In a previous post we had implemented Spring Boot Security – Creating a custom login page. Till now we were making use of in memory configuration for authenticating users and associated roles.

In this Use case we will authenticate users and roles against database tables.

1. Continue with ***"boot-security-loginwith-users"***

By default spring security expects tables named **users table** for storing username, passwords and **authorities table** for storing the associated roles. In the schema-mysql.sql add these schemas and insert statements

```
DROP TABLE IF EXISTS employee;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS authorities;


CREATE TABLE employee (
  empId VARCHAR(10) NOT NULL,
  empName VARCHAR(100) NOT NULL
);

create table users (
    username varchar(50) not null primary key,
    password varchar(120) not null,
    enabled boolean not null
);

create table authorities (
    username varchar(50) not null,
    authority varchar(50) not null,
    foreign key (username) references users (username)
);

insert into users(username, password,
enabled)values('admin','admin',true);
```

```
insert into
authorities(username,authority)values('admin','ROLE_ADMIN');

insert into users(username, password,
enabled)values('employee','employee',true);
insert into
authorities(username,authority)values('employee','ROLE_USER');
insert into
authorities(username,authority)values('admin','ROLE_USER');
```

Spring Boot JDBC runs this script before starting the application

2. Create AppConfig.java

```
import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.jdbc.DataSourceBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;

@Configuration
@PropertySource("classpath:application.properties")
public class AppConfig {

  @Autowired
  private Environment env;

  @Bean
  public DataSource getDataSource() {

    DataSourceBuilder dataSourceBuilder = DataSourceBuilder.create();

dataSourceBuilder.driverClassName(env.getProperty("spring.datasource.
driver"));
    dataSourceBuilder.url(env.getProperty("spring.datasource.url"));

dataSourceBuilder.username(env.getProperty("spring.datasource.usernam
e"));

dataSourceBuilder.password(env.getProperty("spring.datasource.passwor
d"));

    return dataSourceBuilder.build();
  }
}
```

3. Finally modify the Spring Security configuration to switch to jdbc authentication.

EmployeeSecurityConfiguration.java

```
@Autowired
     DataSource dataSource;

// Enable JDBC Authentication
     @Autowired
     public void
configAuthentication(AuthenticationManagerBuilder auth) throws
Exception {
          auth.jdbcAuthentication().dataSource(dataSource);
     }
```

4. Comment out **configureGlobal()** method

5. Test by logging in as these users and check that Menu items work depending on the user

admin/admin
employee/employee