



JPA

Lliurament

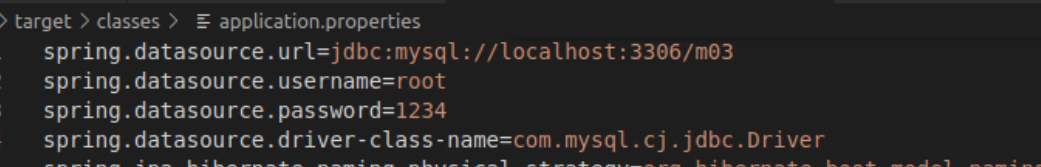
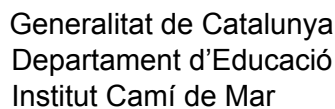
Fes commit de tots els canvis realitzats en una nova branca del teu repositori que es digui uf6NCognom. I quan tinguis tot el codi realitzat, fes un TAG.

Entrega també un PDF a la tasca de moodle amb les captures i respostes que es van demanant al llarg de la pràctica.

Enunciat

1. Partint del codi resultant de la pràctica 1, fes les modificacions necessàries per recuperar la informació de DataSource del fitxer application.properties.
 - 1.1. Descomenta la dependència del pom que està comentada (spring-boot-starter-data-jpa).S
 - 1.2. Crea la classe **DataSourceConfig** que recuperi la informació d'environment i retorni un DataSource amb les propietats del fitxer de configuració.

```
1 package daw.m3.uf6;
2
3 import javax.sql.DataSource;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.jdbc.DataSourceBuilder;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.context.annotation.Configuration;
8 import org.springframework.core.env.Environment;
9
10 @Configuration
11 public class DataSourceConfig {
12
13     @Autowired
14     Environment environment;
15
16     @Bean
17     public DataSource getDataSource() {
18         DataSourceBuilder dataSourceBuilder = DataSourceBuilder.create();
19
20         dataSourceBuilder.driverClassName(environment.getProperty(key:"spring.datasource.driver-class-name"));
21         dataSourceBuilder.url(environment.getProperty(key:"spring.datasource.url"));
22         dataSourceBuilder.username(environment.getProperty(key:"spring.datasource.username"));
23         dataSourceBuilder.password(environment.getProperty(key:"spring.datasource.password"));
24
25         return dataSourceBuilder.build();
26     }
27 }
28
```



```
uf6 > target > classes > application.properties
1  spring.datasource.url=jdbc:mysql://localhost:3306/m03
2  spring.datasource.username=root
3  spring.datasource.password=1234
4  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5  spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategy
6  spring.jpa.show-sql=true
7
8  spring.security.user.name=USER
9  spring.security.user.password=M3DAW
10
11 logging.file.name=./logs/MyApp.log
12 logging.logback.rollingpolicy.file-name-pattern=MyApp-%d{yyyy-MM-dd}.%i.log
13 logging.logback.rollingpolicy.max-file-size=1MB
14 logging.level.daw=debug
```

- 1.3. A la classe **RepositoriJDBCImpl** canvia el necessari per agafar el DataSource generat a la classe creada en el punt 1.1.
- 4.4. ~~[No pots comprovar-ho per ara, ho provaràs al punt 3. Revisa però que al main d'Uf6Application només s'aixeca el context d'spring, i no hi ha res més, únicament `SpringApplication.run(Uf6Application.class, args)`3] Comprova que tot el codi segueix funcionant correctament.~~
2. Revisa el codi que s'ha facilitat, aixeca l'aplicació d'Spring utilitzant el mètode main de la classe Java **Uf6Application**.
 - 2.1. **Prova de llançar una petició Postman a un dels dos endpoints** que ja estan creats al controller. Fes una captura demostrant el correcte funcionament i com es tracen les informacions que hi ha als mètodes definits.

GET

http://localhost:8080/getAllActors

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

Key	Value	Bulk Edit
Key	Value	

Body

Cookies (1)

Headers (13)

Test Results

Status: 500 Internal Server Error

Time: 110 ms

Size: 10.59 KB

Save Response

Pretty

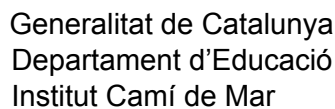
Raw

Preview

Visualize

JSON

```
1 {  
2   "timestamp": "2024-04-30T14:59:45.366+00:00",  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "trace": "java.lang.UnsupportedOperationException: M\u00e9thode no implementat\n\tat daw.m3.uf6.services.ActorService.getAllActors(ActorService.java:22)\n\tat daw.m3.uf6.controllers.AppController.getAllActors(AppController.java:44)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Native Method)\n\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)\n\tat java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)\n\tat java.base/java.lang.reflect.Method.invoke(Unknown Source)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:259)\n\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:192)\n\tat org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:118)\n\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleRequest(RequestMappingHandlerAdapter.java:293)\n\tat org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter.handle(AnnotationMethodHandlerAdapter.java:274)\n\tat org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter.handle(AnnotationMethodHandlerAdapter.java:274)\n\tat org.springframework.test.web.servlet.MockMvc.perform(MockMvc.java:204)"
```



3. Modifica l'ActorService:

- POST

http://localhost:8080/createActor

Send

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

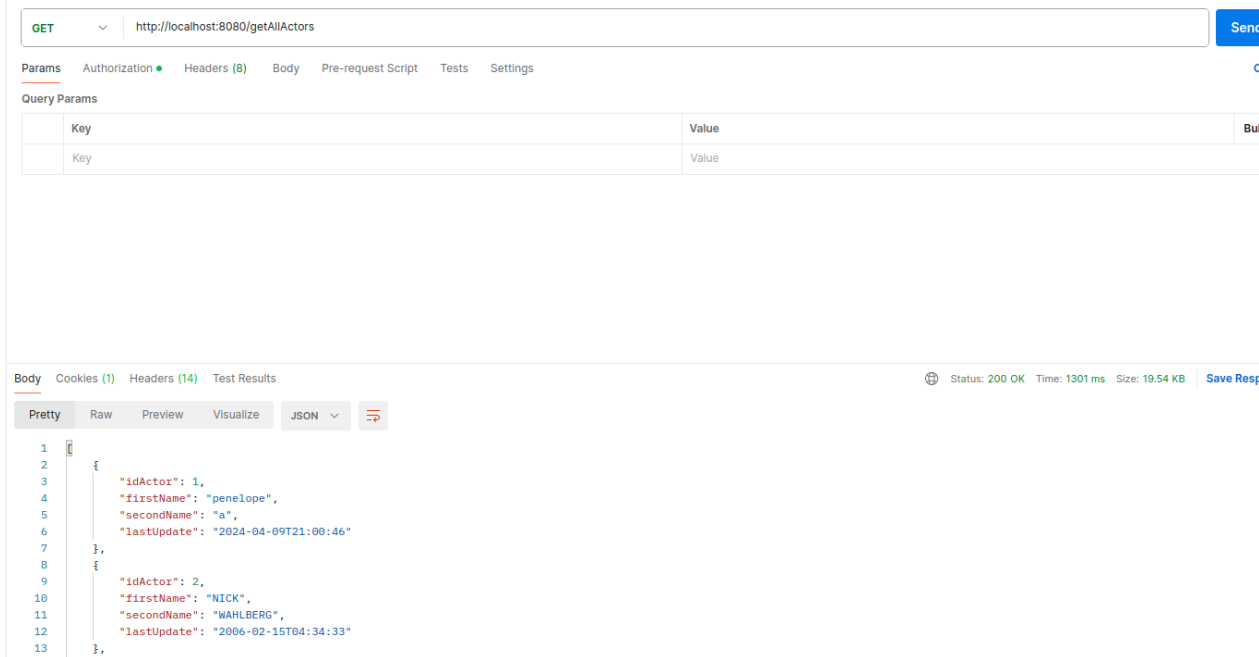
504

505

506

5

- 3.3. Llança petició Postman i adjunta la captura conforme funciona la petició a getAllActors.



- 3.4. Llança petició Postman i adjunta la captura conforme funciona la petició a createActor. Adjunta també captura de la taula de BD on es demostri que s'ha insertat l'actor de forma correcta.



4. Realitza els canvis necessaris en la classe **Actor** per tal que representi exactament la taula **Actor**.



```
src > main > java > daw > m3 > uf6 > objects > http > RequestActor.java > RequestActor > se
package daw.m3.uf6.objects.http;

import java.time.LocalDateTime;

public class RequestActor {
    private int idActor;
    private String firstName;
    private String secondName;
    private LocalDateTime lastUpdate;

    public int getIdActor() {
        return idActor;
    }
    public void setIdActor(int idActor) {
        this.idActor = idActor;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getSecondName() {
        return secondName;
    }
}
```

http://localhost:8080/getAllActors

GET http://localhost:8080/getAllActors Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
Key	Value	

Body Cookies (1) Headers (14) Test Results Status: 200 OK Time: 347 ms Size: 19.63 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "idActor": 1,
3   "firstName": "penelope",
4   "secondName": "a",
5   "lastUpdate": "2024-04-09T21:00:46"
6 },
7 {
8   "idActor": 2,
9   "firstName": "NICK",
10  "secondName": "WAHLBERG",
11  "lastUpdate": "2006-02-15T04:34:33"
12 },
13 }
14 }
```



5. Crea un classe **ActorRepositoryJPA** al package Repositories que extengui de JpaRepository (bastant-te en la teoria).

```
elp
pom.xml Actor.java AppController.java 9+ RequestActor.java ActorService.java 3 RepositoriJDBCImp.java ActorRepositoryJPA.java x ActorRepository.java
src > main > java > daw > m3 > uf6 > repositories > ActorRepositoryJPA.java > ActorRepositoryJPA
package daw.m3.uf6.repositories;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import daw.m3.uf6.objects.Actor;

@Repository
public interface ActorRepositoryJPA extends JpaRepository<Actor, Long> {}
```

6. Modifica de nou **ActorService**:
 - 6.1. Afegeix la dependència del **Repositori** que acabes de crear.
 - 6.2. **Modifica els dos mètodes existents**, de manera que després d'haver executat el mètode del repositori JDBC que vas crear a la pràctica 1, executi ara el mètode d'ActorRepositoryJPA predefinit (no l'has d'implementar tu).
 - 6.3. Llança petició Postman i **adjunta la captura** conforme funciona la petició a **getAllActors**.
 - 6.4. Llança petició Postman i **adjunta la captura** conforme funciona la petició a **createActor**. **Adjunta també captura de la taula de BD** on es demostri que s'ha insertat l'actor de forma correcta.

tengo un error el cual no me permite continuar con el, querria mirarlo contigo

7. Crea un nou endpoint **"/updateActor/{id}"** a la classe AppController (escull el mètode HTTP que creguis més adequat). Rebrà un objecte JSON com el de "createActor" i el paràmetre id passat com a "pathVariable" on s'indicarà l'id de l'actor que es vol actualitzar. Afegeix les traces que permetin veure que funciona. **[De moment NO FARÀ RES MÉS QUE TRAÇAR]**
 - 7.1. Llança petició Postman i **adjunta la captura** conforme funciona la petició a **/updateActor/1**.



HTTP http://localhost:8080/createActor

GET http://localhost:8080/updateActor/1

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies (1) Headers (15) Test Results

Status:

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2024-04-30T20:37:53.959+00:00",
3   "status": 405,
4   "error": "Method Not Allowed",
5   "trace": "org.springframework.web.HttpRequestMethodNotSupportedException: Request method 'GET' is not supported\n\tat org.s
    RequestMappingInfoHandlerMapping.handleNoMatch(RequestMappingInfoHandlerMapping.java:265)\n\tat org.springframework.web
    lookupHandlerMethod(AbstractHandlerMethodMapping.java:441)\n\tat org.springframework.web.servlet.handler.AbstractHandle
    (AbstractHandlerMethodMapping.java:382)\n\tat org.springframework.web.servlet.mvc.method.RequestMappingInfoHandlerMappi
```

8. Afegeix a **ActorService** un mètode **updateActor** que rebrà un Actor i retornarà un Actor. Validarà que existeixi un actor amb l'id rebut, i en cas afirmatiu a posteriori, guardarà l'actor actualitzat. Ha d'utilitzar sí o sí l'**ActorRepositoriJPA** amb els mètodes predefinits que té (revisa la teoria per saber quin mètode utilitzar).



The screenshot shows a Postman client interface. The top part displays a PUT request to `http://localhost:8080/updateActor/1` with a JSON body: `{ "firstName": "John", "secondName": "YO" }`. The bottom part shows the response body in JSON format: `{ "idActor": 0, "firstName": "John", "secondName": "YO", "lastUpdate": null }`. Below this, a database table view is shown with columns `actor_id`, `first_name`, `last_name`, and `last_update`. The table contains two rows: one for `actor_id` 1 (John YO) and another for `actor_id` 214 (John Doe).

actor_id	first_name	last_name	last_update
1	John	YO	2024-04-30 22:51:36
214	John	Doe	2024-04-30 20:52:58

9. **Modifica el controller** per tal que **updateActor** cridi a **updateActor** del Service. Fes també les modificacions necessàries per llançar i controlar les possibles excepcions (tant a controller com a Service).
- 9.1. En cas que es **pugui actualitzar** l'actor correctament retorna un **200** i l'actor actualitzat.
 - 9.2. En cas que **NO es pugui actualitzar per què l'id NO existeix**, retorna un **404**, i el missatge d'error que indica que no s'ha trobat.
 - 9.3. En cas que **NO es pugui actualitzar per qualsevol altre motiu**, retorna un **500**, error intern en el servidor (traceu però l'error al logger.)
 - 9.4. Llança petició postman **per CADA UN dels casos anteriors** i **adjunta la captura** conforme funciona i es comporta com s'ha demanat, **adjunta també captura de la BD** on es vegi el registre actualitzat.



PUT ⌵ http://localhost:8080/updateActor/1

Params Authorization ● Headers (10) Body ● Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON ⌵

```
1 {  
2   "firstName": "John",  
3   "secondName": "Y0prprp"  
4 }
```

body Cookies (1) Headers (14) Test Results

Pretty Raw Preview Visualize JSON ⌵ ↺

```
1 {  
2   "idActor": 1,  
3   "firstName": "John",  
4   "secondName": "Y0prprp",  
5   "lastUpdate": "2024-04-30T22:51:36"  
6 }
```

The screenshot shows an IDE with a Java file named `AppController.java` and a database table view for the `actor` table.

Java Code:

```
11 public ResponseEntity<Actor> updateActor(@PathVariable Long id, @RequestBody RequestActor actorUpdateReq) {
12     try {
13         // Verificar si el actor a actualizar existe
14         Actor existingActor = actorService.getActorById(id);
15         if (existingActor == null) {
16             logger.error(message: "Error interno en el servidor al actualizar el actor");
17             AppErrorResponse errorResponse = new AppErrorResponse(ErrorMessage.E404.name());
18             return new ResponseEntity(errorResponse, errorResponse.getHttpCode());
19         }
20
21         // Validar los campos del actor actualizado
22         HashMap<String, String> validations = new HashMap<>();
23         if (actorUpdateRequest == null) {
24             validations.put("actor", "Objeto actor inválido.");
25             logger.error(message: "Error interno en el servidor al actualizar el actor");
26             AppErrorResponse errorResponse = new AppErrorResponse(ErrorMessage.E001.name());
27             return new ResponseEntity(errorResponse, errorResponse.getHttpCode());
28         }
29
30         existingActor.setFirstName(actorUpdateRequest.getFirstName());
31         existingActor.setSecondName(actorUpdateRequest.getSecondName());
32
33         Actor updatedActor = actorService.updateActorById(id, existingActor);
34
35         // Retornar un 200 con el actor actualizado
36         return ResponseEntity.ok(updatedActor);
37     } catch (Exception e) {
38         // En caso de cualquier otro error, retornar un 500 con un mensaje de error genérico
39         logger.error(message: "Error interno en el servidor al actualizar el actor", e);
40         AppErrorResponse errorResponse = new AppErrorResponse(ErrorMessage.E500.name());
41         return new ResponseEntity(errorResponse, errorResponse.getHttpCode());
42     }
43 }
```

Database Table View:

actor_id	first_name	last_name	last_update
1	John	YOprprp	2024-04-30 23:15:08

10. Crea un nou endpoint `/deleteActor` que **rebrà un id** com a **requestParam** (fes servir el mètode HTTP que creguis més adequat), implementa tots els passos necessaris (a totes les classes que pertorqui) per poder-ho dur a terme, tenint en compte l'estructura del projecte. S'ha d'utilitzar l'`ActorRepositoriJPA` i els mètodes predefinits.
 - 10.1. En cas que es pugui esborrar correctament, retorna un 200 i un objecte que té un atribut "message" que és un String i indica que l'actor s'ha esborrat correctament.
 - 10.2. En cas que **NO es pugui actualitzar per què l'id NO existeix**, retorna un **404**, i el missatge d'error que indica que no s'ha trobat.
 - 10.3. En cas que **NO es pugui actualitzar per qualsevol altre motiu**, retorna un **500**, error intern en el servidor (traceu però l'error al logger.)
 - 10.4. Llança petició postman **per CADA UN dels casos anteriors** i **adjunta la captura** conforme funciona i es comporta com s'ha demanat, **adjunta també captura de la BD** on es vegi el registre actualitzat.



+ Opciones

				actor_id	first_name	last_name	last_update	1
<input type="checkbox"/>				1	John	YOprprp	2024-04-30 23:15:08	
<input type="checkbox"/>				214	John	Doe	2024-04-30 20:52:58	
<input type="checkbox"/>				213	toni	toni	2024-04-09 21:13:30	

http://localhost:8080/createActor

DELETE http://localhost:8080/deleteActor?id=214

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

Key	Value
<input checked="" type="checkbox"/> id	214
Key	Value

Body Cookies (1) Headers (14) Test Results

Status: 200 OK 1

Pretty

Raw

Preview

Visualize

JSON

1 true

1

>

>>

☐

Mostrar todo

Número de filas:

25

Filtrar filas:

Buscar en es

+ Opciones

				actor_id	first_name	last_name	last_update	1
<input type="checkbox"/>				1	John	YOprprp	2024-04-30 23:15:08	
<input type="checkbox"/>				213	toni	toni	2024-04-09 21:13:30	
<input type="checkbox"/>				212	toni	toni	2024-04-09 21:00:17	
<input type="checkbox"/>				211	toni	toni	2024-04-09 15:55:28	
<input type="checkbox"/>				210	toni	jorda	2024-04-09 15:49:37	