FROM SPRINGBOOT TO MICRONAUT

URL VARIABLE - SPRINGBOOT

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RequestMapping("/api")
@RestController
public class GreetingController {

    @GetMapping("/greeting")
    public Greeting greeting(@RequestParam(value="name") String name) {
...
```

URL VARIABLE - SPRINGBOOT

```
GET http://localhost:8080/api/greeting

400
{
"timestamp":"2018-08-27T17:27:11.028+0000",
"status":400,
"error":"Bad Request",
"message":"Required String parameter 'name' is not present",
"path":"/api/greeting"
}
```

URL VARIABLE - SPRINGBOOT

```
GET http://localhost:8080/api/greeting?name=sergio
```

```
200
{"id":3,"content":"Hello, sergio!"}
```

FUNCTIONAL TEST - SPRINGBOOT

```java
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.web.reactive.server.WebTestClient;

@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class GreetingControllerTest {

    @Autowired
    private WebTestClient webClient;

    @Test
    public void invokeApiGreetingWithNameParam() {
        this.webClient.get().uri("/api/greeting?name=sergio").exchange().expectStatus().isOk()
                .expectBody(Greeting.class);
    }
```

URL VARIABLE - MICRONAUT

RFC-6570 URI TEMPLATE

```
import io.micronaut.http.annotation.Controller;
import io.micronaut.http.annotation.Get;

@Controller("/api")
public class GreetingController {

    @Get("/greeting{?name}")
    public Greeting greeting(String name) {
...
```

URL VARIABLE - MICRONAUT

```
GET http://localhost:8080/api/greeting

400
{
  "_links": {
    "self": {
      "href": "/api/greeting",
      "templated": false
    }
  },
  "message": "Required argument [String name] not specified",
  "path": "/name"
}
```

URL VARIABLE - MICRONAUT

```
GET http://localhost:8080/api/greeting?name=sergio

200
{
"id":3,
"content":"Hello, sergio!"
}
```

FUNCTIONAL TEST - MICRONAUT

```
import io.micronaut.runtime.server.EmbeddedServer;

public class GreetingControllerTest {
    private static EmbeddedServer server;
    private static HttpClient client;

    @BeforeClass
    public static void setupServer() {
        server = ApplicationContext.run(EmbeddedServer.class);
        client = server.getApplicationContext().createBean(HttpClient.class, server.getURL());
    }

    @Test
    public void invokeApiGreetingWithNameParam() {
        HttpRequest request = HttpRequest.GET("/api/greeting?name=sergio");
        Greeting greeting = client.toBlocking().retrieve(request, Greeting.class);
        assertNotNull(greeting);
    }
```

OPTIONAL URL VARIABLE - SPRINGBOOT

```
@GetMapping("/greeting")
public Greeting greeting(RequestParam(required = false) String name) {
```

OPTIONAL URL VARIABLE - MICRONAUT

```
@Get("/greeting{?name}")
public Greeting greeting(@Nullable String name) {
```

PATH VARIABLE - SPRINGBOOT

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.validation.Valid;

@RequestMapping("/api")
@RestController
public class BookController {

    private static final String ISBN_REGEX = "^1491950358|1680502395$";

    @GetMapping("/books/{isbn:"+ISBN_REGEX+"}")
    public Book show(@PathVariable String isbn) {
    ...
```

URL VARIABLE - SPRINGBOOT

```
GET http://localhost:8080/api/books/1491950358
```

```
200
{"isbn":"1491950358","name":"Building Microservices"}
```

PATH VARIABLE - SPRINGBOOT

```
GET http://localhost:8080/api/books/1491950555
```

```
404
{
"timestamp":"2018-08-27T17:47:00.883+0000",
"status":404,
"error":"Not Found",
"message":"No message available",
"path":"/api/books/1491950555"
}
```

ERROR CODE TESTS - SPRINGBOOT

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class BookControllerTest {

    @Autowired
    private WebTestClient webClient;

    @Test
    public void invokeApiBooksWithInvalidIsbn() {
        this.webClient.get().uri("/api/books/12356789").exchange().expectStatus().isNotFound();
    }
```

PATH VARIABLE - MICRONAUT

```
import io.micronaut.http.annotation.Controller;
import io.micronaut.http.annotation.Get;

@Controller("/api")
public class BookController {
    static final String ISBN_REGEX = "^1491950358|1680502395$";

    @Get("/books/{isbn:"+ISBN_REGEX+"}")
    public Book show(String isbn) {
    ...
```

PATH VARIABLE - MICRONAUT

```
GET http://localhost:8080/api/books/1491950358
```

```
200
{"isbn":"1491950358","name":"Building Microservices"}
```

PATH VARIABLE - MICRONAUT

```
GET http://localhost:8080/api/books/1491950555

404
{
  "_links": {
    "self": {
      "href": "/api/books/1491950555",
      "templated": false
    }
  },
  "message": "Page Not Found"
}
```

ERROR CODE TESTS - MICRONAUT

```
@Rule
public ExpectedException thrown = ExpectedException.none();

@Test
public void invokeApiBookWithoutParam() {
    thrown.expect(HttpClientResponseException.class);
    thrown.expect(hasProperty("response", hasProperty("status", is(HttpStatus.NOT_FOUND))));
    client.toBlocking().exchange(HttpRequest.GET("/api/books/12356789"), Boolean.class);
}
```

RESPONSE STATUS - SPRINGBOOT

```
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.PostMapping;

@RestController
@RequestMapping("/api")
public class RegisterController {

    @PostMapping("/register")
    @ResponseStatus(HttpStatus.CREATED)
    public void registerAccount() {
    }
```

RESPONSE STATUS TEST - SPRINGBOOT

```java
@Test
public void invokeApiRegister() {
    this.webClient.post().uri("/api/register").exchange().expectStatus().isCreated();
}
```

RESPONSE STATUS - MICRONAUT

```
import io.micronaut.http.HttpResponse;
import io.micronaut.http.HttpStatus;
import io.micronaut.http.annotation.Controller;
import io.micronaut.http.annotation.Post;

@Controller("/api")
public class RegisterController {

    @Post("/register")
    public HttpResponse registerAccount() {
        return HttpResponse.status(HttpStatus.CREATED);
    }
}
```

RESPONSE STATUS TEST - MICRONAUT

```
@Test
public void invokeRegister() {
    HttpRequest request = HttpRequest.POST("/api/register", "");
    HttpResponse rsp = client.toBlocking().exchange(request);
    assertEquals(rsp.getStatus(), HttpStatus.CREATED);
}
```

RESPONSE STATUS TEST - SPRINGBOOT

```java
@Test
public void invokeApiRegister() {
    this.webClient.post().uri("/api/register").exchange().expectStatus().isCreated();
}
```

HTTP RESPONSE & HEADERS - SPRINGBOOT

```java
@GetMapping("/books")
public ResponseEntity<List<Book>> list() {
    Book b = new Book("1491950358", "Building Microservices");
    List<Book> books = Collections.singletonList(b);
    return ResponseEntity.ok()
            .header(HttpHeaders.WARNING, "This maybe expired")
            .body(books);
}
```

HTTP RESPONSE & HEADERS - SPRINGBOOT

```
@Test
public void invokeBooks() {
    this.webClient.get().uri("/api/books").exchange().expectStatus().isOk()
            .expectHeader().exists(HttpHeaders.WARNING);
}
```

HTTP RESPONSE & HEADERS - MICRONAUT

```
@Get("/books")
public HttpResponse<List<Book>> list() {
    Book b = new Book("1491950358", "Building Microservices");
    List<Book> books = Collections.singletonList(b);
    return HttpResponse.ok(books)
        .header(HttpHeaders.WARNING, "This maybe expired");
}
```

HTTP RESPONSE & HEADERS - MICRONAUT

```
@Test
  public void invokeApiBooks() {
      HttpRequest request = HttpRequest.GET("/api/books");
      Argument arg = Argument.of(List.class, Book.class);
      HttpResponse<List<Book>> response = client.toBlocking().exchange(request, arg);
      assertEquals(HttpStatus.OK, response.status());
      assertTrue(response.getHeaders().contains(HttpHeaders.WARNING));
  }
```

BODY & VALIDATION - SPRING BOOT

```
import javax.validation.Valid;
import org.springframework.web.bind.annotation.RequestBody;

@RequestMapping("/api")
@RestController
public class BookController {

@PostMapping("/books")
@ResponseStatus(HttpStatus.CREATED)
public void save(@RequestBody @Valid Book book)
```

VALIDATION TEST - SPRING BOOT

```
@Test
public void invokeBooksSaveWithInvalidPOJO() {
    this.webClient.post().uri("/api/books")
        .body(BodyInserters.fromObject(new Book("", "new Book")))
        .exchange()
        .expectStatus().isBadRequest();
}
```

VALIDATION EXCEPTION - SPRING BOOT

```
curl -X "POST" "http://localhost:8081/api/books" \
     -H 'Content-Type: application/json; charset=utf-8' \
     -d $'{
  "name": "new Book"
}'
```

VALIDATION EXCEPTION - SPRING BOOT

```
{
  "timestamp": "2018-08-28T06:58:03.146+0000",
  "status": 400,
  "error": "Bad Request",
  "errors": [
    {
      "codes": [
        "NotBlank.book.isbn",
        "NotBlank.isbn",
        "NotBlank.java.lang.String",
        "NotBlank"
      ],
     ...
}
```

BODY & VALIDATION - MICRONAUT

```
import io.micronaut.validation.Validated;
import javax.validation.Valid;

@Validated
@Controller("/api")
public class BookController {

@Post("/books")
public HttpResponse save(@Body @Valid Book book) {
```

VALIDATION TEST - MICRONAUT

```java
@Test
public void invokeApiBookSaveWithoutPayload() {
    thrown.expect(HttpClientResponseException.class);
    thrown.expect(hasProperty("response",
                  hasProperty("status", is(HttpStatus.BAD_REQUEST))));
    HttpRequest req = HttpRequest.POST("/api/books", new Book("", ""));
    client.toBlocking().exchange(req, Book.class);
}
```

VALIDATION EXCEPTION - MICRONAUT

```
curl -X "POST" "http://localhost:8081/api/books" \
     -H 'Content-Type: application/json; charset=utf-8' \
     -d $'{
  "name": "new Book"
}'
```

VALIDATION EXCEPTION - MICRONAUT

```
Http Status: 400
{
  "_links": {
    "self": {
      "href": "/api/books",
      "templated": false
    }
  },
  "_embedded": {
    "_embedded": [
      {
        "message": "book.isbn.:must not be blank"
      },
      {
        "message": "book.isbn.:must not be null"
      }
    ]
  },
  "message": "Bad Request"
}
```

HANDLE EXCEPTION - SPRINGBOOT

```
public class BookNotFoundException extends RuntimeException {}
```

```
@GetMapping("/books/ex")
@ResponseStatus(HttpStatus.OK)
public void notFound() {
    throw new BookNotFoundException();
}
```

HANDLE EXCEPTION - SPRINGBOOT

```java
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.NativeWebRequest;

@ControllerAdvice
public class BookNotFoundExceptionHandler {

    @ExceptionHandler(BookNotFoundException.class)
    public ResponseEntity handleBadRequestAlertException(BookNotFoundException ex,
                                                         NativeWebRequest request) {
        return ResponseEntity.notFound().build();
    }
}
```

HANDLE EXCEPTION - MICRONAUT

```
public class BookNotFoundException extends RuntimeException {}
```

```
@Get("/books/ex")
public HttpResponse notFound() {
    throw new BookNotFoundException();
}
```

HANDLE EXCEPTION - SPRINGBOOT

```java
import io.micronaut.context.annotation.Requires;
import io.micronaut.http.HttpRequest;
import io.micronaut.http.HttpResponse;
import io.micronaut.http.server.exceptions.ExceptionHandler;

import javax.inject.Singleton;

@Requires(classes = BookNotFoundException.class)
@Singleton
public class BookNotFoundExceptionHandler implements ExceptionHandler<BookNotFoundException, HttpResponse> {

    @Override
    public HttpResponse handle(HttpRequest request, BookNotFoundException exception) {
        return HttpResponse.notFound();
    }
}
```

CURRENT AUTHENTICATED USER - SPRINGBOOT

```
@GetMapping("/authenticate")
public String isAuthenticated(HttpServletRequest request) {
    return request.getRemoteUser();
}
```

CURRENT AUTHENTICATED USER - MICRONAUT

```
import java.security.Principal;

 @Get("/authenticate")
public String isAuthenticated(Principal principal) {
    return principal.getName();
}
```

URI BUILDER - SPRINGBOOT

```
http://localhost:8080/books?page=2&size=20
```

```
String baseUrl = "http://localhost:8080/books";
return UriComponentsBuilder.fromUriString(baseUrl)
                  .queryParam("page", page)
                  .queryParam("size", size).toUriString();
```

URI BUILDER - MICRONAUT

```
http://localhost:8080/books?page=2&size=20
```

```
String baseUrl = "http://localhost:8080/books";
String template = baseUrl + "{?page,size}";
Map<String, Object> arguments = new HashMap<>();
arguments.put("page", page);
arguments.put("size", size);
UriTemplate uriTemplate = new UriTemplate(template);
return uriTemplate.expand(arguments);
```

SPRINGBOOT

ORG.SPRINGFRAMEWORK.STEREOTYPE.SERVICE

MICRONAUT @SINGLETON