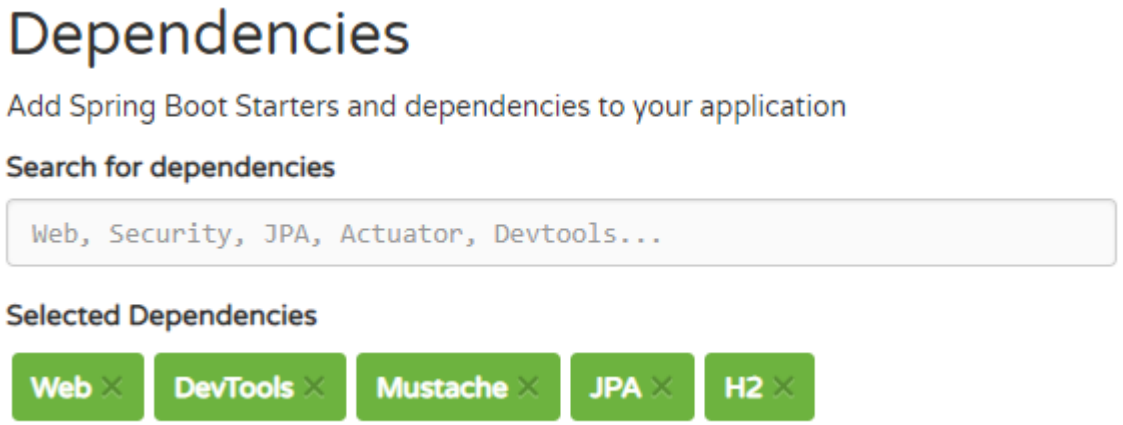
Models

Création du projet

On utilise dans le cadre de cet exemple une base de données embarquée H2Db.

Créer un projet en ajoutant les dépendances suivantes :



Configuration

Modifier le fichier de configuration de l'application Spring pour l'intégration de H2 :

La base de données **dbExemple** est stockée dans le dossier **data**.

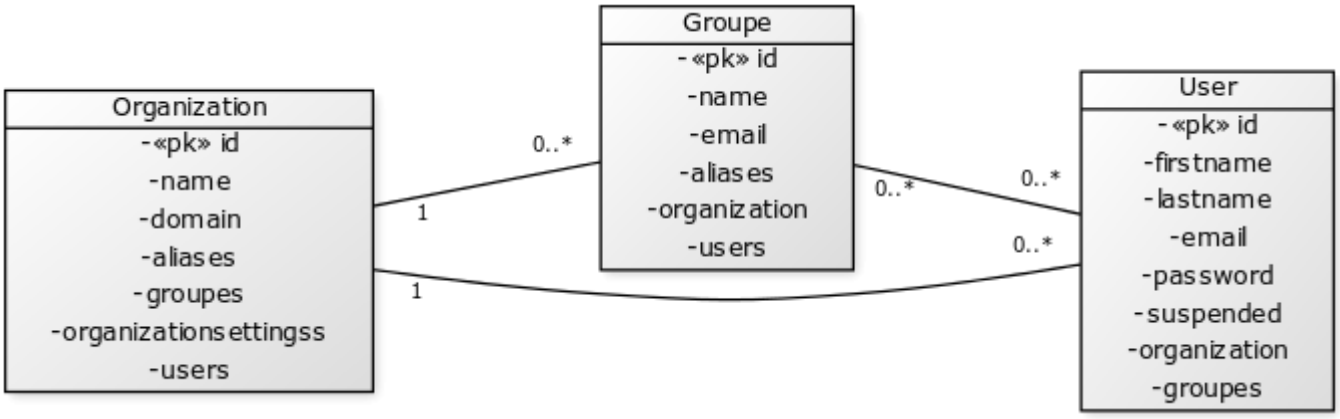
```
spring.datasource.url=jdbc:h2:file:./data/dbExemple;DB_CLOSE_ON_EXIT=FALSE
spring.datasource.username=sa
spring.datasource.password=
spring.datasource.driverClassName=org.h2.Driver
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true
spring.h2.console.path=/h2-console

...
```

Classes

Soit le diagramme de classes suivant, correspondant à une application de gestion de messagerie :



Création d'une Entité

```
package s4.spring.td2.models;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Organization {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private int id;
    private String name;
    private String domain;
    private String aliases;
}
```

Annotations :

- @Entity
- @Id
- @GeneratedValue

Création d'un Repository

```
public interface OrgaRepository extends JpaRepository<Organization, Integer> {
    List<Organization> findByDomain(String domain);
    List<Organization> findByName(String name);
}
```

JPA Repositories

Test d'ajout d'une instance

Les données sont postées depuis un formulaire contenant les champs name, domain et aliases :

```
@Controller
@RequestMapping("/orgas/")
public class OrgasController {
    @Autowired
    private OrgaRepository repo;
    @PostMapping("new")
    @ResponseBody
    public String newOrga(Organization orga) {
        repo.saveAndFlush(orga);
        return orga+" ajoutée.";
    }
}
```

Relations

ManyToOne

Chaque groupe appartient à une organisation.

```
@Entity
public class Groupe {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private int id;
    @ManyToOne
    private Organization organization;
}
```

OneToMany

Chaque organisation possède plusieurs groupes. (la relation est bi-directionnelle dans ce cas)

```
@Entity
public class Organization {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private int id;
    private String name;
    private String domain;
    private String aliases;
    @OneToMany(cascade=CascadeType.ALL,mappedBy="organization")
    private List<Groupe> groupes;
}
```

ManyToMany

Chaque utilisateur peut appartenir à plusieurs groupes. Dans chaque groupe, on a plusieurs utilisateurs.

```
@Entity
public class User {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private int id;
    @ManyToOne
    private Organization organization;
    @ManyToMany(mappedBy="users")
    private List<Groupe> groupes;
}
```

```
@Entity
public class Groupe{
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private int id;
    @ManyToOne
    private Organization organization;
    @ManyToMany
    @JoinTable(name = "user_group")
    private List<User> users;
}
```

Logs

Pour tracer les logs Hibernate et visualiser les requêtes, activez les options suivantes dans application.properties :

```
#spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
# logs the SQL statements
# basic log level for all messages
logging.level.org.hibernate=info
# SQL statements and parameters
logging.level.org.hibernate.SQL=debug
logging.level.org.hibernate.orm.jdbc.bind=trace
```