**Spring MVC, Spring Boot and Rest Controllers**

Question 1

What is MVC?

Model-View-Controller (MVC) is a design pattern used in software engineering. In the context of Spring, MVC is a module of the Spring framework that deals with the MVC pattern. It combines all the advantages of the MVC pattern with the convenience of Spring.

Correct

Is a software design pattern commonly based off a separation of concerns and reuse where components become responsible for View, Directing the request and the business logic of an application. The components are;

* Model: The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.
* View: Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
* Controller: Accepts input and converts it to commands for the model or view.

Question 2

What is a Front Controller?

A Front Controller in Spring MVC is a design pattern where a single controller, the DispatcherServlet, handles all incoming HTTP requests and processes them. It acts as the main controller to route requests to their intended destination.

Correct

Provides a centralized request handling mechanism so that all requests will be handled by a single handler. Spring MVC has a single Controller, the DispatcherServlet, that provides a single point of entry into a Spring Web Application.

Question 3

What does Spring use HandlerMappings for?

Spring uses HandlerMappings to define a mapping between requests and handler objects. When a request comes in, the DispatcherServlet will hand it over to the handler mapping to let it inspect the request and come up with an appropriate HandlerExecutionChain.

Correct

The DispatcherServlet passing requests to the Controllers with the help of handler mappings. The HandlerMapping Interface has several implementations, such as the class SimpleUrlHandlerMapping. This class allows for direct and declarative mapping between Controller methods and Urls

Question 1

What is the goal of Spring Boot?

The goal of Spring Boot is to reduce overall development time and increase efficiency by having a default setup for unit and integration tests. It allows developers to get started quickly without losing time on preparing and configuring their Spring application.

Correct

The main goal of Spring Boot Framework is to reduce Development, Unit Test and Integration Test time and to ease the development of Production ready web applications very easily compared to the existing Spring Framework. Opinionated defaults will make it quick and easy to get a skeletal web application up and running.

Question 2

Where would you typically find a maven Spring Boot configuration file?

In a Maven Spring Boot project, the configuration file, typically pom.xml, is usually located in the root directory of the project.

Correct

src/main/resources or sub directories there of

Question 3

Where would I find CSS files in a maven Spring Boot project?

CSS files in a Maven Spring Boot project are typically located under src/main/resources/static or src/main/resources/public. These directories are served from the root (/) of the application.

Correct

src/main/resources/static or sub directories there of

Question 1

How does the DispatcherServlet locate Controllers?

The DispatcherServlet in Spring MVC is a front controller that routes incoming HTTP requests to appropriate handler methods of controllers. It locates controllers by using a HandlerMapping. The HandlerMapping inspects the request and determines the appropriate HandlerExecutionChain which contains the handler object (a controller) and any handler interceptors.

Correct

The @SpringBootApplicaiton annotation on the Spring boot Application class, encapsulates the annotations @EnableAutoConfiguration, @Configuraiton and @ComponentScan. The later will scan for @Controller/@RestController annotated classes in the Application class package and sub packages by default to create the WebApplicationContext

Question 2

What is the purpose of @RequestMapping?

The @RequestMapping annotation in Spring MVC is used to map web requests to specific handler methods of controllers. It can be applied at both the class level and method level. At the class level, it provides a basic request mapping, while at the method level, it can be used to handle more specific mappings.

Correct

This annotation ties a URL Mapping to a particular Controller/RestController and is used by the HandlerMappings of the application to delegate from the DispatcherServlet to a Controller method at runtime

Question 3

What is request url Concatenation?

Request URL concatenation in Spring MVC occurs when both the class and method are declared with the @RequestMapping annotation along with their respective URLs. In this case, the request URL used by the user to access a given page must be the concatenation of the class’s URL and the method’s URL.

Correct

A Controller/RestController can have a @RequestMapping annotation annotating the class itself. However, in addition, methods within the class can use the “path” attribute (or default attribute “value”) of the @GetMapping or @PostMapping to further append a value to the @RequestMapping url to effectively tie the method to a concatenated url value.

Question 1

What 2 types of file formats can use to inject properties into the Spring Environment Object?

Types of file formats to inject properties into the Spring Environment Object: Spring allows you to externalize your configuration so you can work with the same application code in different environments. You can use properties files and YAML files to inject properties into the Spring Environment Object.

Correct

The files application.properties and application.yaml use a key value “flat” format traditional properties format or a parent child relationship yaml format respectively. Either is parsed and placed into a map like structure in the Spring Environment Object

Question 2

What does ${…} mean in a properties file?

Meaning of …inapropertiesfile∗∗:Inapropertiesfile,‘{…}is used to denote a placeholder for a property value that will be replaced by Spring's PropertySourcesPlaceholderConfigurer[^3^][4][^4^][5]. For example, if you have a property namedmyPropertyin a properties file, you can inject its value into a field like this:@Value(“${myProperty}”) private String myField;

Correct

This is Expression Language which will replace the expression with a literal at parsing time of the file i.e.

spring.app=test

spring.app.title=${spring.app} application

The value of spring.app.title is “test application

Question 3

What is the preferred approach to externalizing configuration and why?

Preferred approach to externalizing configuration and why: The preferred approach to externalizing configuration in Spring is to use a combination of properties files, YAML files, environment variables, and command-line arguments. This approach is preferred because it allows you to work with the same application code in different environments. Property values can be injected directly into your beans using the @Value annotation, accessed through Spring’s Environment abstraction, or bound to structured objects through @ConfigurationProperties

Correct

Spring Java Configuration Classes (@ConfigurationProperties), because they are type safe as they are defined in Java and compiled, they can be validated by JSR2020 validators, the configuration class can be injected into a Spring managed bean as a normal javax injected dependency with no Spring dependency

Question 1

How can you obtain a RequestHeader in a Controller method?

You can obtain a RequestHeader in a Controller method in Spring MVC by using the @RequestHeader annotation. This annotation can be used to bind a request header to a method parameter. For example, if you want to get the value of the “Accept-Encoding” header, you can do something like this:

@GetMapping("/displayHeaderInfo.do")

public void displayHeaderInfo(@RequestHeader("Accept-Encoding") String encoding) {

// ...

}

Correct

You can inject the HttpServletRequest object into your method and extract it via getParameter(header name), or use the @RequestHeader(“header name”) to inject the value of the header directly into your method

Question 2

Why would you select using @RequestParam over @PathVariable for a Parameter?

You would select using @RequestParam over @PathVariable for a parameter when you want to extract values from the query string of the URL@RequestParam is typically used to extract query parameters or form data from an HTTP request. On the other hand, @PathVariable is used when you want to extract values from the URI path.

Correct

If a Request Parameter can be Optional, you should use @RequestParam. While @PathVaitable Parameters comprise data directly from the url and cannot be optional

Question 3

What do we mean by URL Templates?

By URL Templates, we mean a way of parameterizing URLs so that certain parts of them can be replaced with actual values at runtime. In Spring MVC, this is often done using @PathVariable. For example, in the URL template “/users/{userId}”, “{userId}” is a placeholder that can be replaced with an actual user ID.

Correct

URL’s with placeholders in them for data. These are used by @PathVariable annotated method arguments to extract the data from the url and inject it into the RestController method.

Question 1

When would you select the getForObject over the getForEntity method from RestTemplate?

The getForObject() and getForEntity() methods from RestTemplate in Spring both send an HTTP GET request and return the response body. However, getForObject() returns the response body directly converted into the object, while getForEntity() returns a ResponseEntity object, which includes additional information like HTTP status code and headers. Therefore, you would select getForObject() when you only care about the response body, and getForEntity() when you also care about the HTTP metadata.

Correct

If you are not interested in the entire ResponseEntity, including headers and status code, using getForObject will directly extract the payload out of the underlying ResponseEntity for you without you extracting it yourself via the method getBody() on the ReponseEntity

Question 2

What is Content Negotiation?

Content Negotiation is a mechanism that allows a client and server to agree on a data exchange format. The client specifies the desired data format (like JSON or XML) in an HTTP header (usually the “Accept” header), and the server uses this information to formulate its response in the requested format.

Correct

The client of a service can request what the textural response type it requires by sending an “Accept” header with the desired MediaType. If no Accept header is sent, the service will decide on the textural response format. In addition, posting data to a service, the service can identify endpoints that only consume a specific textual format by consuming a request with a specific content-type header

Question 3

What do we mean by URL Templates?

URL Templates are a way of parameterizing URLs so that certain parts of them can be replaced with actual values at runtime. In Spring, this is often done using @PathVariable. For example, in the URL template “/users/{userId}”, “{userId}” is a placeholder that can be replaced with an actual user ID.

Correct

URL’s with placeholders in them for data. These are used by @PathVariable annotated method arguments to extract the data from the url and inject it into the RestController method.

Question 1

What is the return type of a Controller Method interpreted as?

The Controller Method return value is used to interpret the view. This can take various forms, such as a `String` (indicating the view) or a `ModelAndView` object (containing the view name and model data).

Correct

If the class is annotated as a @Controller, the return type of a method tied to a url is interpreted as a destination. Return types of “String” are passed to a ViewReolver, return types of ModelAndView have the String View value extracted and passed to a ViewResovler. A ModelAndView class is a means of convenience to return a Model attribute and a View using its overloaded constructor

Question 2

What is the Model Object?

A Model Object is an object or collection of objects that provides the data that a view uses to render. It is created before the controller method starts processing and is automatically called by the `@ModelAttribute` method.

Correct

Is and object that acts as a holder for model attributes. The DispatcherServlet create a Model before invoking a Controller method. This makes it available to the Method for Injection and into which attributes are held. Upon a Response being received by the DispatcherServlet, attributes in the Model are taken and injected as attributes into the Request. The request is dispatched, forwarded, to a ViewResolver that identifies the View which will receive the request and its attributes.

Question 3

Where does the Thymeleaf ViewResolver look for html files?

Thymeleaf ViewResolver typically looks for html files in the `src/main/resources/templates` directory of the project. However, this can be changed through settings.

Correct

src/main/resource/templates