**Аннотации MVC**

**1. @SpringBootApplication** – Аннотация @SpringBootApplication включает сканирование компонентов и автоматическую конфигурацию приложения. На самом деле, за этой аннотацией скрывается целых три аннотации с настройками по умолчанию:

**@Configuration** (Spring) – помечает класс как конфигурационный (в контексте java-based конфигурации).

**@ComponentScan** (Spring) – включает сканирование компонентов, то есть классы-контроллеры и другие компоненты, которые вы создаете, будут автоматически зарегистрированы как бины в application context.

**@EnableAutoConfiguration** (Spring Boot) – самая магическая аннотация из всех возможных. Именно она под капотом запускает весь механизм автоматической конфигурации Spring Boot. Это означает, что Spring Boot ищет bean-компоненты автоконфигурации в своем пути к классам и автоматически применяет их.

**2. @RequestMapping** – используется для отображения запросов на методы контроллеров. Она имеет различные атрибуты для сопоставления по URL-адресу, HTTP-методу, параметрам запроса, заголовкам и типам данных. Вы можете использовать её на уровне класса для выражения общих отображений или на уровне метода для сужения до конкретного отображения конечной точки.

Проще говоря, аннотация используется для сопоставления веб-запросов с методами Spring Controller.

Существуют также специфические для HTTP-метода варианты сокращения аннотации:

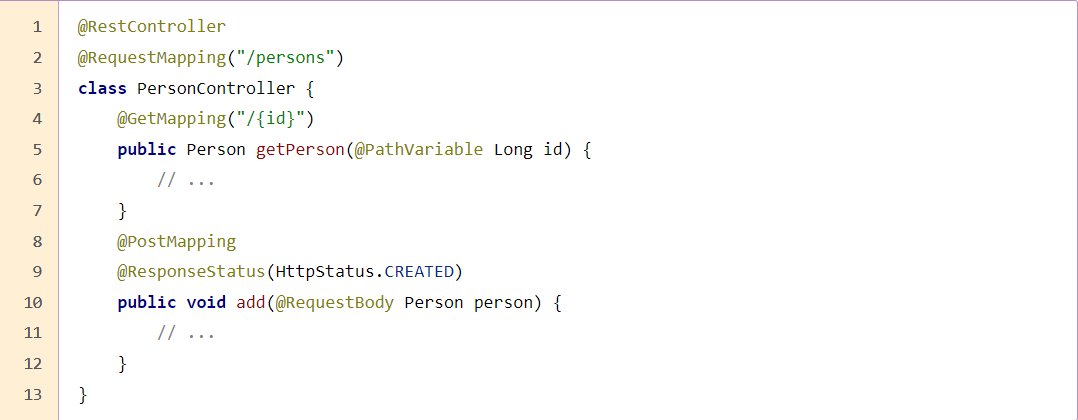
@RequestMapping:

@GetMapping

@PostMapping  
@PutMapping  
@DeleteMapping  
@PatchMapping

Сокращения являются кастомными аннотациями, которые указываются потому, что, пожалуй, большинство методов контроллера нужно сопоставить с конкретным HTTP-методом, а не использовать аннотацию @RequestMapping, которая по умолчанию производит сопоставление со всеми HTTP-методами. Для выражения общих отображений на уровне класса по-прежнему необходима @RequestMapping.

Пример:



**3. @Controller** - указывает, что конкретный класс является контроллером.

DispatcherServlet отправляет запрос контроллерам для выполнения определённых функций.

Аннотация @RequestMapping используется для мапинга (связывания) с URL для всего класса или для конкретного метода обработчика. Обычно мы используем @Controllerв сочетании с аннотацией @RequestMapping для методов обработки запросов.

*@Controller* это такой же *@Component (бин)* только который умеет обрабатывать запросы и умеет работать с url.

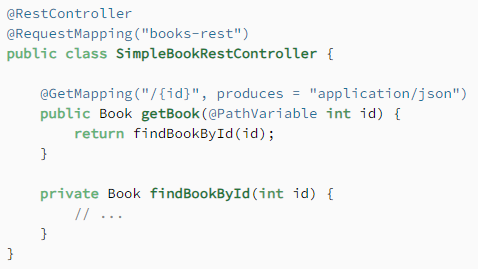
 Пример:

Мы аннотировали метод обработки запроса @ResponseBody. Эта аннотация включает автоматическую сериализацию возвращаемого объекта в HttpResponse.

**4. @RestController -** В Spring 4.0 появилась аннотация *@RestController*, чтобы упростить создание веб-сервисов RESTful. **Это удобная аннотация, объединяющая** @Controller **и** @ResponseBody, которая устраняет необходимость аннотировать каждый метод обработки запроса класса контроллера аннотацией @ResponseBody.

@RestController - специализированная версия контроллера. Он включает аннотации @Controller и @ResponseBody и, как следствие, упрощает реализацию контроллера.

Пример:



**Контроллер помечен аннотацией** @RestController**; поэтому** @ResponseBody **не требуется.**

Каждый метод обработки запросов класса контроллера автоматически сериализует возвращаемые объекты в HttpResponse.

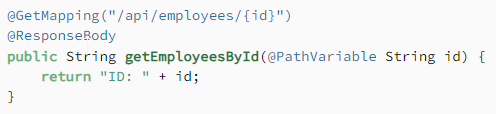
**5. @ResponseBody –** эта аннотация сообщает контроллеру, что возвращаемый объект автоматически сериализуется в JSON и передается обратно в объект HttpResponse (означает, что ответом будет являться объект).

*@ResponseBody* также поддерживается на уровне класса, и в этом случае он наследуется всеми методами контроллера. Чтобы не писать над каждым методом аннотация *@ResponseBody* можно использовать аннотацию класса *@RestController* которая включает в себя *@Controller* и *@ResponseBody.*

**6. @PathVariable** - используется для связывания параметров из URL с параметрами метода. В приведенном выше примере, число123 может быть извлечено с помощью @PathVariable.

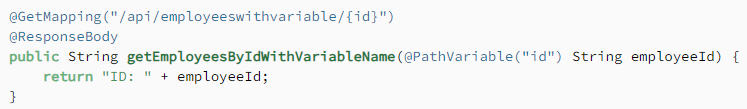
Например, http://localhost:8080/api/employees/111 .

Число 111 в этом случае является идентификатором продукта (id).

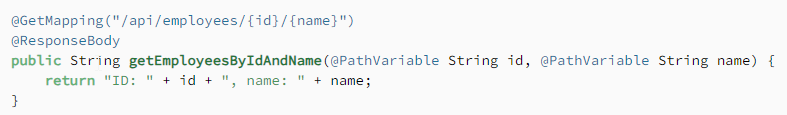


В этом примере, {id} в URL сопоставляется с параметром id метода getEmployeesById.

Если имя переменной пути отличается, мы можем указать его в аргументе аннотации@PathVariable:



В зависимости от варианта использования мы можем иметь более одной переменной пути в нашем URI запроса для метода контроллера, который также имеет несколько параметров метода:



**7. @RequestParam** - используется для чтения значений из параметров запроса. Это обычно используется, когда параметры передаются в URL после символа ?.

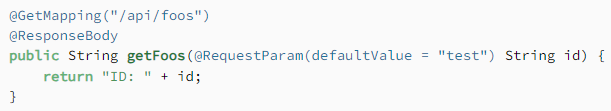
Например, http://localhost:8080/spring-mvc-basics/api/foos?id=abc .



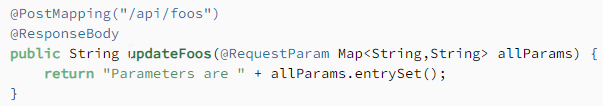
В этом случае, id=abc извлекается из URL и связывается с параметром метода id.

Параметры метода, аннотированные @RequestParam, являются обязательными по умолчанию.  
Это означает, что если параметр отсутствует в запросе, мы получим ошибку.  
Однако мы можем настроить @RequestParam как необязательный с атрибутом required:

Мы также можем установить значение по умолчанию для @RequestParam, используя атрибут defaultValue:



Мы также можем иметь несколько параметров без определения их имен или количества, просто используя Map:



**8. @ModelAttribute** - это аннотация, которая связывает параметр метода или возвращаемое значение метода с именованным атрибутом модели, а затем предоставляет его веб-представлению. Мы можем использовать *@ModelAttribute* либо в качестве параметра метода, либо на уровне метода.

На Уровне метода:

Когда мы используем аннотацию на уровне метода, это указывает на то, что целью метода является добавление одного или нескольких атрибутов модели. Такие методы поддерживают те же типы аргументов, что и методы [*@RequestMapping*](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/bind/annotation/RequestMapping.html) , но их нельзя напрямую сопоставлять с запросами.

### Как аргумент метода:

### Когда мы используем аннотацию в качестве аргумента метода, она указывает на получение аргумента из модели. Если аннотация отсутствует, ее следует сначала создать, а затем добавить в модель. После присутствия в модели поля аргументов должны заполняться всеми параметрами запроса, имеющими совпадающие имена.