* Používa sa knižnica Rest Assured
* Použíté knižnice v dependency:

Junit v 4.12

io.rest-assured v 4.1.2

* Prvy priklad testu statusu 200 OK:
* *when*().get("https://www.potterapi.com/v1/spells?key=$2a$10$CbVViIAfPIiG95DPiktzQemq9YSvQqH0H/JwdT0yrUCbnZvVll7iK")  
   .then().statusCode(200)  
   .and().contentType(ContentType.*JSON*);

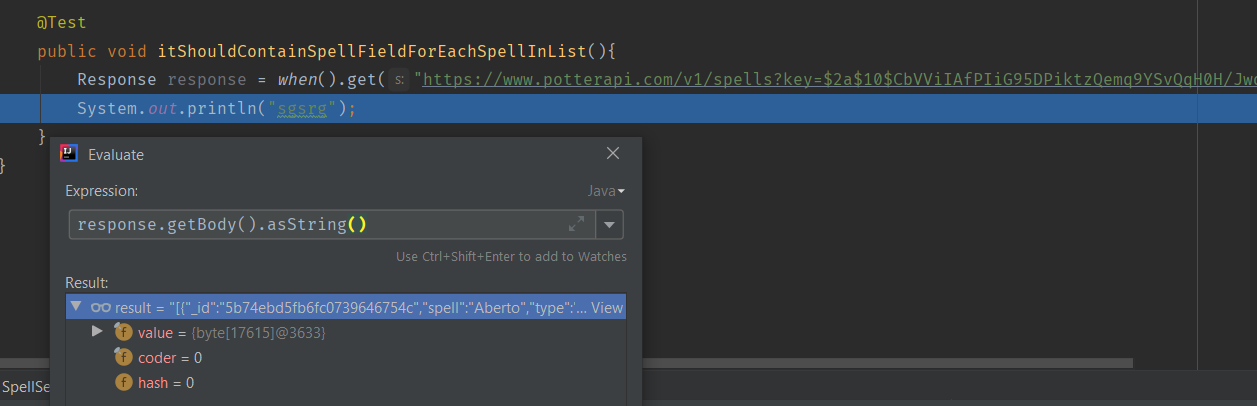
//ContentType sleduje format navratovej hodnoty

Statusy:

* 200 - OK
* 409 - chyba kluc
* 401 - kluc je nespravny (invalid)

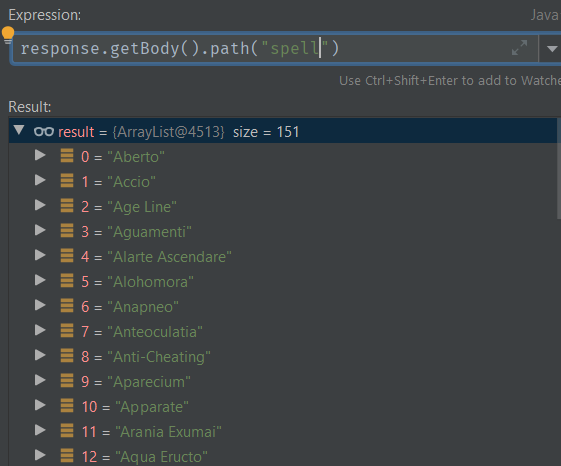
@Test  
public void itShouldReturnStatus409WhenKeyIsMissing(){  
 *when*().get("https://www.potterapi.com/v1/spells")  
 .then().statusCode(409)  
 .and().contentType(ContentType.*JSON*)  
 .and().body("error", *is*("Must pass API key for request"));  
 //nasimulovanie statusu 409 (Conflict) v pripade ze nemam autorizaciu voci API(chyba kluc)  
 //pomocou body kontrolujem text vratenej hlasky, is je z kniznice org.hamcrest.Matchers.is  
}

* Pokial chcem oddebugovat a zistit co obsahuje odpoved, spravim to nasledujucim postupom:



Na nejaky sout nastavim zastavenie debugu, spustim debug, dam ALT+F8 a mozem pisat prikaz na zaklade ktoreho chcem zistit odpoved:

* + response.getBody().asString() - vrati cely string odpovede
  + response.getBody().path(“spell”) - vrati list vsetkych spellov zo vsetkych odpovedi:



* Pocet opakujucich sa elementov v odpovedi, ulozene do listu a metody na listoch:

@Test  
public void itShouldContainSpellFieldForEachSpellInList(){  
 Response response = *when*().get("https://www.potterapi.com/v1/spells?key=$2a$10$CbVViIAfPIiG95DPiktzQemq9YSvQqH0H/JwdT0yrUCbnZvVll7iK");  
 List<String> spells = response.getBody().path("spell");  
 *assertThat*(spells,*hasSize*(*greaterThan*(10)));  
 //pocet spellov je viac ako 10  
  
 *assertThat*(spells,*hasSize*(151));  
 //pocet spellov je presne 151  
  
 spells.forEach(spell -> *assertThat*(spell,*is*(*not*(*emptyString*()))));  
 //spelly neobsahuju prazdny string  
}

* Po nastaveni konfiguracii basePath a baseURL mozem zmazat linky z get:

@BeforeClass  
//spusti sa raz pred vsetkymi testami  
public static void configuration(){  
 RestAssured.*baseURI* = "https://www.potterapi.com/v1";  
 RestAssured.*basePath* = "/spells";  
 //base path je endpoint, ktory mozem pre klasy menit  
}

Po pridani tohto prikazu vo vsetkych get metodach necham bud prazdne zatvorky alebo dam prec zaciatok textu:

<https://www.potterapi.com/v1/spells>

Cize when by vyzeral s klucom nasledovne: when().get("?**key**=$2a$10$CbVViIAfPIiG95DPiktzQemq9YSvQqH0H/JwdT0yrUCbnZvVll7iK")

* Po este lepsej uprave pouzijeme given (queryParam) do ktoreho dam kluc a kod je viac citatelny:
* @Test  
  public void itShouldReturnStatus200WhenKeyIsCorrect(){  
   *given*().queryParam("key",*API\_KEY*)  
   .when().get()  
   .then().statusCode(200)  
   .and().contentType(ContentType.*JSON*);  
   //kluc pre prihlasenie na web je v casti key=$2a$10$CbVViIAfPIiG95DPiktzQemq9YSvQqH0H/JwdT0yrUCbnZvVll7iK  
   //metoda v danom prikaze je spells, potrebovali sme sa ale autentifikovat  
   //ContentTypeJson kontroluje ci odpoved dostavam v Json formate  
   //pomocou given zadam kluc vo public static final Stringu  
  }

Pomocou given si viem upravit pathParam, baseURL aj basePath ale uz ich mam zadefinovane na zaciatku programu

* Porovnanie backendu s frontendom:

@Test  
public void itShouldDisplayEffectsOnPage(){  
 Response response = *given*().queryParam("key",*API\_KEY*).when().get();  
 List<String> effects = response.getBody().path("effect");  
 //vytiahnutie effektov z API (backend)  
  
 *open*("http://localhost:82/spelleology.php");  
 *$*("ul.spells")  
 .findAll("li")  
 .shouldHave(*exactTexts*(effects));  
  
 //najdenie elementov na frontende a porovnanie listov s backendom  
}

* **Json validator schémy**:

Dependency:

<dependency>

<groupId>io.rest-assured</groupId>

<artifactId>json-schema-validator</artifactId>

<version>4.1.2</version>

</dependency>

Samotná schéma:

{  
 "type": "array", //znamena ze definujem pole  
 "items": {  
 "required": [ //povinne elementy  
 "\_id",  
 "spells",  
 "effect",  
 "type"  
 ],  
 "properties": { //do properties definujem typy premennych pre jednotlive elementy  
 "\_id": {  
 "type": "string",  
 "minLength": 10  
 },  
 "spell": {  
 "type": "string",  
 "pattern": "^[A-Z](.\*)$" //definovany regularny vyraz pre element spell  
 },  
 "type": {  
 "type": "string",  
 "minLength": 5  
 },  
 "effect": {  
 "type": "string"  
 }  
 }  
 }  
}

Test na základe schémy:

@Test  
public void itShouldMatchSchema(){  
 *given*().queryParam("key",*API\_KEY*)  
 .when().get()  
 .then().body(*matchesJsonSchema*(new File("src/test/resources/schemas/spell\_schema.json")));  
  
}

* + - machesJsonSchema može byť definované aj inak ako cez file, je to vloženie mojej schémy do kódu pomocou file v tomto prípade
* Querry parametre (**filtrácia** na základe elementov):
* @Test  
  public void itShouldFindCharacterUsingHouseAndDeathEaterParam(){  
   *given*().queryParam("key",*API\_KEY*)  
   .queryParam("house","Gryffindor")  
   .queryParam("deathEater",true)  
   .when().get()  
   .then().body("[0].name",*is*("Peter Pettigrew"));  
  }

//filtracia elementu house, najde vsetky house s hodnotou Gryffindor a sucasne najde vsetky deathEater s hodnotou true, nakoniec skontrolujem ci je v danom zozname meno Peter Pettigrew

* + volanie api s vyssie popisanymi query parametrami vyzera nasledovne:

<https://www.potterapi.com/v1/characters?key=$2a$10$CbVViIAfPIiG95DPiktzQemq9YSvQqH0H/JwdT0yrUCbnZvVll7iK&house=Gryffindor&deathEater=true>

//characters je metoda

//key je autentifikacny kluc (po metode musi byt ? a az potom zadavat querry parameter)

//filtracia na zaklade hodnôt elementov house a deathEater

* Kontrola postavy na zaklade **ID**:
* @Test  
  public void itShouldFindCharacterUsingIdRoute(){  
   *given*().queryParam("key",*API\_KEY*)  
   // .when().get("/5a12292a0f5ae10021650d7e")  
   .pathParam("characterId","5a12292a0f5ae10021650d7e")  
   .when().get("/{characterId}")  
   .then().body("name",*is*("Harry Potter"));  
   //do when mozem vlozit priamo id alebo zadefinovany parameter characterId pomocou pathParam (tymto sposobom je  
   // je v tom vacsi prehlad, v pripade aj ked pathParam vlozim do inej metody  
  }

//do get() doplnim ID postavy a tak vyfiltrujem postavu na zaklade ID a potom uz len spravim kontrolu, ci dana postava je Harry Potter