01

Enviando dados para o WebService

Transcrição

Até o momento só consultamos os dados, vamos agora consumir *webservices* que os recebem. Adicionando um usuário, serviço está disponível em `http://localhost:8080/usuarios (http://localhost:8080/usuarios)` . Para inserir um usuário, precisamos fazer um POST com um XML (ou JSON), o formato é o mesmo da nossa entidade `Usuario` .

Vamos começar nosso teste, então, criando o usuário que será adicionado:

```
@Test
public void deveAdicionarUmUsuario() {
    Usuario joao = new Usuario("Joao da Silva", "joao@dasilva.com");

}
```

Agora usaremos a simples API do Rest-Assured para fazer o POST. Essa requisição é mais complicada. Precisamos:

- Dizer que a requisição enviará em XML (Content-Type no Header)
- Dizer que o retorno deve ser em XML também (Accept)
- O corpo da requisição deve ser o objeto "joao" serializado
- O retorno do serviço web deve ser 200.

Vamos começar configurando os dados da requisição. Veja que invocamos o método `header()` , `contentType` e `body` . Os três são auto-explicativos. Mas repare que o `body` sabe que o objeto deve ser serializado em XML, justamente por causa do valor passado para o *content-type*:

```
@Test
public void deveAdicionarUmUsuario() {
    Usuario joao = new Usuario("Joao da Silva", "joao@dasilva.com");

        given()
            .header("Accept", "application/xml")
            .contentType("application/xml")
            .body(joao);
}
```

Em seguida, precisamos fazer o *post*, e receber os dados de volta como XML (da maneira que já estamos acostumados). Com isso, já conseguimos inclusive fazer o *assert*:

```
@Test
public void deveAdicionarUmUsuario() {
    Usuario joao = new Usuario("Joao da Silva", "joao@dasilva.com");

    XmlPath retorno =
        given()
            .header("Accept", "application/xml")
            .contentType("application/xml")
```

```
                .body(joao)
            .when()
                .post("/usuarios")
            .andReturn()
                .xmlPath();

        Usuario resposta = retorno.getObject("usuario", Usuario.class);

        assertEquals("Joao da Silva", resposta.getNome());
        assertEquals("joao@dasilva.com", resposta.getEmail());

    }
```

Por fim, é legal também garantir que o código de retorno do HTTP seja 200. Para isso, usaremos o método `expect()`, que faz a asserção que queremos:

```
    @Test
    public void deveAdicionarUmUsuario() {
        Usuario joao = new Usuario("Joao da Silva", "joao@dasilva.com");

        XmlPath retorno =
            given()
                .header("Accept", "application/xml")
                .contentType("application/xml")
                .body(joao)
            .expect()
                .statusCode(200)
            .when()
                .post("/usuarios")
            .andReturn()
                .xmlPath();

        Usuario resposta = retorno.getObject("usuario", Usuario.class);

        assertEquals("Joao da Silva", resposta.getNome());
        assertEquals("joao@dasilva.com", resposta.getEmail());

    }
```

Excelente. Nosso teste agora passa! Observe novamente que com o uso do *framework* nossos testes são por demais simples de serem escritos.

Veja que aqui estamos fazendo uma inserção. Ou seja, um novo elemento está sendo inserido no serviço externo. Você, desenvolvedor de testes, precisa ficar atento a isso, pois isso pode mudar o resultado de outros testes. Imagine que você tenha um teste que garanta que o serviço que retorne a quantidade de usuários funcione, se você adicionar um novo usuário, o teste vai quebrar, porque o número de usuários cresceu.

Essa discussão aconteceu no exercício do capítulo anterior. Precisamos sempre tomar cuidado com cenários em serviços externos. Se você criou um usuário, você precisa deletá-lo depois. Para isso, você pode usar algum serviço de deleção, que é disponibilizado pelo serviço web.

Configurando o Rest-Assured

Você reparou que até agora temos passado endereços relativos, mas ele sabe que nossa aplicação está em `http://localhost:8080 (http://localhost:8080)` ? Pois bem, isso é a configuração padrão dele. Você pode mudá-la. Por exemplo, se quiséssemos mudar a URL base, faríamos:

```
RestAssured.baseURI = "http://www.meuendereco.com.br";
RestAssured.port = 80;
```

Geralmente colocamos esse tipo de configuração em algum lugar centralizado. Por exemplo, em algum `@Before` ou `@BeforeClass` da bateria de testes. Assim, fica fácil mudar essa configuração caso o endereço do serviço mude.