Soporte de Spring Boot para REST

Desarrollo de un API REST con Spring Boot

Toda la potencia de Spring Boot

- En nuestros proyectos REST podemos aprovechar toda la potencia de lo que ya sabemos de Spring Boot
  - Inyección de dependencias
  - Acceso a datos
  - Seguridad
  - Utilidades
  - ...

Soporte para REST

- Controladores orientadores a REST con *@RestController*
  - Combinación de *@Controller* y *@ResponseBody*
  - Se modifica el mecanismo de *renderización de la vista*
  - En lugar de redirigirnos a una plantilla Thymeleaf, JSP… se devuelve directamente el contenido que se envía al cliente.

Soporte para REST

- Controladores orientadores a REST con *@RestController*

```
@RestController
public class GreetingController {
    @GetMapping("/greeting")
    public Greeting greeting(@RequestParam Optional<String> name) {
        return new Greeting(counter.incrementAndGet(),
                            String.format(template, name));
    }
}
```

HttpMessageConverter

- Ya que no vamos a utilizar un motor de plantillas en la vista, tenemos que entregar al cliente el contenido en algún formato.
- Spring Boot, al incluir la dependencia *starter* web, incluye algunos conversores por defecto.
  - *StringHttpMessageConverter*: convierte a cadenas de caracteres.
  - *MappingJackson[2]HttpMessageConverter*: convierte a JSON usando Jackson (o Jackson 2) si está presente en el classpath.

Algunas clases que utilizaremos

- *HttpEntity<T>*: representa una petición o respuesta HTTP
  - *RequestEntity<T>*
  - ***ResponseEntity<T>***: añade el código de estado (*HttpStatus*)
- MediaType: subclase de MIME. Listado de constantes.
- HttpHeaders: representa los encabezados de una petición o de una respuesta.

Seguridad

- Se integra por completo con el esquema de Spring Security
  - Idéntica forma de configurar autorización.
  - Múltiples posibilidades de cara a la autenticación
    - Básica
    - JWT
    - OAuth, OAuth2
    - ...

No solo de servidor vive el hombre...

- También podemos consumir de un API REST con Spring
- Clase `RestTemplate`
- Permite hacer peticiones de todo tipo
- Aprovecha los diferentes *converters* que hemos explicado antes.

Elementos más avanzados

- Spring Data REST
  - Permite transformar rápidamente un repositorio de Spring Data en un API REST
- Spring HATEOAS
  - Nos permite crear fácilmente un API siguiendo el principio HATEOAS (*Hypermedia as the Engine of Application State*)
  - Uso de HAL (*Hypertext Application Language*)

Elementos más avanzados

- Spring REST Docs
  - Permite documentar un servicio REST combinando Spring MVC Test y Asciidoctor.