**SERVLET & SPRINGBOOT**

1. **What is servlet ?**

* Servlet is a java programming class that is used for dynamic response to the client
* It is a request/response programming model.
* The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing our own servlets.
* All servlets must implement the javax.servlet.Servlet interface, which defines servlet lifecycle methods. When implementing a generic service, we can extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet() and doPost(), for handling HTTP-specific services.

1. **What are the types of request?**

* There are two types of requests done by client they are
* Static and dynamic request
* Static requests are in page request which is accessible by webpages in html
* Dynamic requests are out page request dealing with servlets that what page have to display recommended.

1. **What are the servlets architecture**?

Request

Client

* Numbers of servlets
* XML files

Response

Web container

1. **What is clients?**

* Clients are the one who is giving request to the webpages.
* Servlets response may vary according to clients request the response may be in any formats.

**Get?**

* Limited amount of data can be sent because data is sent in header.
* Not Secured because data is exposed in URL bar.
* Can be bookmarked
* It is more efficient and used than Post

**Post?**

* Large amount of data can be sent because data is sent in body.
* Secured because data is not exposed in URL bar
* Cannot be bookmarked
* It is less efficient and used

**Request dispatcher?**

* RequestDispatcher interface is used to forward the request to another resource that can be HTML, JSP or another servlet in same application. We can also use this to include the content of another resource to the response.
* There are two methods defined in this interface:
* 1.void forward()
* 2.void include()

**Forward():**

* forward() sends the same request to another resource.
* forward() method works at server side.
* forward() method works within the server only.

**SendRedirect()**

* sendRedirect() method sends new request always because it uses the URL bar of the browser.
* sendRedirect() method works at client side.
* sendRedirect() method works within and outside the server.

**Life Cycle of Servlets?**

* Servlet is loaded
* Servlet is instantiated
* Servlet is initialized
* Service the request
* Servlet is destroyed

**Servlet config?**

* Servlet config object represent single servlet
* Its like local parameter associated with particular servlet
* It’s a name value pair defined inside the servlet section of web.xml file so it has servlet wide scope
* getServletConfig() method is used to get the config object
* for example shopping cart of a user is a specific to particular user so here we can use servlet config

**servletContext?**

* It represent whole web application running on particular JVM and common for all the servlet
* Its like global parameter associated with whole application
* ServletContext has application wide scope so define outside of servlet tag in web.xml file.
* getServletContext() method is  used to get the context object.
* To get the MIME type of a file or application session related information is stored using servlet context object.

### What is JDBC Driver?

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

**What are the steps to connect to a database in java?**

* Registering the driver class
* Creating connection
* Creating statement
* Executing queries
* Closing connection

### ****What is the role of JDBC DriverManager class?****

The DriverManager class manages the registered drivers. It can be used to register and unregister drivers. It provides factory method that returns the instance of Connection.

### ****What is Spring?****

Wikipedia defines the Spring framework as “an application framework and inversion of control container for the Java platform. The framework’s core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.” Spring is essentially a lightweight, integrated framework that can be used for developing enterprise applications in java.

**Name the different modules of the Spring framework.**

Some of the important Spring Framework modules are:

* Spring Context – for dependency injection.
* Spring AOP – for aspect oriented programming.
* Spring DAO – for database operations using DAO pattern
* Spring JDBC – for JDBC and DataSource support.
* Spring ORM – for ORM tools support such as Hibernate
* Spring Web Module – for creating web applications.
* Spring MVC – Model-View-Controller implementation for creating web applications, web services etc.

**List some of the important annotations in annotation-based Spring configuration.**

The important annotations are:

* @Required
* @Autowired
* @Qualifier
* @Resource
* @PostConstruct
* @PreDestroy

### ****Explain Bean in Spring and List the different Scopes of Spring bean.****

Beans are objects that form the backbone of a Spring application. They are managed by the Spring IoC container. In other words, a bean is an object that is instantiated, assembled, and managed by a Spring IoC container.

* **Singleton**: Only one instance of the bean will be created for each container. This is the default scope for the spring beans. While using this scope, make sure spring bean doesn’t have shared instance variables otherwise it might lead to data inconsistency issues because it’s not thread-safe.
* **Prototype**: A new instance will be created every time the bean is requested.
* **Request**: This is same as prototype scope, however it’s meant to be used for web applications. A new instance of the bean will be created for each HTTP request.
* **Session**: A new bean will be created for each HTTP session by the container.
* **Global-session**: This is used to create global session beans for Portlet applications.

**How we can set the spring bean scope? And what supported scopes does it have?**

In Spring framework, the scope of a bean determines the lifecycle and visibility of the bean instance within the container. You can set the scope of a Spring bean using the `@Scope` annotation or the XML configuration.

### ****Explain the role of DispatcherServlet and ContextLoaderListener.****

**DispatcherServlet** is basically the front controller in the Spring MVC application as it loads the spring bean configuration file and initializes all the beans that have been configured. If annotations are enabled, it also scans the packages to configure any bean annotated with @Component, @Controller, @Repository or @Service annotations.

**ContextLoaderListener,** on the other hand, is the listener to start up and shut down the WebApplicationContext in Spring root. Some of its important functions includes tying up the lifecycle of Application Context to the lifecycle of the ServletContext and automating the creation of ApplicationContext.

### What is the best way to inject dependency? Also, state the reason

The best way to inject dependencies in an application is through constructor injection. Constructor injection involves passing dependencies as parameters to a class’s constructor. Here are the reasons why constructor injection is considered the best approach:

1. Explicit dependencies: Constructor injection makes dependencies explicit, as they are clearly defined as constructor parameters. This improves code readability and makes it easier to understand the class’s dependencies.
2. Compile-time safety: With constructor injection, dependencies are resolved at compile-time rather than runtime. This allows for early detection of missing or incorrect dependencies, reducing the chances of runtime errors.
3. Testability: Constructor injection facilitates easy testing of classes by allowing for the injection of mock or stub dependencies during unit testing. By providing test doubles through the constructor, you can isolate the class under test and verify its behavior without relying on real dependencies.
4. Immutability: Constructor injection promotes immutability as dependencies can be declared as final or read-only. Immutable objects are generally easier to reason about and less prone to bugs related to state changes.

### ****What is autowiring in Spring? What are the autowiring modes?****

Autowiring enables the programmer to inject the bean automatically. We don’t need to write explicit injection logic. Let’s see the code to inject bean using dependency injection.

### ****How to handle exceptions in Spring MVC Framework?****

Spring MVC Framework provides the following ways to help us achieving robust exception handling.

#### **Controller Based:**

We can define exception handler methods in our controller classes. All we need is to annotate these methods with @ExceptionHandler annotation.

#### **Global Exception Handler:**

Exception Handling is a cross-cutting concern and Spring provides @ControllerAdvice annotation that we can use with any class to define our global exception handler.

#### **HandlerExceptionResolver implementation:**

For generic exceptions, most of the times we serve static pages. Spring Framework provides HandlerExceptionResolver interface that we can implement to create global exception handler. The reason behind this additional way to define global exception handler is that Spring framework also provides default implementation classes that we can define in our spring bean configuration file to get spring framework exception handling benefits.

### ****What are some of the important Spring annotations which you have used?****

Some of the Spring annotations that I have used in my project are:

**@Controller** – for controller classes in Spring MVC project.

**@RequestMapping** – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through Spring MVC RequestMapping Annotation Examples

**@ResponseBody** – for sending Object as response, usually for sending XML or JSON data as response.

**@PathVariable** – for mapping dynamic values from the URI to handler method arguments.

**@Autowired** – for autowiring dependencies in spring beans.

**@Qualifier** – with @Autowired annotation to avoid confusion when multiple instances of bean type is present.

**@Service** – for service classes.

**@Scope** – for configuring the scope of the spring bean.

**@Configuration, @ComponentScan and @Bean** – for java based configurations.

### ****How to integrate Spring and Hibernate Frameworks?****

We can use Spring ORM module to integrate Spring and Hibernate frameworks if you are using Hibernate 3+ where SessionFactory provides current session, then you should avoid using HibernateTemplate or HibernateDaoSupport classes and better to use DAO pattern with dependency injection for the integration.

Also, Spring ORM provides support for using Spring declarative transaction management, so you should utilize that rather than going for hibernate boiler-plate code for transaction management.