01

Testando retornos com JSONs

Transcrição

Vamos agora testar o serviço que nos devolve um usuário de acordo com o seu ID. Dessa vez, o retorno será em JSON. A URL para pegarmos o usuário 1, por exemplo, http://localhost:8080/usuarios/show?usuario.id=1&_format=json (http://localhost:8080/usuarios/show?usuario.id=1&_format=json) .

O teste é parecido com o anterior, mas dessa vez precisamos mudar duas coisas:

- Precisamos passar um parâmetro pela *querystring* ("usuario.id").
- Precisamos tratar o retorno que vem em JSON.

O tratamento em JSON é bem simples. O Rest-Assured abstraiu bem isso para nós. Lembra do `XmlPath` ? Pois bem, ele tem também o `JsonPath` , que é idêntico. A sintaxe é a mesma. Veja só como faríamos se o teste anterior fosse em JSON:

```
@Test
public void deveRetornarListaDeUsuarios() {
    JsonPath path = given()
            .header("Accept", "application/json")
            .get("/usuarios")
            .andReturn().jsonPath();

    List<Usuario> usuarios = path.getList("list.usuario", Usuario.class);

    Usuario esperado1 = new Usuario(1L, "Mauricio Aniche", "mauricio.aniche@caelum.com.br");
    Usuario esperado2 = new Usuario(2L, "Guilherme Silveira", "guilherme.silveira@caelum.cor

    assertEquals(esperado1, usuarios.get(0));
    assertEquals(esperado2, usuarios.get(1));

}
```

Repare que só mudamos de um para outro.

Para passar o parâmetro também é bem simples. Podemos usar o método `parameter()` , que recebe o nome do parâmetro, bem como o conteúdo a ser enviado. Veja o novo teste completo:

```
@Test
public void deveRetornarUsuarioPeloId() {
    JsonPath path = given()
            .parameter("usuario.id", 1)
            .header("Accept", "application/json")
            .get("/usuarios/show")
            .andReturn().jsonPath();

    Usuario usuario = path.getObject("usuario", Usuario.class);
    Usuario esperado = new Usuario(1L, "Mauricio Aniche", "mauricio.aniche@caelum.com.br");

    assertEquals(esperado, usuario);
```

```
    }
```

Note que estamos passando o ID=1, e JSON no Accept. Ao rodar o teste, ele passa.

Mesmo se seu JSON não voltar um objeto que possa ser desserializado, a API de Path (tanto de XML quanto de Json) possibilita que você apenas navegue por ele. Por exemplo, se quiséssemos pegar apenas a String contendo o nome "Mauricio Aniche", poderíamos fazer:

```
path.getString("usuario.nome")
```

Veja os métodos disponíveis: getBoolean(), getDouble(), entre outros. É importante conhecê-los para que você faça bom uso da biblioteca.