**Integrációs tesztelés**

**webesre 3 féle:**

**1) Webes unit teszt: csak Controller teszt és a Service mockolt, @WebMvcTest, @MockBean használat (ez mégn nem integrációs)**

**2) teljes alkalmazás indításával(=@SpringBootTest , eredeti service működés is!) konténer nélkül, mockMVc-vel a servlet API kimockolásával**

**3) teljes alkalmazás indításával(=@SpringBootTest , eredeti service működés is!) konténerrel, RestTemplate-el és ControllerRestTestIT osztály**

**konténerrel, teljes alkalmazás konténer nélkül; mockMVc a servlet APIt kimockolja;**

**webszolgáltatások tesztelésére** több mód: Springgel **kizárólag a Controller réteget indítjuk el és a Servicet mockoljuk**; de elindíthatjuk a teljes alkalmazást is: 1**)valódi konténerrel. Tomcat fut,** foglal egy portot és meg lehet szólítani; 2)**vagy webkonténer nélkül**, így ki lesz mockolva a servlet API

**1.csak a Controller réteget indítjuk el, és a serviceket mockoljuk**

@SpringBootTest helyett **@WebMvcTest** annotáció

a mock objektumokat a **@MockBean-el** látjuk el

**MockMvc** **kis keretrendszer kérések beküldésére és válaszok ellenőrzésére,** naplózására, a válaszok különböző feldolgozására (Stringként vagy jSONként)

**a MockMvc nem indít valódi konténert, a servlet API-t kimockolja**, de a jSON szerializáció egyébként megtörténik

**WEB réteg tesztelése – példa**

when-nel service réteg mockolása

majd mockMvc fluent APIval; get metódus meghívása performmal az adott urlen, visszaadott kód ellenőrzése és andDo-val kiírása a konzolra végül jSonPath-al annak levizsgálása, hogy a jSONben visszakapott tömb nulladik eleme egy objektum, amiben a name mező értéke az e, ami



**Teljes alkalmazás tesztelése konténer nélkül**

@SpringBootTest és @AutoConfigureMockMvc(=MockMvc keretrendszert engedélyezi; valódi Servicet fog használni) annotációk



**Teljes alkalmazás tesztelése konténerrel = RestTemplate**

**random porton induljon a Tomcat: @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM\_PORT); RestTemplate teszt verziójának használata -> REST kliensként http requesteket tudunk vele kiadni kódból; a template elvégzi a jSON-java objektum konvertálást**

**Tomcat** is el lesz **indítva**

**Random porton kerüljön elindításra**: ne akadjanak össze különböző Tomcat konténerek portjai több alkalmazás futása esetén pl. egy Continuous Integration környezeten

**@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM\_PORT)**

Amennyiben tudni szeretnénk, hányas porton fut a konténer, akkor a port számát a @LocalServerPort annotációval injektálhatjuk a tesztesetbe.

**Mire való a RestTemplate és mit ellenőrizünk vele?**

Keretrendszer a Springen belül, ami REST kliensként működik és REST kéréseket tudunk vele kiadni java-ban = **API, amivel http requesteket lehet küldeni java kódból. Ennek a** teszt verzióját (=TestRestTemplate) automatikusan injektljuk tesztesetekbe **(=@AutoWired TestRestTemplate template**).

**Azt ellenőrízzük vele, hogy a megfelelő urlen megfelelő paramétereket átadva(=http request) tényleg visszaadásra kerül-e a Dto.**

a template használata esetén **a háttérben megtörténik a jSON kommunikáció** és a **template fogja jSONból Java objektummá (Dto) visszaalakítani a jSON dokut**; erre példa:



az urlen egy employee listát fogunk visszakapni, json formátumban; a RESTEMPLATE automatikusan deszerializálja java objektummá

**Integrációs tesztelés - gyakorlat - MockMvc**

**Integrációs tesztelés - gyakorlat – RestTemplate**

tesztelés a teljes alkalmazás meghajtásával, valós service osztállyal

a http kérést nem mockMVCvel hanem RestTemplate-el adjuk ki

0.új osztály: EmployeesControllerRestTemplateIT; @SpringBootTest(random port!) és **@Autowired TestrestTemplate**

1.test metódus: testListEmployees();

**EmployeeDto dto = postforObject(request: url és new CeateEmployeeCommand és a visszatérés EmployeeDto) ->** erre egy getnames assert

Tomcat elindul, benne az alkalmazás contorllerrel serviccel; a http kérés sikeres, zöld

még egy post Jane Dora, majd

2.Get kérés küldése; listát fog visszaadni

template.**exchange**(url, htt metódus, null(=törzsben nem küldök semmit), és meg kell mondani, egy elég furcsa módon, hogy egy EmployeeDto listát várok vissza**, mert javaban class-t lista + generikus típussal alapba nem lehet definiálni**



majd erre assert extractingos containsexactly

elbukott; mert a Serviceben a memóriában alapba hozzáadtunk már 2 nevet; néztünk rá egy @RepeatedTest(2)et; folyamatosan növekedett az alkalmazottak száma -> **az alkalmazást megfelelő állapotra kell hozni, a services memóriában ki kell törölni a futás előtt az employeekat**

Servicebe -> deleteAllEmployees());idgenerator = new() és employees.clear()

tesztesetből meg kell hívni a deletet, meg hozzá atriként felvenni @autowireddel egy servicet

**Kérdések**

*Milyen tesztelési lehetőségek vannak webes alkalmazásokra javaban? (3)*

*Hogy konfiguráljuk a teszt osztályt a teljes alkalmazás konténerrel való tesztelése esetén?*

*Mire való a RestTemplate és mit ellenőrzünk vele?*

*A RestTemplate-et hogyan injektáljuk a tesztesetekbe?*

*Mit ellenőrzünk vele?*

*Használatakor mi történik a háttérben?*