مطمئناً! در اینجا توضیحات کامل‌تری درباره انواع مختلف `@RequestMapping` و آنوتیشن‌های مرتبط در Spring Boot به همراه مثال‌ها و روش‌های تست آن‌ها ارائه می‌دهم.

۱. `@GetMapping`

این آنوتیشن برای مدیریت درخواست‌های HTTP GET استفاده می‌شود. معمولاً برای دریافت اطلاعات از سرور به کار می‌رود.

مثال:

```java
@RestController
@RequestMapping("/api")
public class MyController {

    @GetMapping("/hello")
    public String sayHello() {
        return "Hello, World!";
    }
}
```

تست: برای تست این متد می‌توانید از ابزارهایی مانند Postman استفاده کنید. یک درخواست GET به URL زیر ارسال کنید:

```awk
GET http://localhost:/8080api/hello
```

پاسخ مورد انتظار:

```autohotkey
Hello, World!
```

۲. `@PostMapping`

این آنوتیشن برای مدیریت درخواست‌های HTTP POST استفاده می‌شود. معمولاً برای ارسال داده به سرور و ایجاد منابع جدید به کار می‌رود.

مثال:

```java
@RestController
@RequestMapping("/api")
public class ItemController {

    @PostMapping("/create")
    public ResponseEntity<String> createItem@)RequestBody Item item) {
        // فرض کنید که آیتم در دیتابیس ذخیره می‌شود
        //itemService.save(item);
        return ResponseEntity.status(HttpStatus.CREATED).body("Item created");
    }
```

```
}
```

تست: در Postman یک درخواست POST به URL زیر ارسال کنید و در بخش Body داده‌ها را به فرمت JSON ارسال کنید:

```bash
POST http://localhost:8080/api/create
Content-Type: application/json

{
    "name": "Item1",
    "price": 100
}
```

پاسخ مورد انتظار:

```ebnf
Item created
```

۳. `@PutMapping`

این آنوتیشن برای مدیریت درخواست‌های HTTP PUT استفاده می‌شود. معمولاً برای به‌روزرسانی منابع موجود به کار می‌رود.

مثال:

```java
@RestController
@RequestMapping("/api")
public class ItemController {

    @PutMapping("/update/{id}")
    public ResponseEntity<String> updateItem@)PathVariable Long id,@ RequestBody Item item) {
        //ID فرض کنید که آیتم با مشخص شده به‌روز می‌شود
        //itemService.update(id, item);
        return ResponseEntity.ok("Item updated");
    }
}
```

تست: در Postman یک درخواست PUT به URL زیر ارسال کنید:

```bash
PUT http://localhost:8080/api/update/1
Content-Type: application/json

{
    "name": "Updated Item",
    "price": 150
}
```

پاسخ مورد انتظار:

```ebnf
Item updated
```

۴. `@DeleteMapping`

این آنوتیشن برای مدیریت درخواست‌های HTTP DELETE استفاده می‌شود. معمولاً برای حذف منابع به کار می‌رود.

مثال:

```java
@RestController
@RequestMapping("/api")
public class ItemController {

    @DeleteMapping("/delete/{id}")
    public ResponseEntity<String> deleteItem@)PathVariable Long id) {
        //ID فرض کنید که آیتم با مشخص شده حذف می‌شود
        //itemService.delete(id);
        return ResponseEntity.ok("Item deleted");
    }
}
```

تست: در Postman یک درخواست DELETE به URL زیر ارسال کنید:

```awk
DELETE http://localhost:/8080api/delete/1
```

پاسخ مورد انتظار:

```ebnf
Item deleted
```

۵. `@RequestMapping`

این آنوتیشن عمومی‌ترین روش برای مدیریت درخواست‌هاست و می‌تواند برای انواع مختلف درخواست‌ها استفاده شود. با استفاده از این آنوتیشن می‌توانید نوع درخواست را با استفاده از `method` مشخص کنید.

مثال:

```java
@RestController
@RequestMapping("/api")
public class ItemController {

    @RequestMapping(value = "/items", method = RequestMethod.GET)
    public List<Item> getAllItems() {
        // فرض کنید که لیست آیتم‌ها بازگردانده می‌شود
        //return itemService.findAll();
        return Arrays.asList(new Item("Item1", 100), new Item("Item2", 200));
    }
```

```
}
```

تست: در Postman یک درخواست GET به URL زیر ارسال کنید:

```awk
GET http://localhost:/8080api/items
```

پاسخ مورد انتظار:

```json
]
    }
        "name": "Item1",
        "price": 100
    ,{
    }
        "name": "Item2",
        "price": 200
    {
[
```

۶. `@PatchMapping`

این آنوتیشن برای مدیریت درخواست‌های PATCH استفاده می‌شود که معمولاً برای به‌روزرسانی جزئی منابع به کار می‌رود.

مثال:

```java
@RestController
@RequestMapping("/api")
public class ItemController {

    @PatchMapping("/update/{id}")
    public ResponseEntity<String> partiallyUpdateItem@)PathVariable Long id,
@RequestBody Map>String, Object <updates) {
        // فرض کنید که فقط فیلدهای مشخص شده به‌روز می‌شوند
        //itemService.updatePartially(id, updates);
        return ResponseEntity.ok("Item partially updated");
    }
}
```

تست: در Postman یک درخواست PATCH به URL زیر ارسال کنید:

```bash
PATCH http://localhost:8080/api/update/1
Content-Type: application/json

{
    "price": 120
}
```

پاسخ مورد انتظار:

```ebnf
Item partially updated
```

نتیجه‌گیری

این آنوتیشن‌ها به شما اجازه می‌دهند تا به راحتی درخواست‌های مختلف HTTP را در Spring Boot مدیریت کنید. برای تست هر یک از این متدها، می‌توانید از Postman یا ابزارهای مشابه استفاده کنید تا درخواست‌ها را به سرور ارسال کنید و پاسخ‌ها را مشاهده نمایید. این روش‌ها به شما کمک می‌کنند تا APIهای RESTful را به سادگی پیاده‌سازی کنید.