MANEJO DE SESIONES

1.

HTTPSESSION

HTTP ES UN PROTOCOLO...

- **SIN ESTADOS**
- Cuando realiza una petición, no puede recordar nada de lo anterior.
- Necesita un mecanismo auxiliar para que un dato sobreviva más allá de una petición.
- A dicho mecanismo se le conoce como **sesiones**.

HTTPSESSION

- Interfaz que ofrece Java EE
- Es un Map en el que se almacenan pares clave - objeto.
- Cuando creamos una sesión, se envía una cookie, que el navegador mantiene y reenvía en todas las peticiones siguientes. (*JSESSIONID*).
- La sesión es individual: cada usuario tiene la suya; los datos no se comparten entre sesiones.

PARA QUÉ SE USAN LAS SESIONES

- Spring Security la usa cuando nos logueamos.
  - Así nos permite navegar entre páginas que requieren autenticación sin volver a loguearnos.
- Lógica de nuestra app
  - Carrito de la compra
  - Asistentes (proceso a realizar en varios pasos).

USO DE HTTPSESSION CON SPRING

- Podemos *@Autowired* el objeto *HttpSession* en cualquier controlador.
- Podemos pasarlo como argumento de cualquier método en un controlador.

DESVENTAJAS DE HTTPSESSION

- Dependemos de la implementación del servidor (contenedor de servlets) sobre el que desplegamos nuestra app.
- Problemas de escalabilidad
  - En un grupo de servidores, podemos sincronizar sesiones entre ellos → mayor cargo.
  - Integración continua: un despliegue de una nueva *release* provoca pérdida de sesiones.

¡SPRING AL RESCATE!
SPRING SESSION

- Integración transparente con *HttpSession*.
- Independencia de la implementación del servidor de aplicaciones.
- Sesiones en cluster
- Múltiples sesiones en un solo navegador
- Uso en APIs RESTful
- ...

2.

SPRING SESSION

SPRING SESSION 2.0

- API de Spring que permite el manejo de sesiones en cluster sin estar atado al contenedor de la aplicación.
- Módulos (entre otros)

Spring Session Core

Funcionalidades principales y comunes.

Spring Session Data Redis

Almacenamiento de sesiones en Redis.

Spring Session Data JDBC

Almacenamiento en una base de datos relacional.

¿QUÉ ES REDIS?

- Motor de base de datos NoSQL en memoria.
- Almacenamiento de hashes (*key/value*)
- Implementada en C. Redis Lab (BSD License)
- Almacenamiento de sesiones, caché, …
- Rendimiento *extremo* comparado con otros SGBD.
- Replicación *master/slave* en árbol.

3.

INTEGRACIÓN DE NUESTRO PROYECTO CON SPRING SESSION

MODIFICACIONES

- Instalar redis (vía docker)
- Actualizar pom.xml
- Configuración (*application.properties*)
- **Spring Boot se encarga de lo demás.**

INSTALAR REDIS (VÍA DOCKER)

- **Docker**: gestor de contenedores
- **Docker hub**: repositorio de imágenes prefabricadas.
- Hay una imagen con la última versión estable de *redis*.

https://docs.docker.com/samples/library/redis/

```
$ docker run -d --name myredis -p 6379:6379 redis
```

ACTUALIZAR POM.XML

- Añadimos dos nuevas dependencias

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>

<dependency>
  <groupId>org.springframework.session</groupId>
  <artifactId>spring-session-data-redis</artifactId>
</dependency>
```

CONFIGURACIÓN

- Gracias a Spring Boot, la configuración se puede hacer vía *application.properties*

  ```
  spring.session.store-type=redis
  ```

- Spring Boot se encarga de crear un filtro, *springSessionRepositoryFilter*, encargado de la gestión de los objetos *HttpSession*.

OTROS ASPECTOS DE CONFIGURACIÓN

- *server.servlet.session.timeout*

  Duración del timeout de una sesión.
- *spring.session.redis.flush-mode=on-save*

  Modo de volcado de datos
- *spring.session.redis.namespace=spring:session*

  Espacio de nombre usado para las claves

OTROS ASPECTOS DE CONFIGURACIÓN

- *spring.redis.host=localhost*
  URL/IP del servidor Redis
- *spring.redis.password*
  Contraseña
- *spring.redis.port=6379*
  Puerto del servidor Redis

¿CÓMO COMPROBAMOS QUE FUNCIONA?

- Ejecutamos la aplicación
- Nos logueamos
- Abrimos una consola y ejecutamos

```
$ docker exec -ti myredis bash
root@......# redis-cli keys '*'
```

```
1) "spring:session:expirations:1542834540000"
2) "spring:session:index:org.springframework.session.FindByIndexNameSessionRepository.PR
INCIPAL_NAME_INDEX_NAME:admin"
3) "spring:session:expirations:1542834600000"
4) "spring:session:sessions:02e57ce5-734d-4983-9a52-7c2efc9b116d"
5) "spring:session:sessions:expires:02e57ce5-734d-4983-9a52-7c2efc9b116d"
```

¿CÓMO COMPROBAMOS QUE FUNCIONA?

- Probamos a borrar la sesión, y comprobamos que nos la aplicación nos vuelve a llevar al login.

```
root@......# redis-cli del
  spring:session:sessions:02e57ce5-734d-....
```

```
1) "spring:session:expirations:1542834540000"
2) "spring:session:index:org.springframework.session.FindByIndexNameSessionRepository.PR
INCIPAL_NAME_INDEX_NAME:admin"
3) "spring:session:expirations:1542834600000"
4) "spring:session:sessions:02e57ce5-734d-4983-9a52-7c2efc9b116d"
5) "spring:session:sessions:expires:02e57ce5-734d-4983-9a52-7c2efc9b116d"
```