Q1.What is the Spring MVC framework?

Ans:

Spring MVC is a lightweight, open-source framework for developing web applications on the Java platform. It is based on the Model-View-Controller (MVC) architectural pattern, which separates the application's concerns into three logical components:

- The model encapsulates the application's data and business logic.
- The view is responsible for rendering the model data to the user.
- The controller handles the user's requests and delegates them to the appropriate model or view.

Q2.What are the benefits of Spring MVC framework over other MVC frameworks?

Ans:

- Dependency injection. Spring MVC uses dependency injection to decouple the model, view, and controller components. This makes the application more modular and easier to test.
- AOP. Spring MVC supports aspect-oriented programming (AOP) for implementing cross-cutting concerns, such as logging and security. This makes the application more maintainable and easier to extend.
- A rich set of annotations. Spring MVC provides a rich set of annotations for mapping requests to controllers and views. This makes it easier to develop and maintain the application.
- Performance. Spring MVC is a lightweight framework that does not add a lot of overhead to the application. This makes it a good choice for applications that require high performance.
- A flexible view resolution mechanism. Spring MVC supports a flexible view resolution mechanism that allows you to use a variety of view technologies, such as JSP, Velocity, and FreeMarker.

Q3.What is DispatcherServlet in Spring MVC? In other words, can you explain the Spring MVC architecture?

Ans:

- The DispatcherServlet is the central component of the Spring MVC framework. It is a servlet that receives all HTTP requests for the application and delegates them to the appropriate controllers.
- The DispatcherServlet also handles the view resolution process.

MVC Architecture:

The Spring MVC architecture is based on the Model-View-Controller (MVC) pattern. The MVC pattern divides an application into three logical components:

- The model encapsulates the application's data and business logic.
- The view is responsible for rendering the model data to the user.
- The controller handles the user's requests and delegates them to the appropriate model or view.

Note:- The DispatcherServlet acts as the front controller in the Spring MVC architecture. It receives all HTTP requests for the application and delegates them to the appropriate controllers. The controllers then interact with the model to retrieve the data.

Q4.What is a View Resolver pattern and explain its significance in Spring MVC?

Ans: A view resolver is a component in a web application framework that maps logical view names to actual views. In Spring MVC, the view resolver is responsible for determining which view to render for a given request.

Q5.What are the differences between @RequestParam and @PathVariable annotations?

Ans: The @RequestParam and @PathVariable annotations are both used to extract values from HTTP requests in Spring MVC.

- The @RequestParam annotation is used to extract query parameters from the request URL.
- The @PathVariable annotation is used to extract values from the path of the request URL.

Q6.What is the Model in Spring MVC?

Ans: The Model in Spring MVC is a container that holds the data that is to be rendered by the view. It is a Map<String, Object> that is passed to the view by the controller. The view can then access the data in the Model to render the response.

Q7.What is the role of @ModelAttribute annotation?

Ans: The @ModelAttribute annotation is a powerful tool that can help you to expose data to the view in a consistent way.

- Consistency. The @ModelAttribute annotation provides a consistent way to expose data to the view. This makes your applications more readable and maintainable.
- Flexibility. The @ModelAttribute annotation can be used to bind any type of object to a model attribute. This gives you a lot of flexibility in how you expose data to the view.
- Reusability. The @ModelAttribute annotation can be used to reuse data across multiple views. This can help you to reduce code duplication and improve the performance of your applications.

Q9.What does REST stand for? and what is RESTful web services?

Ans: REST stands for Representational State Transfer. It is an architectural style for designing web services. RESTful web services are web services that are designed according to the REST architectural style.

- Resources. Everything in a RESTful web service is a resource. A resource can be anything, such as a document, a user, or a product.
- URIs. Resources are identified by URIs (Uniform Resource Identifiers). URIs are unique identifiers that are used to access resources.
- Methods. RESTful web services use a limited set of HTTP methods to interact with resources. The HTTP methods are GET, POST, PUT, and DELETE.
- Representations. Resources are represented in a format that is suitable for the application. The most common representation format for RESTful web services is JSON (JavaScript Object Notation).

Q10.What is differences between RESTful web services and SOAP web services?

Ans:

SOAP : - It stand **for Simple Object Protocol**

It uses XML document

It is slow than REST because it xml pages are heavy.

REST:- It stand Representational State Transfer

I is flexible and use the JSON, XML and plain text document as well

It is faster than SOAP because it mainly uses the JSON document which is very light weighted.