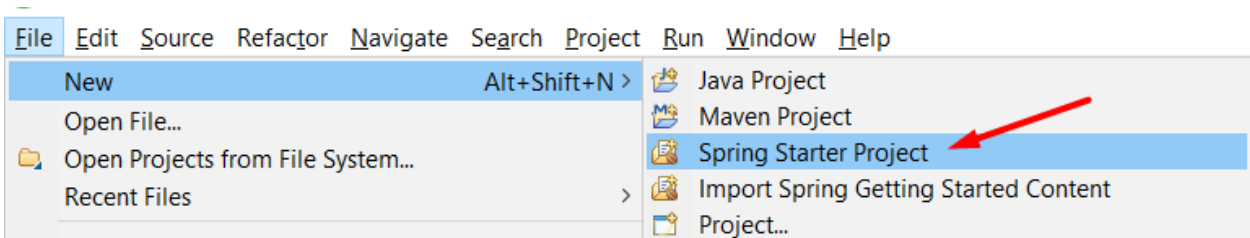


PROYECTO “redis-session-demo”

Paso 1: Crear el Proyecto en STS

Abrir Spring Tool Suite (STS).

Ir a **File → New → Spring Starter Project**.



Completar los datos del proyecto:

- **Name:** redis-session-demo
- **Type:** Maven
- **Java Version:** 17
- **Group:** com.example
- **Artifact:** redis-session-demo
- **Packaging:** Jar

A screenshot of the 'New Spring Starter Project' dialog box in STS. The 'Service URL' is set to 'https://start.spring.io'. The 'Name' field contains 'redis-session-demo'. The 'Use default location' checkbox is checked. The 'Location' field shows 'C:\Users\JR\Documents\workspace\redis-session-demo' with a 'Browse' button. The 'Type' is 'Maven', 'Packaging' is 'Jar', 'Java Version' is '17', and 'Language' is 'Java'. The 'Group' is 'com.example', 'Artifact' is 'redis-session-demo', 'Version' is '0.0.1-SNAPSHOT', 'Description' is 'Demo project for Spring Boot', and 'Package' is 'com.example.redissessiondemo'. There is a 'Working set' field at the bottom.

- **Dependencies:**
 - Lombok
 - Spring Web (Para la API REST).
 - Spring Boot DevTools (Para recarga en vivo).
 - Spring Data Redis (Access+Driver) (Para conectarnos a Redis).

A screenshot of the 'Dependencies' section in the STS dialog. It shows a grid of checkboxes for various dependencies. The 'Lombok' checkbox is checked and highlighted with a dashed box. Other checked dependencies include 'Spring Boot DevTools', 'Spring Data Redis (Access+Driver)', and 'Spring Web'. Other dependencies like 'H2 Database', 'PostgreSQL Driver', 'Spring Cache Abstraction', 'Spring Data JPA', and 'Spring Web Services' are unchecked.

Hacer clic en **Finish** para generar el proyecto.

Paso 2: Configurar application.properties

Abre el archivo **src/main/resources/application.properties** y agrega:



```
# Configuración de Redis
spring.data.redis.host=redis-15022.c82.us-east-1-2.ec2.redns.redis-cloud.com
spring.data.redis.port=15022
spring.data.redis.username=default
spring.data.redis.password=x4M4OjOpwpwr5B331n4NH43FSIRgrwkm
```

Explicación:

- Esto le dice a Spring Boot que use Redis como almacenamiento de sesiones.

Paso 3: Crear la Entidad UserSession

Representa una sesión de usuario almacenada en **Redis**.



```
package com.example.redissessiondemo.model;

import lombok.*;
import org.springframework.data.annotation.Id;
import org.springframework.data.redis.core.RedisHash;
import org.springframework.data.redis.core.TimeToLive;

import java.io.Serializable;

@RedisHash("UserSession") // Almacena en Redis como un Hash
@Getter @Setter
@NoArgsConstructor @AllArgsConstructor

public class UserSession implements Serializable {

    @Id
    private String id;
    private String username;
    private String token;

    @TimeToLive // Define la expiración en segundos
    private Long expiration;
}
```

Explicación:

- **@RedisHash("UserSession")**: Indica que esta clase se almacenará en Redis bajo el nombre UserSession.
- **@Id**: Define el campo clave (key) en Redis (en este caso, el id del usuario).
- **@TimeToLive**: Define la expiración automática en segundos (si no se usa, la sesión es permanente).
- **Serializable**: Necesario para almacenar objetos en Redis.
- **Lombok (@Getter, @Setter, @NoArgsConstructor, @AllArgsConstructor)**: Genera automáticamente getters, setters y constructores.

Paso 4: Crear el Repository `UserSessionRepository`

Permite acceder a las sesiones almacenadas en **Redis**.



```
package com.example.redissessiondemo.repository;

import com.example.redissessiondemo.model.UserSession;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserSessionRepository extends CrudRepository<UserSession, String> {
}
```

Explicación:

- **@Repository**: Marca la interfaz como un repositorio de datos.
- **extends `CrudRepository<UserSession, String>`**: Permite hacer operaciones CRUD sobre las sesiones en Redis.
 - **UserSession**: Tipo de entidad que maneja.
 - **String**: Tipo de la clave en Redis (el id del usuario).

Paso 5: Crear el service `UserSessionService`

Gestiona la creación, consulta y eliminación de sesiones.



```
package com.example.redissessiondemo.service;

import com.example.redissessiondemo.model.UserSession;
import com.example.redissessiondemo.repository.UserSessionRepository;
import org.springframework.stereotype.Service;
import java.util.Optional;
import java.util.UUID;

@Service
public class UserSessionService {

    private final UserSessionRepository userSessionRepository;

    public UserSessionService(UserSessionRepository userSessionRepository) {
        this.userSessionRepository = userSessionRepository;
    }

    public UserSession createSession(String username) {
        String token = UUID.randomUUID().toString();
        minutos UserSession session = new UserSession(username, username, token, 20L); // Expira en 30
        userSessionRepository.save(session);
        return session;
    }

    public Optional<UserSession> getSession(String username) {
        return userSessionRepository.findById(username);
    }

    public void deleteSession(String username) {
        userSessionRepository.deleteByUsername(username);
    }
}
```

Explicación:

- **createSession(String username):**
 - Genera un token único (UUID).
 - Crea una sesión con un TTL de 20 segundos (expiration = 20L).
 - Guarda la sesión en Redis.
 - Devuelve la sesión creada.
- **getSession(String username):**
 - Busca la sesión en Redis usando el id (username).
 - Devuelve un Optional<UserSession> (puede estar vacía si la sesión no existe o ya expiró).
- **deleteSession(String username):**
 - Elimina la sesión en Redis.

Nota:

El campo expiration = 20L hace que la sesión expire automáticamente después de 20 segundos.

Paso 6: Crear el Controlador UserSessionController

Expone los endpoints REST para interactuar con las sesiones.



```
package com.example.redissessiondemo.controller;

import com.example.redissessiondemo.model.UserSession;
import com.example.redissessiondemo.service.UserSessionService;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;

@RestController
@RequestMapping("/session")
public class UserSessionController {

    private final UserSessionService userSessionService;

    public UserSessionController(UserSessionService userSessionService) {
        this.userSessionService = userSessionService;
    }

    @PostMapping("/{username}")
    public ResponseEntity<UserSession> createSession(@PathVariable String username) {
        return ResponseEntity.ok(userSessionService.createSession(username));
    }

    @GetMapping("/{username}")
    public ResponseEntity<Optional<UserSession>> getSession(@PathVariable String username) {
        return ResponseEntity.ok(userSessionService.getSession(username));
    }

    @DeleteMapping("/{username}")
    public ResponseEntity<Void> deleteSession(@PathVariable String username) {
        userSessionService.deleteSession(username);
        return ResponseEntity.noContent().build();
    }
}
```

Paso 8: Ejecutar y Probar

Crea una sesión (POST):

- **POST** <http://localhost:8080/session/johndoe>

Cuyo response o respuesta es lo siguiente:



```
{
  "id": "johndoe",
  "username": "johndoe",
  "token": "57eb137f-9854-4de6-b684-c55667693a56",
  "expiration": 20
}
```

Redis Session API / Session / Create Session

POST {{baseUrl}}/session/johndoe

Params Authorization Headers (9) Body Scripts Tests Settings

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

200 OK • 330 ms • 264 B • Save Response

JSON Preview Visualize

```
1 {
2   "id": "johndoe",
3   "username": "johndoe",
4   "token": "57eb137f-9854-4de6-b684-c55667693a56",
5   "expiration": 20
6 }
```

- GET http://localhost:8080/session/johndoe

GET {{baseUrl}}/session/johndoe

Params Authorization Headers (7) Body Scripts Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

200 OK • 104 ms • 168 B • Save Response

JSON Preview Visualize

```
1 null
```

- DELETE http://localhost:8080/session/johndoe

DELETE {{baseUrl}}/session/johndoe

Params Authorization Headers (7) Body Scripts Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

204 No Content • 110 ms • 112 B • Save Response

Raw Preview Visualize