in Starter Parent?
How does Spring Boot enforce common dependency management for all its Starter projects?

For any Spring boot application we have a spring-boot-starter-parent defined as a parent pom.

Starter Parent:

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.0.RELEASE</version>
</parent>
```

If we look at the spring-boot-starter parent we can see that it has parent of it own called spring-boot-dependencies and it defines lot of different things.

Inside Starter Parent:

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-dependencies</artifactId>
  <version>2.0.0.RELEASE</version>
  <relativePath>../../spring-boot-dependencies</relativePath>
</parent>

<java.version>1.6</java.version>
<resource.delimiter>@</resource.delimiter> <!-- delimiter that doesn't clash with Spring ${} placeholders -->
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
<maven.compiler.source>${java.version}</maven.compiler.source>
<maven.compiler.target>${java.version}</maven.compiler.target>
```

Inside Spring Boot Dependencies:

```
<ehcache3.version>3.1.1</ehcache3.version>
...
<h2.version>1.4.192</h2.version>
```

```
        <hamcrest.version>1.3</hamcrest.version>
        <hazelcast.version>3.6.4</hazelcast.version>
        <hibernate.version>5.0.9.Final</hibernate.version>
        <hibernate-validator.version>5.2.4.Final</hibernate-validator.version>
        <jackson.version>2.8.1</jackson.version>
        ....
        <jersey.version>2.23.1</jersey.version>
        <spring-security.version>4.1.1.RELEASE</spring-security.version>
        <tomcat.version>8.5.4</tomcat.version>
        <xml-apis.version>1.4.01</xml-apis.version>
```

In spring-boot-dependencies all the version for dependencies are defined. So when we are using spring boot we don't have to define version of any dependencies because all the version are defined in spring-boot-dependencies.
On of the problems with just spring people use to face was, what version of dependency(hibernate, jackson, etc) will play very well with the spring version I am using. We need compatable version or else there might be conflicts. This is what spring boot eliminates by specifying all dependencies versions. There versions for more than 250 jars defined.
So just need to specify the dependencies without version, based on the current spring version the spring boot will download the appropriate version.
All these things we get with spring-boot starter parent.

38. What is Spring Initializr?

    Spring Initializr http://start.spring.io/ is great tool to bootstrap your Spring Boot projects.

39. What is application.properties?
    What are some of the important things that can customized in application.properties?

    application.properties is used to configure your Spring Boot Application

    Example Configuration:

```
        logging.file=
        logging.level.*=
        spring.autoconfigure.exclude=
        spring.profiles.active=
        server.error.path=/error
        server.port=8080
        spring.http.converters.preferred-json-mapper=jackson

        spring.jackon.serialization.write-dates-as-timestamos=false         //don't serialize dates as timestamp

        management.security.enabled=fasle       //when we use accuator this is enabled bu default.
        security.basic.enabled=true
        security.user.name=username
        security.user.password=password

        spring.jpa.show-sql=true
        spring.h2.console.enabled=true
```

    https://docs.spring.io/spring-boot/docs/1.1.6.RELEASE/reference/html/common-application-properties.html

    https://docs.spring.io/spring-boot/docs/1.4.x/reference/html/common-application-properties.html

40. How do you externalize configuration using Spring Boot?
    How can you add custom application properties using Spring Boot?

What is @ConfigurationProperties?

application.properties is the way to externalize configuration.

.....

yaml(Yet Another Mark-up Language) syntax -

```
logging:
    level:
        org.springframework: DEBUG
app:
    name: In28Minutes
        description: ${app.name} is your first Spring Boot application
```

.....

```
import org.springframework.boot.context.properties.ConfigurationProperties;

@Component
@ConfigurationProperties("basic")
public class BasicConfiguration {
    private boolean value;
    private String message;
    private int number;
```

.....

```
@Autowired
private BasicConfiguration configuration;

@RequestMapping("/dynamic-configuration")
public Map dynamicConfiguration() {
    // Not the best practice to use a map to store differnt types!
    Map map = new HashMap();
    map.put("message", configuration.getMessage());
    map.put("number", configuration.getNumber());
    map.put("key", configuration.isValue());
    return map;
}
```

.....

application.properties:

```
    basic.value= true
    basic.message= Dynamic Message
    basic.number= 100
```

application.yaml:

```
    basic:
       value: true
       message: Dynamic Message YAML
       number: 100
```

Advantage:

Type Safety

```
        **************************
        APPLICATION FAILED TO START
        **************************
```

```
            Description:

            Binding to target
            com.in28minutes.springboot.configuration.BasicConfiguration@391b8545
            failed:

            Property: basic.number
            Value: ABC
            Reason: Failed to convert property value of type [java.lang.String]
            to required type [int] for property 'number'; nested exception is
            org.springframework.core.convert.ConverterNotFoundException:
            No converter found capable of converting from
            type [java.lang.String] to type [int]

            Action:
            Update your application's configuration


    Good Practice:
        Design all your application configuration using ConfigurationProperties


41. What is a profile?
    How do you define beans for a specific profile?
    How do you create application configuration for a specific profile?


    How do you have different configuration for different environments?

    Profile:
        application-dev.properties
        application-qa.properties
        application-stage.properties
        application-prod.properties
        application.properties              //define all common properties here.

    Profile:
        Based on the active profile, appropriate configuration is picked up.
        Used to Configure Resources - Databases, Queues, External Services

    Setting a profile:
        - Using -Dspring.profiles.active=prod in VM Arguments
        OR
        - In application.properties, spring.profiles.active=prod

    Profiles in code:   @Profile("dev") on a bean

        @Profile("dev")
        @Bean
        public String devBean() {
          return "I will be available in profile dev";
        }

        @Profile("prod")
        @Bean
        public String prodBean() {
          return "I will be available in profile prod";
        }


42. What is Spring Boot Actuator?
    How do you monitor web services using Spring Boot Actuator?
    How do you find more information about your application envrionment using Spring
    Boot?

    Spring Boot Actuator: Is a application which we can use to monitor our application.
    We just need to add its dependency, thats it. It exposes a lot of URIs / services -
```

Monitoring
/env, /metrics, /trace, /dump
/beans, / autoconfig, /configprops, /mappings

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Actuator provides lot of monitoring facilities around your services.

We also would like to see the services which are provided by actuator in a browser. For this we add below dependency -

```
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-rest-hal-browser</artifactId>
</dependency>
```

HAL is a specific format - Hypertext Application Language. HAL is a simple format that gives a consistent and easy way to hyperlink between resources in your API. spring-boot-starter-actuator apis are in HAL format. So what HAL does it it looks at apis, identifies the links and show them on screen.

Actuator URLs are also changing a lot along with spring-boot versions -
localhost:8080/actuator
localhost:8080/application

To enable few things we need to add a property:
management.endpoint.web.exposure.include=* //we are enabling a web exposure here. Enabling everything may impact the performance, so it better to enable only those that we need.

Now after localhost:8080/actuator or localhost:8080/application we will get lot more links.

If we just type localhost:8080 we will go to HAL browser. And after entering /actuator in Explorer input box we will get all links

httptrace - will show all the requests that are comming in, response, time taken, etc. of all previous request also. This will impact performance so can't use in production enviornment.

43. What is a CommandLineRunner?

CommandLineRunner:

Spring Documentation - interface used to indicate that a bean should run when it is contained within a SpringApplication

```
public interface CommandLineRunner {
  void run(String... args) throws Exception;
}
```

Once spring application context has launched up then, the code in run() method will be executed.
Can be used if we want to do something after application startup, like populate the data, configure somthing, etc.

44. What is Spring JDBC? How is different from JDBC?
What is a JdbcTemplate?
What is a RowMapper?

JDBC - Update Todo:

```
    Connection connection = datasource.getConnection();
```

```
1529
1530            PreparedStatement st = connection.prepareStatement("Update todo set user=?,
                desc=?, target_date=?, is_done=? where id=?");
1531
1532            st.setString(1, todo.getUser());
1533            st.setString(2, todo.getDesc());
1534            st.setTimestamp(3, new Timestamp(todo.getTargetDate().getTime()));
1535            st.setBoolean(4, todo.isDone());
1536            st.setInt(5, todo.getId());
1537
1538            st.execute();
1539            st.close();
1540            connection.close();
1541
1542            ........ lot of code to do simple update.
1543
1544
1545        Spring JDBC:
1546
1547            jdbcTemplate.update("Update todo set user=?, desc=?, target_date=?, is_done=?
                where id=?",
1548                        todo.getUser(), todo.getDesc(), new
                            Timestamp(todo.getTargetDate().getTime()), todo.isDone(),
                            todo.getId());
1549
1550            jdbcTemplate.update("delete from todo where id=?", id);
1551
1552
1553            jdbcTemplate has lot of othe ruseful methods. It reduces the number of lines of
                code, complexity of JDBC. No need to handle the exception with try-catch. There
                is no need to manage resource connections (st.close(), connection.close(), etc).
1554
1555
1556        Spring JDBC - RowMapper: helps in mapping a bean with a table.
1557
1558            new BeanPropertyRowMapper(Todo.class)       //if your bean name or name of
                properties in it exactly matches the column names, then we can use
                BeanPropertyRowMapper directly, or else we will have to do explicit mapping
                like below using mapRow()
1559
1560            ......
1561
1562            class TodoMapper implements RowMapper<Todo> {
1563
1564              @Override
1565              public Todo mapRow(ResultSet rs, int rowNum) throws SQLException {
1566
1567                Todo todo = new Todo();
1568
1569                todo.setId(rs.getInt("id"));
1570                todo.setUser(rs.getString("user"));
1571                todo.setDesc(rs.getString("desc"));
1572                todo.setTargetDate(rs.getTimestamp("target_date"));
1573                todo.setDone(rs.getBoolean("is_done"));
1574                return todo;
1575              }
1576            }
1577
1578            Once we create a RowMapper like above, we can reuse it anywhere like below code
                -
1579
1580                return jdbcTemplate.query(
1581                        "SELECT * FROM TODO where user = ?",
1582                        new Object[] { user }, new TodoMapper());
1583
1584                .......
1585
1586                return jdbcTemplate.queryForObject(
1587                        "SELECT * FROM TODO where id=?",
```

```
                    new Object[] { id }, new TodoMapper())


45. What is JPA?
    What is Hibernate?
    How do you define an entity in JPA?
    What is an Entity Manager?
    What is a Persistence Context?


    In above examples we were writting lot of queries. Some queries can get complex and
    big and generally these are written by java developers who might not be expert in
    database.
    JPA defines a mapping from our java object to a row in a table. We just need to
    define the proper mapping and JPA implementation(hibernate) will take care of
    generating the queries for us.

    Hibernate - JPA implementation. JPA is specification or standard or interface.

    JPA(Java Persistence API) - Update Todo:    Defining an entity

    @Entity
    @Table(name = "Todo")
    public class Todo {

      @Id
      @GeneratedValue(strategy = GenerationType.IDENTITY)
      private int id;

      private String user;

      private String desc;

      private Date targetDate;

      private boolean isDone;

    ......

    public class TodoJPAService implements TodoDataService {

      @PersistenceContext
      private EntityManager entityManager;

      @Override
      public void updateTodo(Todo todo) {
        entityManager.merge(todo);
      }

    ......

    Once we have defined all the entities, PersistenceContext is the one which is used
    to manage all those entities. So whatever changes we make has to go through
    PersistenceContext.
    We can access the PersistenceContext using EntityManager.


46. How do you map relationships in JPA?
    What are the different types of relationships in JPA?
    How do you define One to One Mapping in JPA?
    How do you define One to Many Mapping in JPA?
    How do you define Many to Many Mapping in JPA?

    One to One Relationship:

        @Entity
        public class Passport {

          ....
```

```
            //we are doing mappedBy="passport" because we want passport_id column in
            Student table. If we remove it we will have both, student_id in Passport
            table and also passport_id in Student table and this is not what we want, in
            one to one we would want one of the table to own the relationship.

            @OneToOne(fetch = FetchType.LAZY, mappedBy = "passport")
            private Student student;

        ..............

        @Entity
        @Table(name = "Student")
        public class Student {

            @OneToOne
            private Passport passport;

        ........
```

One to Many Relationship:

```
        @Entity
        public class Project {
            @OneToMany(mappedBy = "project")
            private List<Task> tasks;

        .....

        @Entity
        public class Task {
            @ManyToOne
            @JoinColumn(name="PROJECT_ID")
            private Project project;

        .....
```

Many to Many Relationship: Third table is created.

```
        @Entity
        public class Project {

            @ManyToMany
            // @JoinTable(name="STUDENT_PROJ",
            // joinColumns=@JoinColumn(name="STUDENT_ID"),
            // inverseJoinColumns=@JoinColumn(name="PROJECT_ID"))
            private List<Student> students;

        .....

        public class Student {
        @ManyToMany(mappedBy = "students")
            private List<Project> projects;
```

47. How do you define a datasource in a Spring Context?
    What is the use of persistence.xml
    How do you configure Entity Manager Factory and Transaction Manager?
    How do you define transaction management for Spring – Hibernate integration?

Defining a Data Source:

```
        #HSQL in-memory db
        db.driver=org.hsqldb.jdbcDriver
        db.url=jdbc:hsqldb:mem:firstdb
        db.username=sa
        db.password=
```

```
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
    destroy-method="close">
        <property name="driverClass" value="${db.driver}" />
        <property name="jdbcUrl" value="${db.url}" />
        <property name="user" value="${db.username}" />
        <property name="password" value="${db.password}" />
    </bean>


Configuring Hibernate: src\main\resources\config\hibernate.properties

    hibernate.dialect=org.hibernate.dialect.HSQLDialect
    hibernate.show_sql=false
    hibernate.format_sql=false
    hibernate.use_sql_comments=true


persistence.xml: src\main\resources\META-INF\persistence.xml - Used to configure
our persistence unit. JPA mandates that we will need persistence.xml

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

    <persistence xmlns="http://java.sun.com/xml/ns/persistence"
      version="2.0">
      <persistence-unit name="hsql_pu" transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <properties>
          <property name="hibernate.dialect"
          value="org.hibernate.dialect.HSQLDialect" />
          <property name="hibernate.connection.url" value="jdbc:hsqldb:mem:firstdb"
          />
          <property name="hibernate.connection.driver_class"
          value="org.hsqldb.jdbcDriver" />
          <property name="hibernate.connection.username" value="sa" />
          <property name="hibernate.connection.password" value="" />
        </properties>
      </persistence-unit>
    </persistence>


Configure Entity Manager Factory and Transaction Manager:

    <bean
    class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"
    id="entityManagerFactory">
        <property name="persistenceUnitName" value="hsql_pu" />
        <property name="dataSource" ref="dataSource" />
    </bean>

    <bean id="transactionManager"
    class="org.springframework.orm.jpa.JpaTransactionManager">
        <property name="entityManagerFactory" ref="entityManagerFactory" />
        <property name="dataSource" ref="dataSource" />
    </bean>

    <tx:annotation-driven transaction-manager="transactionManager"/>


Making Service Transactional:

    @Service
    public class StudentService {

      @Autowired
      StudentRepository service;

      @Transactional
      public Student insertStudent(Student student) {
        return service.insertStudent(student);
```

```
  }

.........

@Service
@Transactional
public class StudentService {

  @Autowired
  StudentRepository service;

..........
```

48. What is Spring Data?
    What is the need for Spring Data?
    What is Spring Data JPA?

    Duplication in JPA Repositories: Passport Repository

```
@PersistenceContext
private EntityManager entityManager;

public Passport getPassport(final long id) {
  Passport passport = entityManager.find(Passport.class, id);
  return passport;
}

public Passport createPassport(Passport passport) {
  return entityManager.merge(passport);
}
```

    Duplication in JPA Repositories: Student Repository

```
@PersistenceContext
private EntityManager entityManager;

public Student retrieveStudent(final long id) {
  return entityManager.find(Student.class, id);
}

public Student createStudent(Student student) {
  return entityManager.merge(student);
}
```

    So there is lot of duplication in the Passport repository and in Student repository

    Explosion of Data Stores:
        Variety of Big Data Stores

    Spring Data:
        Common Abstractions to store and retrieve data from data stores
        Independent of type of data store - it doesn't matter if we are talking to relation database, or big data store

    Spring Data JPA:
        Extends Spring Data for connecting to JPA

    Using Spring Data JPA:

```
public interface StudentRepository extends CrudRepository<Student, Long> {
}

public interface PassportRepository extends CrudRepository<Passport, Long> {
}
```

49. What is a CrudRepository?
What is a PagingAndSortingRepository?

CrudRepository:

```
public interface CrudRepository<T, ID> extends Repository<T, ID> {
  <S extends T> S save(S entity);
  Optional<T> findById(ID id);
  boolean existsById(ID id);
  Iterable<T> findAll();
  void deleteById(ID id);
  long count();
  //Other Methods
}
```

PagingAndSortingRepository: Pagination and Sorting:

```
public interface PagingAndSortingRepository<T, ID> extends CrudRepository<T,
ID> {
  Iterable<T> findAll(Sort sort);
  Page<T> findAll(Pageable pageable);
}
```

Using PagingAndSortingRepository:

```
Sort sort = new Sort(Sort.Direction.DESC,"field_name");
passportRepository.findAll(sort);
//Page Size - 10
PageRequest pageable = new PageRequest(0,10);
Page<Passport> page = passportRepository.findAll(pageable);
System.out.println(userPage.getTotalPages());
System.out.println(userPage.nextPageable());
```

50. How does Spring Framework Make Unit Testing Easy?
What is Mockito?
What is your favorite mocking framework?
How do you do mock data with Mockito?
What are the different mocking annotations that you worked with?

```
public class SomeBusinessImpl {

  private DataService dataService;

  //Constructor - public SomeBusinessImpl(DataService dataService)

  int findTheGreatestFromAllData() {

    int[] data = dataService.retrieveAllData();
    int greatest = Integer.MIN_VALUE;

    for (int value : data) {
      if (value > greatest) {
        greatest = value;
      }
    }
    return greatest;
  }

}
```

Basic Mocking:

```
@Test
public void testFindTheGreatestFromAllData() {
```

```
            //we don't want database dependency, because if its down unit test will fail.
            If we talk with real object and communicate with db then it becomes a
            integration test. It is no longer a Unit test.
            DataService dataServiceMock = mock(DataService.class);

            when(dataServiceMock.retrieveAllData())
                    .thenReturn(new int[] { 24, 15, 3 });

            SomeBusinessImpl businessImpl = new SomeBusinessImpl(dataServiceMock);

            int result = businessImpl.findTheGreatestFromAllData();

            assertEquals(24, result);
          }


      Using Annotations:

          @RunWith(MockitoJUnitRunner.class)
          public class SomeBusinessMockAnnotationsTest {

            @Mock
            DataService dataServiceMock;

            @InjectMocks
            SomeBusinessImpl businessImpl;

            @Test
            public void testFindTheGreatestFromAllData() {
              when(dataServiceMock.retrieveAllData())
                      .thenReturn(new int[] { 24, 15, 3 });
              assertEquals(24, businessImpl.findTheGreatestFromAllData());
            }

          .........

      MockitoJUnitRunner will create a mock and will inject it into the SomeBusinessImpl.


  51. What is MockMvc?
      What is @WebMvcTest?
      What is @MockBean?
      How do you write a unit test with MockMVC?
      What is JSONAssert?


      Mock MVC Test with Spring Boot:

      public Question retrieveDetailsForQuestion(@PathVariable String surveyId,
      @PathVariable String questionId) {
          return surveyService.retrieveQuestion(surveyId, questionId);
      }
      .....

      @RunWith(SpringRunner.class)
      @WebMvcTest(value = SurveyController.class, secure = false)
      public class SurveyControllerTest {

        @Autowired
        private MockMvc mockMvc;

        @MockBean
        private SurveyService surveyService;

        @Test
        public void retrieveDetailsForQuestion() throws Exception {
            Question mockQuestion = new Question("Question1", "Largest Country in the
            World", "Russia", Arrays.asList("India", "Russia", "United States", "China"));

```

```
            Mockito.when(
                surveyService.retrieveQuestion(Mockito.anyString(),
                Mockito.anyString())).thenReturn(mockQuestion);

            RequestBuilder requestBuilder =
            MockMvcRequestBuilders.get("/surveys/Survey1/questions/Question1").accept(Media
            Type.APPLICATION_JSON);

            MvcResult result = mockMvc.perform(requestBuilder)
                                      .andReturn();

            String expected = "{id:Question1,description:Largest Country in the
            World,correctAnswer:Russia}";

            JSONAssert.assertEquals(expected, result.getResponse().getContentAsString(),
            false);

            // Assert
          }


52. How do you write an integration test with Spring Boot?
    What is @SpringBootTest?
    What is @LocalServerPort?
    What is TestRestTemplate?

    In integration test we launch entire application as it is and then check what
    result comes back.
    @SpringBootTest - launches up the real server
    @LocalServerPort - helps to get the port on which application is running
    @TestRestTemplate - To execute the request.


    Integration Test with Spring Boot:

        @RunWith(SpringRunner.class)
        @SpringBootTest(classes = Application.class, webEnvironment =
        SpringBootTest.WebEnvironment.RANDOM_PORT)    //we are specifying the random
        port to be used. We can also specify a port number.
        public class SurveyControllerIT {

          @LocalServerPort
          private int port;             //whatever the port is used will be autowired
          here. We will need it later to fire request.

        }
    .....

    @Test
    public void testRetrieveSurveyQuestion() {

      String url = "http://localhost:" + port + "/surveys/Survey1/questions/Question1";

      HttpHeaders headers = new HttpHeaders();
      headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));

      HttpEntity<String> entity = new HttpEntity<String>(null, headers);        //when
      we want to send a request with accept headers we need HttpEntity. HttpEntity
      allows us to create a request with headers.

      TestRestTemplate restTemplate = new TestRestTemplate();
      ResponseEntity<String> response = restTemplate.exchange(url, HttpMethod.GET,
      entity, String.class);

      String expected = "{id:Question1,description:Largest Country in the
      World,correctAnswer:Russia}";

      JSONAssert.assertEquals(expected, response.getBody(), false);
    }
```

53. What are cross cutting concerns?
How do you implement cross cutting concerns in a web application?
If you would want to log every request to a web application, what are the options you can think of?
If you would want to track performance of every request, what options can you think of?

Check the diagram.

When we talk about a web application there are multiple layer, like web, business, data, integration, etc. Each of these layer has its own purpose and specialization. But some these are commons amongs all these layer, like log, security, etc. These are concerns of all the layers in the application and they are called cross-cutting concerns.
Typically all these cross cutting concerns are implemented by Aspect Oriented Programming (AOP).

Example - If we want to log every request that is comming to the application - One of the option which Java EE provides is to configure a filter and then add methods to log the request and log the response back. Other way is AOP, saying we would want to configure all the calls to a particular controller.

We have separete notes on AOP. Check.

54. What is an Aspect and Pointcut in AOP?
What are the different types of AOP advices?
What is weaving?

```
@Component
class HiByeService {
  public void sayHi(String name) {
    System.out.println("Hi " + name);
  }

  public void sayBye() {
    System.out.println("Bye");
  }

  public String returnSomething() {
    return "Hi Bye";
  }
}

.......

@Aspect
@Component
class MyAspect {
  @Before("execution(* HiByeService.*(..))")                          //any method on
  HiByeService.* ...... (..) -> for any arguments
  public void before(JoinPoint joinPoint) {
    System.out.print("Before ");
    System.out.print(joinPoint.getSignature().getName());
    System.out.println(Arrays.toString(joinPoint.getArgs()));
  }

  @AfterReturning(pointcut = "execution(* HiByeService.*(..))", returning = "result")
  public void after(JoinPoint joinPoint, Object result)
  {                                 //result or return value will available to us in
  Object result
    System.out.print("After ");
    System.out.print(joinPoint.getSignature().getName());
    System.out.println(" result is " + result);
```

```
    }
  }

  .......

  @Around(value = "execution(* HiByeService.*(..))")
  public void around(ProceedingJoinPoint joinPoint) throws Throwable {
    long startTime = new Date().getTime();
    Object result = joinPoint.proceed();
    long endTime = new Date().getTime();
    System.out.print("Execution Time - " + (endTime - startTime));
    //calculating the entire execution time of method.
  }

  .......
```

If we have abean and if property value of bean is changed, even such kind of things we can find out using AspectJ.

56.

AOP concepts:

- Joinpoint -> is a specific result of one execution. If a method is called 100 times, we will have 100 different JoinPoints.
- Advice -> what you want to do, which method you want to execute. Above all AOP methods and code is advise, Around advice, Before advice, etc.
- Pointcut -> helps to identify what kind of methods we want to intercept. What exactly we want to intercept.
- Aspect -> Combination of Advice and Pointcut. Advice + Pointcut
- Weaving -> is just a process of making sure that these methods are getting called at the right instance or right time.
- Weaver -> is AOP framework like Spring AOP or AspectJ

With Spring AOP we can do basic weaving while with AspectJ we can do advanced weaving.

With Spring AOP we can only intercept method calls, with AspectJ we can do lot more.

Advice Types:

- Before advice
- After returning advice
- After throwing advice
- After (finally) advice - Always executed - similar to finally
- Around advice - Most powerful - Performance Logging

AspectJ vs Spring AOP:

AspectJ is a full fledged AOP framework
Advise objects not managed by the Spring container
Advise join points other than simple method executions
(for example, field get or set join points)

57. What is a Web Service?

Check diagram.

3 Keys:

Designed for machine-to-machine (or application-to-application) interaction
Should be interoperable - Not platform dependent - we can do this by making our request and response platform independent.
Should allow communication over a network

58. What is SOAP Web Service?
What is SOAP?
Waht is a SOAP Envelope?

What is SOAP Header and SOAP Body?

This is Not SOAP Web Service - check diagram - C:\Users\inarajp\Desktop\temp\spring-interview-guide-master\spring-interview-guide-master\1.presentation\images\Web_Service_Basic_Interaction_SOAP

This is SOAP Web Service - check diagram - C:\Users\inarajp\Desktop\temp\spring-interview-guide-master\spring-interview-guide-master\1.presentation\images\Web_Service_Basic_Interaction_SOAP_2

check diagram - C:\Users\inarajp\Desktop\temp\spring-interview-guide-master\spring-interview-guide-master\1.presentation\images\Web_Service_SOAP-Envelope.svg

SOAP (Simple Object Access Protocol):
    Format
        - SOAP XML Request
        - SOAP XML Response
    Transport
        - SOAP over MQ
        - SOAP over HTTP
    Service Definition
        - WSDL

SOAP-ENV: Envelope

SOAP-ENV: Header - Contains meta information, authetication information, etc.

SOAP-ENV: Body - Has the actual content

59. Can you give an example of SOAP Request and SOAP Response?
    What is a SOAP Header? What kind of information is sent in a SOAP Header?
    Can you give an example of a SOAP Header with Authentication information?

SOAP Request:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <ns2:GetCourseDetailsRequest xmlns:ns2="http://in28minutes.com/courses">
            <ns2:course>
                <ns2:id>1</ns2:id>
            </ns2:course>
        </ns2:GetCourseDetailsRequest>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <ns2:getCourseDetailsResponse xmlns:ns2="http://in28minutes.com/courses">
            <ns2:course>
                <ns2:id>1</ns2:id>
                <ns2:name>Spring</ns2:name>
                <ns2:description>10 Steps</ns2:description>
            </ns2:course>
        </ns2:getCourseDetailsResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Header:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Header>
```

```
              <wsse:Security
              xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurit
              y-secext-1.0.xsd" mustUnderstand="1">
                <wsse:UsernameToken>
                  <wsse:Username>user</wsse:Username>
                  <wsse:Password>password</wsse:Password>
                </wsse:UsernameToken>
              </wsse:Security>
            </Header>
            <Body>
              <GetCourseDetailsRequest xmlns="http://in28minutes.com/courses">
                <id>1</id>
              </GetCourseDetailsRequest>
            </Body>
          </Envelope>


          wsse is one of the implementation of the standards for SOAP security

60. What is WSDL (Web Service Definition Language)?
    What are the different parts of a WSDL?


    WSDL - Web Service Definition Language

    If we provide WSDL to any of the client, they will know every detail about how to
    call a web-service (odb-adapter)

    Diagram -
    C:\Users\inarajp\Desktop\temp\spring-interview-guide-master\spring-interview-guide-ma
    ster\1.presentation\images\Web_Services_WSDL_HighLevel

    All Operations - Which are webservices(methods), what is input to it
    Endpoint
    Request Structure
    Response Structure
    We can also import XSD, XSDs can define request and response sturctures.



    C:\Users\inarajp\Desktop\temp\spring-interview-guide-master\spring-interview-guide-ma
    ster\1.presentation\images\Web_Services_WSDL_LowLevel

    definitions
            - types(defines what are different xml structure which we are going to use)
                - element
                - conplexType

            - message (defines different requests and responses for different
            operations. The only thing that can be used as request or response elements
            are the one that are mapped as a message. We can't directly use types as
            request or response element, it has to be a messasge)

            -portType (defines different operations, maps request message and response
            message to a operation. Its like an interface - this webservice offers
            these services)
                - operation
                    - input
                    - output

            - binding (defines the implementation, how do you really call these
            services or operations defined in portType. Here we define style (document
            - to indicate that we are exchanging the complete XML request and response,
            Other option - RPC(Remote procedure call - like calling a procedure defined
            somewhere else)) and transport (http - web/internet, other option is MQ))
                - operation
                    - input
                    - output
```

- service (what is the location, how client will use it. Maps it to endpoint, what is the url to call web service)
    - port

61. What is Contract First Approach?
    What is an XSD?
    Can you give an example of an XSD?

    Contract: Service Definition specifying
    - Format of Request
    - Format of Response
    - Request/Response Structure
    - Transport used
    - Endpoint details

    Contract First - We define a contract first!

    With Spring Web Services, we define an XSD first

    XSD:
    XML Specification Document!
    How does a valid XML Look like?

    Request XML:

```
<GetCourseDetailsRequest xmlns="http://in28minutes.com/courses">
      <id>1</id>
</GetCourseDetailsRequest>
```

    XSD:

```
<xs:element name="GetCourseDetailsRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="id" type="xs:int" />
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

    Response XML:

```
<ns2:GetCourseDetailsResponse xmlns:ns2="http://in28minutes.com/courses">
    <ns2:CourseDetails>
        <ns2:id>1</ns2:id>
        <ns2:name>Spring</ns2:name>
        <ns2:description>10 Steps</ns2:description>
    </ns2:CourseDetails>
</ns2:GetCourseDetailsResponse>
```

    XSD:

```
<xs:element name="GetCourseDetailsResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="CourseDetails" type="tns:CourseDetails" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

    XSD:

```
<xs:complexType name="CourseDetails">
  <xs:sequence>
    <xs:element name="id" type="xs:int" />
    <xs:element name="name" type="xs:string" />
    <xs:element name="description" type="xs:string" />
  </xs:sequence>
</xs:complexType>
```

    Once XSDs are created, Spring web services provides us a feature where in we can generate WSDL out of our XSDs.

62. What is JAXB?
How do you configure a JAXB Plugin?

JAXB (Java API for XML Binding)

Convert from Java to SOAP XML

Request or input will be XML and Response or output will be also a XML.

So JAXB will convert XML(Request) to java objects and java objects to XML(Response).

JAXB defines a specification about how to do this kind of conversion.

JAXB Plugin:

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>jaxb2-maven-plugin</artifactId>
  <version>1.6</version>
  <executions>
    <execution>
      <id>xjc</id>
      <goals>
        <goal>xjc</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <schemaDirectory>${project.basedir}/src/main/resources</schemaDirectory>
    <outputDirectory>${project.basedir}/src/main/java</outputDirectory>
    <clearOutputDir>false</clearOutputDir>
  </configuration>
</plugin>
```

JAXB plugins takes a schema directory and it generates java class from it.

63. What is an Endpoint?
Can you show an example endpoint written with Spring Web Services?

Endpoint: Where are web services are exposed. It will accept the request, it will then call the service to execute the business logic, and it will send the response out.

```
@PayloadRoot(namespace = "http://in28minutes.com/courses", localPart =
"GetCourseDetailsRequest")
@ResponsePayload
public GetCourseDetailsResponse processCourseDetailsRequest(@RequestPayload
GetCourseDetailsRequest request) {

  Course course = service.findById(request.getId());

  if (course == null)
    throw new CourseNotFoundException("Invalid Course Id " + request.getId());

  return mapCourseDetails(course);
}
```

@PayloadRoot - defines what kind of requests it can handle, we are saying that any request with namespace as ... and localPart as ... can be handled by this service.
GetCourseDetailsRequest and GetCourseDetailsResponse are classes generated by JAXB.
@RequestPayload - will map the details from request to the GetCourseDetailsRequest parameter object. XML bound to java object.
@ResponsePayload - will map the object to response. Java object converted to XML response.

64. What is a MessageDispatcherServlet?
How do you configure a MessageDispatcherServlet?

MessageDispatcherServlet: Now know that in spring mvc DispatcherServlet acts as a front controller, all the request first comes to DispatcherServlet and from there it goes to respective controller. In case of spring web services MessageDispatcherServlet does exactly the same thing. The request first goes to MessageDispatcherServlet, it then looks at the request, looks at the namespace and messages details and then maps it to the appropriate endpoint method and call it.

Configuring the MessageDispatcherServlet -

```
@Bean
public ServletRegistrationBean messageDispatcherServlet(ApplicationContext context) {

  MessageDispatcherServlet messageDispatcherServlet = new MessageDispatcherServlet();
  messageDispatcherServlet.setApplicationContext(context);
  messageDispatcherServlet.setTransformWsdlLocations(true);

  return new ServletRegistrationBean(messageDispatcherServlet, "/ws/*");
}
```

MessageDispatcherServlet is actually a servlet, so in spring if we want to configure a servlet we would use ServletRegistrationBean.
ServletRegistrationBean accepts two things - Servlet and URI.

65. How do you generate a WSDL using Spring Web Services?

```
@Bean(name = "courses")
public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema coursesSchema) {

  DefaultWsdl11Definition definition = new DefaultWsdl11Definition();
  definition.setPortTypeName("CoursePort");
  definition.setTargetNamespace("http://in28minutes.com/courses");
  definition.setLocationUri("/ws");
  definition.setSchema(coursesSchema);
  return definition;
}

@Bean
public XsdSchema coursesSchema() {
  return new SimpleXsdSchema(new ClassPathResource("course-details.xsd"));
}
```

66. How do you implement error handling for SOAP Web Services?
What is a SOAP Fault?

Whenever an error occurs we would want to give a correct error response back to the client. And SOAP fault is an structure in which we can send error response back.

Endpoint:

```
@PayloadRoot(namespace = "http://in28minutes.com/courses", localPart =
"GetCourseDetailsRequest")
@ResponsePayload
public GetCourseDetailsResponse processCourseDetailsRequest(@RequestPayload
GetCourseDetailsRequest request) {

  Course course = service.findById(request.getId());

  if (course == null)
    throw new CourseNotFoundException("Invalid Course Id " + request.getId());

  return mapCourseDetails(course);
}
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Server</faultcode>                          //we can change this
            <faultstring xml:lang="en">Invalid Course Id 1000</faultstring>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>


@SoapFault(
     faultCode = FaultCode.CUSTOM,
     customFaultCode = "{http://in28minutes.com/courses}001_COURSE_NOT_FOUND")
public class CourseNotFoundException extends RuntimeException {
//custom exception class
}


<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header />
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode xmlns:ns0="http://in28minutes.com/courses">ns0:001_COURSE_NOT_FOUND</faultcode>
      <faultstring xml:lang="en">Invalid Course Id 1234</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>


We can also use FaultCode.CLIENT


67. What is REST?

    REpresentational State Transfer

    REST is a style of software architecture for distributed hypermedia systems.

    When we are talking about SOAP web services we are focussed on the format. We are focussed on using SOAP envolop, SOAP header and SOAP body. REST only sets some architectural guidelines and thats all. It does not worry about what format of request we are sending, etc.

    REST wants to make best use of HTTP. HTTP provides HTTP Methods, HTTP Status

    Diagram -

    Anything and everything is a resource.

    Key abstraction - Resource
    A resource has an URI (Uniform Resource Identifier)
        /users/Ranga/todos/1
        /users/Ranga/todos
        /users/Ranga

    We can perform operations on these resources.

    Example:
        Create a User - POST /users
        Delete a User - DELETE /users/1
        Get all Users - GET /users
        Get one Users - GET /users/1

    REST:
        Data Exchange Format
```

No Restriction. JSON is popular
Transport
Only HTTP
Service Definition
No Standard. WADL/Swagger/...

68. What are the Best Practices of RESTful Services?

Best Practices in RESTful Design -
- Consumer First - what kind of consumers we have, what is there language, what consumer wants, how will it consume it, is it a web app or mobile app, etc.
- Have good documentation
- Make best use of HTTP
- Request Methods
GET
POST
PUT
DELETE
- Response Status - Send the appropriate response status.
200 - SUCCESS
404 - RESOURCE NOT FOUND
400 - BAD REQUEST //validation error in a request
201 - CREATED //successful post request to create a resource
401 - UNAUTHORIZED
500 - SERVER ERROR
- No Secure Info in URI
- Use Plurals
Prefer /users to /user
Prefer /users/1 to /user/1

69. Can you show the code for an example Get Resource method with Spring REST?
What happens when we return a bean from a Request Mapping Method?
What is GetMapping and what are the related methods available in Spring MVC?

```
@GetMapping("/users")
public List<User> retrieveAllUsers() {
  return service.findAll();
}
```

@GetMapping is shortcut to @RequestMapping. Similar is @PostMapping, @PutMapping, @DeleteMapping.

We are returning a List<user> as a response, so how does this gets converted into a JSON or whatever format our response has. It happens due to MessageConverters. When we use Spring Boot to develop RESTful services, spring boot registers MessageConverters by default. The default MessageConverters for JSON format is Jackson. So when we return a List<User> it will be converted to JSON and JSON response is send back.

70. Can you show the code for an example Post Resource method with Spring REST?
Why do we use ResponseEntity in a RESTful Service?

```
@PostMapping("/users")
public ResponseEntity<Object> createUser(@Valid @RequestBody User user) {

  User savedUser = service.save(user);

  URI location = ServletUriComponentsBuilder
    .fromCurrentRequest()                                    //  /users
    .path("/{id}")                                           //  /users/{id}  - id will will replaced with value
    .buildAndExpand(savedUser.getId()).toUri();              //URI for new resource is created. /users/10

  return ResponseEntity.created(location).build();
```

```
        //ResponseEntity.created - indicates status of created.

    }
```

@RequestBody annotation will map POST request content with User object.

Later in code we have send a Created status as response and a URI for the created user in the response.

We needed @ResponseEntity because we wanted to send status as Created with URI of created resource, we didn't wanted Success.

71. What is HATEOAS?
    Can you give an Example Response for HATEOAS?
    How do we implement it using Spring?

HATEOAS: Hypermedia as The Engine of Application State.

Example:

    When requested for details of a facebook post, we return
        Link for actions to like, unlike or comment on the post

    Its not just showing the data, but also the actions which we can do with the data.
    When we return a response don't just return a data, also return what actions consumer can do with that data or consumer might be intrested in.

```
{
    "id": 1,
    "name": "Adam",
    "birthDate": "2017-07-19T09:26:18.337+0000",
    "_links": {
        "all-users": {
            "href": "http://localhost:8080/users"
        }
    }
}
```

We can implement this by add a spring starter and then using a Resource class.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-hateoas</artifactId>
</dependency>
```

```
@GetMapping("/users/{id}")
public Resource<User> retrieveUser(@PathVariable int id) {

  User user = service.findOne(id);

  Resource<User> resource = new Resource<User>(user);

  ControllerLinkBuilder linkTo =
  linkTo(methodOn(this.getClass()).retrieveAllUsers());      //URI mapped to method handler retrieveAllUsers will be added

  resource.add(linkTo.withRel("all-users"));

  return resource;
}
```

72. How do you document RESTful web services?
    Can you give a brief idea about Swagger Documentation?

How do you automate generation of Swagger Documentation from RESTful Web Services?
How do you add custom information to Swagger Documentation generated from RESTful Web Services?
What is Swagger-UI?

Documentation for RESTful web services. There is WADL/Swagger/OpenDocs, etc. for documenting RESTful web services.

OpenAPI Specification (formerly called the Swagger Specification).
The specification creates the RESTful contract for your API, detailing all of its resources and operations in a human and machine readable format for easy development, discovery, and integration.

```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.4.0</version>
</dependency>

<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.4.0</version>
</dependency>
```

Configure Swagger and enable it -

```
    @Configuration
    @EnableSwagger2
    public class SwaggerConfig {

      @Bean
      public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2);  //version of specification to be used
      }

    }
```

.....

Swagger will show following things -

- swagger version

- info - high level information about API - what kind of APIs we are offering, etc.
    - description of api
    - version
    - title
    - termOfServices
    - contact
    - license

- host - where we are hosting the api

- basepath

- tags - are something which we can use to group resources. So for each of our resource and resource methods we can assign tags.

- paths - shows the details of all the resources that we are exposing and the different operations that can be performed on each of these resources.
    /users: {...}
        - get: {
            - tags:{...}
            - summary:{...}
            - operationId:{...}

```
                    - consumes:{...}
                    - produces:{...}
                    - responses:{
                        - 200: {...}
                        ...
                    }
                }
                - post:{...}

            /error: {...}
                - get: {...}
                - post:{...}

            /helloworld: {...}
                - get: {...}
                - post:{...}


        - definitions - includes different elements that are used in out API, eg - what
        is inside User, etc.
            - User: {
                type: "object",
                properties:{
                    - birthdate: {...}
                }
            }
```

When we want to expose our swagger document to client then there are two ways -
- We can download it as json
- Or we can use Swagger-UI

.....

Customizing Swagger more:

```
public static final Contact DEFAULT_CONTACT = new Contact("Ranga Karanam",
"http://www.in28minutes.com", "in28minutes@gmail.com");

public static final ApiInfo DEFAULT_API_INFO = new ApiInfo( "Awesome API Title",
"Awesome API Description", "1.0", "urn:tos", DEFAULT_CONTACT, "Apache 2.0",
"http://www.apache.org/licenses/LICENSE-2.0");

private static final Set<String> DEFAULT_PRODUCES_AND_CONSUMES = new
HashSet<String>(Arrays.asList("application/json", "application/xml"));
```

now we can use above defined constants in our Docket bean -

```
@Bean
public Docket api() {
  return new Docket(DocumentationType.SWAGGER_2)
      .apiInfo(DEFAULT_API_INFO)
      .produces(DEFAULT_PRODUCES_AND_CONSUMES)
      .consumes(DEFAULT_PRODUCES_AND_CONSUMES);
}
```

Above constants in bean will add information in info, if we want to add more details in definitions for User properties then we need to do more in User Entity class -

```
@ApiModel(description="All details about the user.")
@Entity
public class User {

  @Size(min=2, message="Name should have atleast 2 characters")
  @ApiModelProperty(notes="Name should have atleast 2 characters")
  //This will also appear in Swagger doc.
  private String name;
```

```
  @Past
  @ApiModelProperty(notes="Birth date should be in the past")
  private Date birthDate;

.....
```

We can use many other annotaions to improve the documentation. Its present in swagger-annotations.jar

ctrl + 1 => shortcut to create undeclared something.

73. What is "Representation" of a Resource?
What is Content Negotiation?
Which HTTP Header is used for Content Negotiation?
How do we implement it using Spring Boot?
How do you add XML support to your RESTful Services built with Spring Boot?

A resource can have different representations
- XML
- HTML
- JSON

JSON is default when we develop in spring boot.

GET http://localhost:8080/users

```
[
    {
        "id": 2,
        "name": "Eve",
        "birthDate": "2017-07-19T04:40:20.796+0000"
    },
    {
        "id": 3,
        "name": "Jack",
        "birthDate": "2017-07-19T04:40:20.796+0000"
    }
]
```

Using Accept header we can tell that we want a XML response back. Accept application/xml or application/json

GET http://localhost:8080/users

- Accept application/xml

```
<List>
    <item>
        <id>2</id>
        <name>Eve</name>
        <birthDate>2017-07-19T10:25:20.450+0000</birthDate>
    </item>
    <item>
        <id>3</id>
        <name>Jack</name>
        <birthDate>2017-07-19T10:25:20.450+0000</birthDate>
    </item>
</List>
```

When we are sending a GET request then Accept header - application/xml or application/json
When we are sending a POST request then Content-type - application/xml or application/json

Representation is format we are using to represent a resource, it can XML or json.

Content Negotiation is , when we are seding Accept header - application/xml the

server response with XML, when we are Accept header - application/json then it responses json. Its the negotiation happening between server and client, the client says I want it in this format and server sends in that format.

Add XML support -

```
<dependency>
  <groupId>com.fasterxml.jackson.dataformat</groupId>
  <artifactId>jackson-dataformat-xml</artifactId>
</dependency>
```

When we add above dependency spring will automatically confingure the XML MessageConverters.

74. How do you implement Exception Handling for RESTFul Web Services?
What are the different error status that you would return in RESTful Web Services?
How would you implement them using Spring Boot?
How do you handle Validation Errors with RESTful Web Services?

Check code in his git...

Response Status:
- 200 - SUCCESS
- 201 - CREATED
- 404 - RESOURCE NOT FOUND
- 400 - BAD REQUEST
- 401 - UNAUTHORIZED
- 500 - SERVER ERROR

GET http://localhost:8080/users/1000 - Get request to a non existing resource.

.....in code.....

```
if(user==null)
    throw new UserNotFoundException("id-" + id);            //This is send response like below -
```

.....

```
{
  "timestamp": "2017-07-19T05:28:37.534+0000",
  "status": 500,                                            //it is 500, it should be 404
  "error": "Not Found",
  "message": "id-500",
  "path": "/users/500"
}
```

To get status of 404 we need to add a annotation on UserNotFoundException class -

```
@ResponseStatus(HttpStatus.NOT_FOUND)
public class UserNotFoundException extends RuntimeException {
```

.....

GET http://localhost:8080/users/1000 - Get request to a non existing resource. - Default Spring Boot Structure -

```
{
  "timestamp": "2017-07-19T05:28:37.534+0000",
  "status": 404,
  "error": "Not Found",
  "message": "id-500",
  "path": "/users/500"
}
```

We can customize this default spring boot structure -

GET http://localhost:8080/users/1000 - Get request to a non existing resource. - The response shows a Customized Message Structure

```
{
  "timestamp": "2017-07-19T05:31:01.961+0000",
  "message": "id-500",
  "details": "Any details you would want to add"
}
```

We can do this using defining ExceptionResponse -

```
public class ExceptionResponse {
  private Date timestamp;
  private String message;
  private String details;
```

Secondly we will have to implement custome response handling by extending class ResponseEntityExceptionHandler. ResponseEntityExceptionHandler has default structure implemented, so we will override it -

```
@ControllerAdvice                    //because it applies to all the controllers
@RestController
public class CustomizedResponseEntityExceptionHandler extends
ResponseEntityExceptionHandler {

            .........

  @ExceptionHandler(Exception.class)
  public final ResponseEntity<Object> handleAllExceptions(Exception ex, WebRequest
  request) {

    ExceptionResponse exceptionResponse = new ExceptionResponse(new Date(),
    ex.getMessage(), request.getDescription(false));            //our own response
    object
    return new ResponseEntity(exceptionResponse, HttpStatus.INTERNAL_SERVER_ERROR);
  }

.............

  @ExceptionHandler(UserNotFoundException.class)
  public final ResponseEntity<Object>
  handleUserNotFoundException(UserNotFoundException ex, WebRequest request) {

    ExceptionResponse exceptionResponse = new ExceptionResponse(new Date(),
    ex.getMessage(), request.getDescription(false));

    return new ResponseEntity(exceptionResponse, HttpStatus.NOT_FOUND);
  }

 ..............
```

POST http://localhost:8080/users with Validation Errors

```
{
    "name": "R",                                        //More than two chars
    "birthDate": "2000-07-19T04:29:24.054+0000"         //should be in past
}
```

We can add above validation using java validation api -

```
@Entity
public class User {

  @Id
  @GeneratedValue
```

```
  private Integer id;

  @Size(min=2, message="Name should have atleast 2 characters")
  @ApiModelProperty(notes="Name should have atleast 2 characters")
  private String name;

  @Past
  @ApiModelProperty(notes="Birth date should be in the past")
  private Date birthDate;
```

Once we have added validations on the bean we can then add the invocation of our validation onto our resource. When someone is calling a POST request we would want to do binding and then invoke these validations. We do it using @Valid annotation -

```
@PostMapping("/users")
public ResponseEntity<Object>
        createUser(@Valid @RequestBody User user) {
```

So whenever any one send a request, first the validation on User will get fired. It its not valid then it will throw an exception. Add the custom exception response -

```
@Override
protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException ex, HttpHeaders
headers, HttpStatus status, WebRequest request) {

  ExceptionResponse exceptionResponse = new ExceptionResponse(new Date(),
  "Validation Failed", ex.getBindingResult().toString());

  return new ResponseEntity(exceptionResponse, HttpStatus.BAD_REQUEST);
}

.............
```

Response - 400 Bad Request

```
{
    "timestamp": "2017-07-19T09:00:27.912+0000",
    "message": "Validation Failed",
    "details": "org.springframework.validation.BeanPropertyBindingResult:
    1 errors\nField error in object 'user' on field 'name': rejected value [R];
    codes [Size.user.name,Size.name,Size.java.lang.String,Size]; arguments
    [org.springframework.context.support.DefaultMessageSourceResolvable: codes
    [user.name,name]; arguments []; default message [name],2147483647,2];
    default message [Name should have atleast 2 characters]"
}
```

75. Why do we need Versioning for RESTful Web Services?
    What are the versioning options that are available?
    How do you implement Versioning for RESTful Web Services?

```
public class PersonV1 {
    private String name;

    public PersonV1() {
        super();
    }

    public PersonV1(String name) {
        super();
        this.name = name;
    }

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }
}
```

PersonV1 is the version 1 class and in next version it was updated to -

```
public class PersonV2 {
    private Name name;

    public PersonV2() {
        super();
    }

    public PersonV2(Name name) {
        super();
        this.name = name;
    }

    public Name getName() {
        return name;
    }

    public void setName(Name name) {
        this.name = name;
    }
}

public class Name {
    private String firstName;
    private String lastName;

    public Name() {
    }

    public Name(String firstName, String lastName) {
        super();
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

}
```

Who ever is using the first version of a service will not be able to use the second version of the service because the structure of response has changed.
The old consumers are still expecting name to be returned as String. We can solve this by giving two different versions to same api.

Versioning Options:

- URI Versioning

```
            http://localhost:8080/v1/person
            http://localhost:8080/v2/person

        - Request Param Versioning
            http://localhost:8080/person/param?version=1
            http://localhost:8080/person/param?version=2

        - Header Versioning
            http://localhost:8080/person/header
                headers=[X-API-VERSION=1]

                i.e in postman select Headers - put Key as X-API-VERSION and value as 1

            http://localhost:8080/person/header
                headers=[X-API-VERSION=2]

        - MIME Type or Accept Header Versioning
            http://localhost:8080/person/produces
                produces=[application/vnd.company.app-v1+json]          //We will
                expect client to send accept headers like this.

                    i.e in postman select Headers - select Key as Accept and pass value
                    as application/vnd.company.app-v1+json

            http://localhost:8080/person/produces
                produces=[application/vnd.company.app-v2+json]

            called MIME type because application/json is MIME

    @RestController
    public class PersonVersioningController {

        @GetMapping("v1/person")
        public PersonV1 personV1() {
            return new PersonV1("Bob Charlie");
        }

        @GetMapping("v2/person")
        public PersonV2 personV2() {
            return new PersonV2(new Name("Bob", "Charlie"));
        }

        @GetMapping(value = "/person/param", params = "version=1")
        public PersonV1 paramV1() {
            return new PersonV1("Bob Charlie");
        }

        @GetMapping(value = "/person/param", params = "version=2")
        public PersonV2 paramV2() {
            return new PersonV2(new Name("Bob", "Charlie"));
        }

        @GetMapping(value = "/person/header", headers = "X-API-VERSION=1")
        public PersonV1 headerV1() {
            return new PersonV1("Bob Charlie");
        }

        @GetMapping(value = "/person/header", headers = "X-API-VERSION=2")
        public PersonV2 headerV2() {
            return new PersonV2(new Name("Bob", "Charlie"));
        }

        @GetMapping(value = "/person/produces", produces =
        "application/vnd.company.app-v1+json")   //produces attribute indicates what
        kind of output this service is producing. Here we are appending something to it
        to differentiate for versions
        public PersonV1 producesV1() {
            return new PersonV1("Bob Charlie");
        }
```

```
        @GetMapping(value = "/person/produces", produces =
        "application/vnd.company.app-v2+json")
        public PersonV2 producesV2() {
            return new PersonV2(new Name("Bob", "Charlie"));
        }
    }


    Versioning
        Media type versioning (a.k.a "content negotiation" or "accept header")      -
        we cannot execute this type directly on browser, we will need client like postman
            GitHub
        (Custom) headers versioning             - we cannot execute this type directly
        on browser, we will need client like postman
            Microsoft
        URI Versioning
            Twitter
        Request Parameter versioning
            Amazon

    Versioning
        - Factors
            - URI Pollution
            - Misuse of HTTP Headers        - because http headers were not ment for
            versioning of apis.
            - Caching
            - Can we execute the request on the browser?
            - API Documentation
        - No Perfect Solution


76. Which is the client you use to test RESTful Web Services?
    How do you use Postman to execute RESTful Service Requests?
    How can you send Request Headers using Postman?

    check Diagram.
```