Spring MVC

**Spring MVC(Model-View-Controller) provides a convenient way to develop a java based web application.**

**It has a central servlet called as DispatcherServlet which is well-known as a front controller that intercepts all the requests, identifies the appropriate handler i.e. controllers, and renders views to the client.**

**It is defined at org. spring frameworkk.web.servlet.DispatcherServlet in org. spring framework.web package.**

# Spring MVC flow

**In Spring Web MVC, DispatcherServlet class works as the front controller. It is responsible to manage the flow of the spring mvc application.**

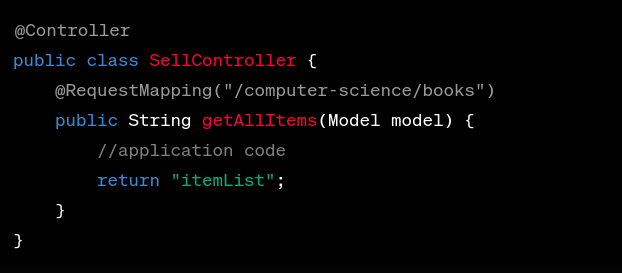
**The @Controller annotation is used to mark the class as the controller in Spring 3.**

**The @RequestMapping annotation is used to map the request url. It is applied on the method.**

Spring Web Annotations

## @RequestMapping

**The @RequestMapping annotation is used to map the web requests. It can hold several optional components such as consumes, header, method, name, params, path, value, etc. This annotation can be used with classes as well as methods. Consider the below example:**



@RequestBody

**The @RequestBody annotation is used to bind HTTP requests with an object in a method parameter. Internally it uses HTTP MessageConverters to convert the request's body. When a method parameter is annotated with @RequestBody annotation, the Spring framework wraps the incoming HTTP request body to that parameter.**

|  |
| --- |
| @PostMapping("/save")  void saveVehicle(@RequestBody Vehicle vehicle) {  *//*  } |

@PathVariable

**This annotation indicates that a method argument is bound to a URI template variable. We can specify the URI template with the @RequestMapping annotation and bind a method argument to one of the template parts with @PathVariable.**

**We can achieve this with the name or its alias, the value argument:**

|  |
| --- |
| @RequestMapping("/{id}") Vehicle getVehicle(@PathVariable("id") long id) {  *// ...* } |

**If the name of the part in the template matches the name of the method argument, we don't have to specify it in the annotation:**

|  |
| --- |
| @RequestMapping("/{id}") Vehicle getVehicle(@PathVariable long id) {  *// ...* } |

**Moreover, we can mark a path variable optional by setting the argument required to false:**

|  |
| --- |
| @RequestMapping("/{id}") Vehicle getVehicle(@PathVariable(required = false) long id) {  *// ...* } |

@RequestParam

**The @RequestParam annotation aka query parameter is used to extract the query parameters from the URL. It is most suitable for developing web applications. It can define the default values if the query parameter is not present in the URL.**

@ResponseBody

**If we mark a request handler method with @ResponseBody, Spring treats the result of the method as the response itself:**

|  |
| --- |
| @ResponseBody @RequestMapping("/hello") String hello() {  return "Hello World!"; } |

@Controller

**We can define a Spring MVC controller with @Controller.@Controller is a class level annotation which tells the Spring Framework that this class serves as a controller in Spring MVC:**

|  |
| --- |
| @Controller public class VehicleController {  *// ...* } |

@RestController

**@RestController is a specialized version of the controller. It includes the @Controller and @ResponseBody annotations, and as a result, simplifies the controller implementation:**

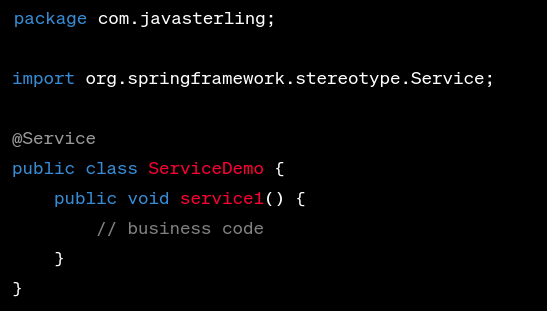
|  |
| --- |
| @Controller @ResponseBody class VehicleRestController {  *// ...* } |

**is same as :**

|  |
| --- |
| @RestController class VehicleRestController {  *// ...* } |

@Service

**The @Service annotation is also used at the class level to mark a service implementation including business logic, calculations, call external APIs, etc. Generally, it holds the business logic of the application. Consider the below example:**



### @GetMapping

**The @GetMapping annotation is used to map the HTTP GET request on the specific handler method. It is an alternative to @RequestMapping(method = RequestMethod.GET).**

|  |
| --- |
| @GetMapping(value = "/users") public Users getUsers() { ... }  @GetMapping(value = "/users/{id}") public User getUser(@PathVariable("id") String id) { ... }  @PostMapping(value = "/users") public User createUser(User user) { ... } |

### @PostMapping

**The @PostMapping annotation is used for mapping HTTP POST requests onto specific handler methods. It is an alternative to @RequestMapping(method = RequestMethod.POST).**

|  |
| --- |
| @PostMapping(value = "/members")  *//or*  @PostMapping("/members") |

### @PutMapping

**The @PutMapping annotation is used to map the HTTP PUT requests on the specific handler method. It is useful for creating a web service endpoint that creates or updates. It is an alternative to @RequestMapping(method = RequestMethod.PUT).**

|  |
| --- |
| @PutMapping("/api/myresource/{id}")  public String updateResource(@PathVariable("id") String resourceId, @RequestBody String updatedResource) {  *// Your logic to update the resource with the given resourceId using the updatedResource data*  *// Return any response, such as success message or updated resource object*  }   *// Other methods and endpoints in your controller* } |

### 

### @DeleteMapping

**The @DeleteMapping is used to map the HTTP DELETE requests on the specific handler method. It is useful for creating a web service endpoint that deletes a resource. It is an alternative to @RequestMapping(method = RequestMethod.DELETE).**

|  |
| --- |
| @DeleteMapping("/api/myresource/{id}")  public String deleteResource(@PathVariable("id") String resourceId) {  *// Your logic to delete the resource with the given resourceId*  *// Return any response, such as success message or deleted resource object*  }   *// Other methods and endpoints in your controller* } |

# JSON

**JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for both humans and machines to read and write. It is commonly used for transmitting data between a server and a web application as an alternative to XML.**

**Here's a simple example of a JSON object:**

|  |
| --- |
| {  "name": "John",  "age": 30,  "city": "New York" } |

**In the above example, we have a JSON object with three key-value pairs. The key is always a string and the value can be a string, number, boolean, array, or another JSON object.**