

Previously on Betabeers Málaga...

<http://es.slideshare.net/escobeitor1/introduccion-a-android-annotations>
<https://github.com/josescgar/BetaLoL>

Spring Boot

Crea tu API RESTful a toda
pastilla

@SpringBootApplication

```
public class MyAPIStarter {  
  
    public static void main(String[] args) {  
        SpringApplication.run(MyAPIStarter.class, args);  
    }  
}
```

FIN



¿Preguntas?

Yo



Twitter: @jescobeitor

Email: hello@escobeitor.com

LinkedIn: <http://linkedin.com/in/jaescobar/>

Spring

Lo bueno...

- Framework Java de referencia
- Inyección de dependencias
- Modularidad
- Documentación
- Librerías (MVC, AOP, Cloud, Data, etc...)

Lo no tan bueno...

- Configuración inicial

Spring Boot

- Configuración por defecto de Spring
- Listo para producción
- Un solo JAR ejecutable
- Servidor de aplicaciones embebido (Tomcat)
- Configuración automática siempre que sea posible
- Todas las ventajas de Spring

Hate Notepad



<https://github.com/josescgar/HateNotepad>

API Pública para registro de Trolls y
gente odiosa en general

“Porque odiar a trolls es gratis...”

Dependencias

```
dependencies {  
    testCompile group: 'junit', name: 'junit', version: '4.11'  
  
    compile("org.springframework.boot:spring-boot-starter-web:$springBootVersion")  
    compile("org.springframework.boot:spring-boot-starter-security:$springBootVersion")  
    compile("org.springframework.boot:spring-boot-starter-data-mongodb:$springBootVersion")  
    compile("org.springframework.boot:spring-boot-starter-actuator:$springBootVersion")  
  
    testCompile("org.springframework.boot:spring-boot-starter-test:$springBootVersion")  
}
```

spring-boot-starter-web
spring-boot-starter-security
spring-boot-starter-data-mongodb
spring-boot-starter-test

Starter

```
@SpringBootApplication
public class HatefulStarter {

    public static void main(String[] args) {
        SpringApplication.run(HatefulStarter.class, args);
    }
}
```

- Incluir en el paquete padre del proyecto
- Escanea todas las clases en directorios inferiores
- Arranca el servidor y la aplicación

Configuración

```
spring:
  profiles.active: development
  security:
    basic:
      enabled: false
hatenote:
  notesPerPage: 5
management:
  context-path: /audit
```

```
---
spring:
  profiles: development
  data:
    mongodb:
      host: 127.0.0.1
      port: 27017
      database: hatedb
```

```
---
spring:
  profiles: production
  data:
    mongodb:
      host: XXX.XXX.XXX.XXX
      port: 27017
      database: hatedb
```

- Opcional
- .xml
- .properties
- .yaml
- Java (@Configuration)

Configuración

```
@Configuration
@EnableConfigurationProperties
@ConfigurationProperties(prefix = "hatenote")
public class HatenoteConfiguration {

    private int notesPerPage;

    public int getNotesPerPage() {
        return notesPerPage;
    }
}
```

Seguridad

Spring security

- Autenticación
- Autorización
- Gestión de credenciales
- Control de acceso
- **Configuración opcional**
- Por defecto: Autenticación HTTP básica con contraseña global aleatoria

Seguridad

```
@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http.authorizeRequests()
            .antMatchers("/person/insert").hasRole("ADMIN")
            .antMatchers("/audit/**").hasRole("AUDITOR")
            .antMatchers("/note/**/delete").hasRole("ADMIN")
            .antMatchers("/**").hasRole("USER")
            .anyRequest().authenticated();

        http.httpBasic();
        http.csrf().disable();

    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.inMemoryAuthentication()
            .withUser("admin").password("admin").roles("USER", "AUDITOR", "ADMIN").and()
            .withUser("escobeitor").password("manolo").roles("USER");
    }
}
```

Repositorios

Spring data

- ¿Ni p*** idea de MongoDB/xSQL? No hay problema
- Definición de queries de forma semántica
- Operaciones CRUD “Out of the box”
- 0 código necesario

Repositorios

```
@Document
public class HateNote {

    @Id
    private String id;

    @NotEmpty
    private String message;

    private Date created;

    @Indexed
    private String hatefulPerson;













    public HateNote(String message, Date created, String hatefulPerson) {
        this.message = message;
        this.created = created;
        this.hatefulPerson = hatefulPerson;
    }
}
```


Repositorios

```
public interface HateNoteRepository extends MongoRepository<HateNote, Serializable> {  
  
    /**  
     * Returns all hate notes directed to the given person with pagination  
     * @param hatefulPerson The hateful person id  
     * @param page Results page number  
     * @return  
     */  
    Page<HateNote> findByHatefulPerson(String hatefulPerson, Pageable page);  
  
    /**  
     * Deletes all notes for a given person  
     * @param hatefulPerson The person's id  
     */  
    long deleteByHatefulPerson(String hatefulPerson);  
}
```

Repositorios

hateNoteRepository.find_

		findByHatefulPerson (String hatefulPerson, Pageable page)	Page<HateNote>
		findAll ()	List<HateNote>
		findAll (Iterable<Serializable> ids)	Iterable<HateNote>
		findAll (Pageable pageable)	Page<HateNote>
		findAll (Sort sort)	List<HateNote>
		findOne (Serializable id)	HateNote
Use Ctrl+Shift+Enter to syntactically correct your code after completing (balance parentheses etc.) >> π			

Controladores

Nuestra API

- Controlador REST → `@RestController`
- Endpoint REST →
`@RequestMapping(value = "/XXXX", method = YYY)`
- Variable en la URL → `@PathVariable`
- Variable en el body → `@RequestParam`
- Responder con objeto JSON → `@ResponseBody`

Controladores

Inyección de dependencias

```
@Autowired  
HatefulPersonRepository hatefulPersonRepository;
```

- El verdadero potencial de Spring
- Modularización

```
if(!hatefulPersonRepository.exists(person)) {  
    throw new RuntimeException("No person found with ID " + person);  
}
```

@Autowired

Controladores

@Autowired

Controladores

to wire

Controladores

```
@RestController
@RequestMapping(value = "/note")
public class HateNoteController {

    @Autowired
    HatefulPersonRepository hatefulPersonRepository;

    @Autowired
    HateNoteRepository hateNoteRepository;

    @Autowired
    HatenoteConfiguration hatenoteConfiguration;

    @RequestMapping(value = "/{person}/{page}", method = RequestMethod.GET)
    public @ResponseBody NoteListDto getForPerson(@PathVariable String person, @PathVariable int page) {

        if(!hatefulPersonRepository.exists(person)) {
            throw new RuntimeException("No person found with ID " + person);
        }

        PageRequest pageRequest = new PageRequest(page, hatenoteConfiguration.getNotesPerPage());

        Page<HateNote> notes = hateNoteRepository.findByHatefulPerson(person, pageRequest);

        return new NoteListDto(notes.getNumber(), (int) notes.getTotalElements(), notes.getContent());
    }
}
```

Controladores

```
@RequestMapping(value = "/insert", method = RequestMethod.POST)
public String insert(@RequestParam(value = "email", required = true) String email,
                    @RequestParam(value = "nickname", required = true) String nickname,
                    @RequestParam(value = "assholeness", required = true) EAssholeLevel assholeLevel) {

    if(hatefulPersonRepository.findByEmail(email) != null) {
        throw new RuntimeException("User with email " + email + " already exists");
    }

    HatefulPerson person = new HatefulPerson(nickname, email, assholeLevel);
    person = hatefulPersonRepository.save(person);

    return person.getId();
}
```


Excepciones

```
@ControllerAdvice
public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {

    @ExceptionHandler(value = {Exception.class, RuntimeException.class})
    @ResponseBody
    public ResponseEntity<Object> generalExceptionHandler(Exception e) {
        return new ResponseEntity<>("Unexpected error: " + e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
    }

    @Override
    protected ResponseEntity<Object> handleMissingServletRequestParameter(MissingServletRequestParameterException ex) {
        return new ResponseEntity<>("Missing parameter: " + ex.getMessage(), HttpStatus.BAD_REQUEST);
    }
}
```

- A nivel de controlador → @ExceptionHandler
- Globalmente → @ControllerAdvice

Testing

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes = HatefulStarter.class)
@WebAppConfiguration
@ActiveProfiles("development")
public class TestHatefulPersonController extends AbstractHatefulTest {

    @Test
    public void testFindNotAuthorized() {
        try {

            mockMvc.perform(get("/person/get")
                            .param("email", "csgermanico@gmail.com"))
                    .andExpect(status().isUnauthorized());

        } catch (Exception e) {
            Assert.fail();
        }
    }
}
```

- Podemos simular requests HTTP a nuestra API
- Podemos simular métodos, parámetros, headers, etc.

Despliegue

```
$ gradle build
```

```
$ java -jar hatenotepad-1.0.jar
```

¿Intrusismo profesional?

Hemos creado una API RESTful:

- Sin saber que significa RESTful
- Sin tener ni p*** idea de MongoDB
- Nivel de Java: tutorial de Taringa
- ¿Tomcat?



¿Preguntas?