01-2025

code
URJC

Desarrollo Web

Tecnologías de servidor web

Tema 2.3: Generación de tablas desde entidades

Universidad Rey Juan Carlos

Micael Gallego, Francisco Gortázar, Michel Maes, Óscar Soto e Iván Chicano

code
URJC

©2025

Micael Gallego, Francisco Gortázar, Michel Maes, Óscar Soto, Iván Chicano

Algunos derechos reservados

Este documento se distribuye bajo la licencia
"Atribución-CompartirIgual 4.0 Internacional"
de Creative Comons Disponible en
https://creativecommons.org/licenses/by-sa/4.0/deed.es

Generación de tablas desde entidades

code URJC

ejem2

- **Entidad con atributos básicos**
  - Se genera una tabla por cada entidad
  - Por cada atributo de la clase de un tipo simple (entero, float, String, boolean…), se crea un campo en la tabla

```
@Entity
public class Post {

  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private long id;

  private String username;
  private String title;
  private String text;
}
```

```
create table post (
  id bigint not null,
  text varchar(255),
  title varchar(255),
  username varchar(255),
  primary key (id)
)
```

Generación de tablas desde entidades

code URJC

ejem2

- **Entidad con atributos básicos**
  - Si el nombre de una entidad o uno de sus atributos es una **palabra reservada en SQL** para esa base de datos, se obtendrá un **ERROR al ejecutar la sentencia**

```java
@Entity
public class Post {

  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private long id;

  private String user;
  private String title;
  private String text;
}
```

```
create table post (
  id bigint not null,
  text varchar(255),
  title varchar(255),
  user varchar(255),
  primary key (id)
)
```

ERROR!!!

Generación de tablas desde entidades

- **Palabras reservadas SQL en entidades**
  - **1)** Se puede cambiar el nombre de la tabla o atributo por uno que no sea palabra reservada con

```
@Table(name="nombreTabla")
```

```
@Column(name="nombreCampo")
```

2) Se puede escapar el nombre de la tabla o atributo

```
@Table(name="\"nombreTabla\"")
```

```
@Column(name="\"nombreCampo\"")
```

  - **3)** Se puede pedir a Hibernate que escape TODOS los nombres de tablas y campos en las sentencias SQL

```
spring.jpa.properties.hibernate.globally_quoted_identifiers=true
spring.jpa.properties.hibernate.globally_quoted_identifiers_skip_column_definitions=true
```

https://vladmihalcea.com/escape-sql-reserved-keywords-jpa-hibernate/

Generación de tablas desde entidades

- **Relaciones entre entidades**
  - Existe una relación cuando una entidad tiene como **atributo otra entidad**
  - En memoria, un **objeto** está asociado a uno o más **objetos**
  - También es posible tener una colección de entidades (**List, Set, Map…**)
  - Las relaciones pueden ser **1:1, 1:N, M:N**

Generación de tablas desde entidades

- **Relaciones entre entidades**
  - **1:1**
    - Unidireccional
    - Bidireccional
  - **1:N**
    - Unidireccional (1 → N y N → 1)
    - Bidireccional
  - **N:M**
    - Bidireccional

Generación de tablas desde entidades

code URJC

ejem3

- **Relación 1:1 unidireccional**
  - Una entidad tiene una referencia a otra entidad
  - El atributo que apunta a otro es anotado con **@OneToOne**
  - Cada objeto tiene que ser guardado en la base de datos de forma **independiente**
  - Al **cargar un objeto** de la BBDD podemos acceder al objeto **relacionado sin cargarlo explícitamente**

Generación de tablas desde entidades

code URJC

ejem3

- Relación 1:1 unidireccional

```
@Entity
public class Student {

  @Id
  @GeneratedValue(...)
  private long id;

  private String name;
  private int year;

  @OneToOne
  private Project project;

}
```

```
@Entity
public class Project {

  @Id
  @GeneratedValue(...)
  private long id;

  private String title;
  private int calification;
}
```

Generación de tablas desde entidades

code URJC

ejem3

- Relación 1:1 unidireccional

```
@Entity
public class Student {

  @Id
  @GeneratedValue(...)
  private long id;

  private String name;
  private int year;

  @OneToOne
  private Project project;

}
```

```
@Entity
public class Project {

  @Id
  @GeneratedValue(...)
  private long id;

  private String title;
  private int calification;
}
```

Relación 1:1 unidireccional

code URJC

ejem3

```java
@Entity
public class Student {

  @Id
  @GeneratedValue(...)
  private long id;

  private String name;
  private int year;

  @OneToOne
  private Project project;
}
```

```java
@Entity
public class Project {

  @Id
  @GeneratedValue(...)
  private long id;

  private String title;
  private int calification;
}
```

Generación de tablas

```
create table project (
  id bigint not null,
  calification integer not null,
  title varchar(255),
  primary key (id)
)

create table student (
  id bigint not null,
  name varchar(255),
  year integer not null,
  project_id bigint,
  primary key (id)
)

alter table student
  add constraint Fkr...
  foreign key (project_id)
  references project
```

Generación de tablas desde entidades

code URJC

ejem3

- Relación 1:1 unidireccional

```
Project p1 = new Project("TFG1", 8);
projectRepository.save(p1);

Student s1 = new Student("Pepe", 2010);
s1.setProject(p1);

Student s2 = new Student("Juan", 2011);

studentRepository.save(s1);
studentRepository.save(s2);
```

Cada objeto se tiene que guardar en su repositorio

Inicialización de datos

Generación de tablas desde entidades

code URJC

ejem3

- Relación 1:1 unidireccional

Consulta de datos

```
studentRepository.findAll();
```

```
[
    Student [
            id=1,
            name=Pepe,
            startYear=2010,
            project=Project [
                id=1,
                title=TFG1,
                calification=8]
            ],
    Student [
            id=2,
            name=Juan,
            startYear=2011,
            project=null
            ]
]
```

Generación de tablas desde entidades

code URJC

ejem3

- **Relación 1:1 unidireccional**
  - Si borramos un estudiante, su proyecto no se borra

```
@PostMapping("/students/{id}/delete")
public String deleteStudent(@PathVariable Long id, Model model) {
    studentRepository.deleteById(id);
    return "deleted_student";
}
```

Generación de tablas desde entidades

code URJC

ejem3

- **Relación 1:1 unidireccional**
  - Para borrar un proyecto, es necesario deasignarlo del estudiante

```
@PostMapping("/students/{studentId}/project/delete")
public String deleteProject(Model model, @PathVariable Long studentId) {

    Student student = studentRepository.findById(studentId).get();
    Project project = student.getProject();
    student.setProject(null);
    studentRepository.save(student);
    projectRepository.delete(project);
    model.addAttribute("student", student);
    return "show_student";
}
```

Generación de tablas desde entidades

code URJC

ejem4

- **Operaciones en cascada**
  - Hay veces que un objeto **siempre está asociado** a otro objeto (relación de composición)
  - Por ejemplo, un proyecto siempre está asociado a un estudiante
  - Cuando se **crea el estudiante**, **se crea el proyecto**, cuando se borra el estudiante, se borra el proyecto

Generación de tablas desde entidades

code URJC

ejem4

- **Operaciones en cascada**
  - Si la anotación @OneToOne se configura con **cascade = CascadeType.ALL** entonces ambos objetos de la relación tienen el mismo ciclo de vida
  - Al **guardar** el objeto principal, se **guarda** el asociado
  - Al **borrar** el objeto principal, se **borra** el asociado

Generación de tablas desde entidades

code URJC

ejem4

- Operaciones en cascada

```
@Entity
public class Student {

    @Id
    @GeneratedValue(...)
    private long id;

    private String name;
    private int year;

    @OneToOne(cascade=CascadeType.ALL)
    private Project project;
}
```

Como la relación en **cascade=ALL**, no es necesario guardar el objeto project explícitamente.

El **project** se guarda cuando el **student** se guarda

```
Student s1 = new Student("Pepe", 2010);
s1.setProject(new Project("TFG1", 8));
studentRepository.save(s1);
```

Generación de tablas desde entidades

code URJC

ejem4

- **Operaciones en cascada**
  - Se borrará el proyecto asociado

```
@PostMapping("/students/{id}/delete")
public String deleteStudent(@PathVariable Long id, Model model) {
    studentRepository.deleteById(id);
    return "deleted_student";
}
```

Generación de tablas desde entidades

code URJC

ejem5

- **Relación 1:1 bidireccional**
  - Para facilitar el desarrollo, los **dos objetos de una relación** pueden tener un atributo apuntando al otro objeto
  - Ambas entidades tienen que anotarse con **@OneToOne**
  - Aunque en memoria ambos objetos se apunten entre sí, en la base de datos la relación se guarda en la **tabla de una entidad**

Generación de tablas desde entidades

code URJC

ejem5

- **Relación 1:1 bidireccional**
  - La entidad principal cuya tabla guarda la información anota el atributo con **@OneToOne**
  - El atributo debe apuntar al otro objeto de la otra entidad cuando se **guardar en la BBDD**
  - La otra entidad se anota con **@OneToOne(mappedby="attr")** indicando el nombre del atributo de la otra principal
  - Este atributo sólo se usa en lecturas

Generación de tablas desde entidades

ejem5

- Relación 1:1 bidireccional

```java
@Entity
public class Student {

  @Id
  @GeneratedValue(...)
  private long id;

  private String name;
  private int year;

  @OneToOne(cascade=CascadeType.ALL)
  private Project project;
}
```

```java
@Entity
public class Project {

  @Id
  @GeneratedValue(...)
  private long id;

  private String title;
  private int calification;

  @OneToOne(mappedBy="project")
  private Student student;

}
```

```java
Student s1 = new Student("Pepe", 2010);
s1.setProject(new Project("TFG1", 8));
studentRepository.save(s1);
```

Al guardar, la entidad principal debe apuntar a la otra entidad

Generación de tablas desde entidades

code URJC

ejem5

- Relación 1:1 bidireccional

```
create table project (
   id bigint not null,
   calification integer not null,
   title varchar(255),
   primary key (id)
)

create table student (
   id bigint not null,
   name varchar(255),
   year integer not null,
   project_id bigint,
   primary key (id)
)

alter table student
   add constraint FKr6av6arpoy7wpbqipg41id1mn
   foreign key (project_id)
   references project
```

Como la entidad principal sigue siendo Student, las tablas son las mismas que con relación unidireccional

Generación de tablas desde entidades

- **Relaciones entre entidades**
  - **1:1**
    - Unidireccional
    - Bidireccional
  - **1:N**
    - Unidireccional (1 → N y N → 1)
    - Bidireccional
  - **N:M**
    - Bidireccional

Generación de tablas desde entidades

code URJC

ejem6

- **Relación 1:N**
  - Cuando existe una relación **1:N** entre entidades se usan las anotaciones **@OneToMany** y **@ManyToOne**
  - Puede ser **unidireccional** o **bidireccional**
  - Cualquier sentido de la relación puede ser **la entidad principal** (la que se usa para guardar los datos)
  - También se puede usar **cascade**

Generación de tablas desde entidades

code URJC

ejem6

- Relación 1:N unidireccional

```
@Entity
public class Team {

  @Id
  @GeneratedValue(...)
  private long id;

  private String name;
  private int ranking;

  @OneToMany
  private List<Player> players;

}
```

```
@Entity
public class Player {

  @Id
  @GeneratedValue(...)
  private long id;

  private String name;
  private int goals;

}
```

Generación de tablas desde entidades

ejem6

- Relación 1:N unidireccional

```
create table player (
   id bigint not null,
   goals integer not null,
   name varchar(255),
   primary key (id)
)

create table team (
   id bigint not null,
   name varchar(255),
   ranking integer not null,
   primary key (id)
)

create table team_players (
   team_id bigint not null,
   players_id bigint not null
)
```

Como es @OneToMany, un player sólo puede estar asociado a un equipo

```
alter table team_players
   add constraint UK... unique (players_id)

alter table team_players
   add constraint FK...
   foreign key (players_id)
   references player

alter table team_players
   add constraint FK...
   foreign key (team_id)
   references team
```

Tabla que relaciona Teams y Players

Generación de tablas desde entidades

code URJC

ejem6

- Relación 1:N unidireccional

```
Player p1 = new Player("Torres", 10);
Player p2 = new Player("Iniesta", 10);
Player p3 = new Player("Messi", 20);

playerRepository.save(p1);
playerRepository.save(p2);
playerRepository.save(p3);

Team team = new Team("Selección Española", 1);

team.getPlayers().add(p1);
team.getPlayers().add(p2);

teamRepository.save(team);
```

Generación de tablas desde entidades

code URJC

ejem6

- **Relación 1:N unidireccional**
  - Sin cascade los objetos tienen ciclos de vida independientes (Un player puede estar sin team)

```
// Deleting a team doesn't delete its associated players
@PostMapping("/teams/{id}/delete")
public String deleteTeam(@PathVariable Long id) {
    Optional<Team> teamOptional = teamRepository.findById(id);
    if(teamOptional.isEmpty()) {
        return "team_not_found";
    }else{
        teamRepository.deleteById(id);
        return "deleted_team";
    }
}
```

Generación de tablas desde entidades

code URJC

ejem6

- **Relación 1:N unidireccional**
  - Un jugador no podrá ser borrado si pertenece a un equipo (Spring devuelve una excepción)

```
// A player only can be deleted if it has no associated team
@PostMapping("/players/{id}/delete")
public String deleteProject(@PathVariable Long id) {
     Optional<Player> playerOptional = playerRepository.findById(id);
     if(playerOptional.isEmpty()) {
          return "player_not_found";
     }else{
          try{
               playerRepository.deleteById(id);
               return "deleted_player";
          }
          catch(DataIntegrityViolationException e){
               return "player_has_team";
          }
     }
}
```

Generación de tablas desde entidades

code URJC

ejem7

- Relación 1:N unidireccional (cascade)

```
@Entity
public class Post {

  @Id
  @GeneratedValue(...)
  private long id;

  private String title;
  private String text;

  @OneToMany(cascade=CascadeType.ALL)
  private List<Comment> comments;
}
```

```
@Entity
public class Comment {

  @Id
  @GeneratedValue(...)
  private long id;

  private String author;
  private String message;
}
```

```
Post post = new Post("Spring Post", "Spring is cool");
post.getComments().add(new Comment("Pepe", "Cool!"));
post.getComments().add(new Comment("Juan", "Very cool"));
postRepository.save(post);
```

Generación de tablas desde entidades

code URJC

ejem7

- **Relación 1:N unidireccional (cascade)**
  - Al borrar un post, se borrarán todos sus comentarios

```
// Deleting a post delete its associated comments
@PostMapping("/posts/{id}/delete")
public String deletePost(@PathVariable long id) {
    Optional<Post> post = postRepository.findById(id);
    if (post.isPresent()) {
        postRepository.deleteById(id);
        return "redirect:/";
    } else {
        return "post_not_found";
    }
}
```

Generación de tablas desde entidades

code URJC

ejem8

- Relación 1:N bidireccional

```
@Entity
public class Team {

  @Id
  @GeneratedValue(...)
  private long id;

  private String name;
  private int ranking;

  @OneToMany(mappedBy="team")
  private List<Player> players;

}
```

```
@Entity
public class Player {

  @Id
  @GeneratedValue(...)
  private long id;

  private String name;
  private int goals;

  @ManyToOne
  private Team team;
}
```

Generación de tablas desde entidades

code URJC

ejem8

- Relación 1:N bidireccional

```
create table player (
  id bigint not null,
  goals integer not null,
  name varchar(255),
  team_id bigint,
  primary key (id)
)

create table team (
  id bigint not null,
  name varchar(255),
  ranking integer not null,
  primary key (id)
)

alter table player
  add constraint FK...
  foreign key (team_id)
  references team
```

Ahora la entidad principal es **Player**, por eso tiene el atributo **team_id** y no hay tabla de relación

Generación de tablas desde entidades

code URJC

ejem8

- Relación 1:N bidireccional

```
Team team = new Team("Selección Española", 1);

teamRepository.save(team);

Player p1 = new Player("Torres", 10);
Player p2 = new Player("Iniesta", 10);
Player p3 = new Player("Messi", 20);

p1.setTeam(team);
p2.setTeam(team);

playerRepository.save(p1);
playerRepository.save(p2);
playerRepository.save(p3);
```

La **entidad principal** ahora es **Player,** por eso se configura el Team en el objeto Player antes de guardar (y no al revés)

Generación de tablas desde entidades

code URJC

ejem8

- **Relación 1:N bidireccional**
  - Ahora, cuandose borra un equipo, habrá que deshacer la relación o el borrado fallará

```
// Deleting a team doesn't delete its associated players
@PostMapping("/teams/{id}/delete")
public String deleteTeam(@PathVariable Long id) {
     Optional<Team> teamOptional = teamRepository.findById(id);
     if(teamOptional.isEmpty()) {
          return "team_not_found";
     }else{
          Team team = teamOptional.get();
          team.getPlayers().forEach(player -> player.setTeam(null));
          teamRepository.deleteById(id);
          return "deleted_team";
     }
}
```

Generación de tablas desde entidades

code URJC

ejem8

- **Relación 1:N bidireccional**
  - Ahora, un jugador si podrá ser borrado incluso si pertenece a un equipo

```
// A player with a team now can be deleted
@PostMapping("/players/{id}/delete")
public String deleteProject(@PathVariable Long id) {
     Optional<Player> playerOptional = playerRepository.findById(id);
     if(playerOptional.isEmpty()) {
          return "player_not_found";
     }else{
          playerRepository.deleteById(id);
          return "deleted_player";
     }
}
```

Generación de tablas desde entidades

code URJC

ejem9

- **Relación 1:N bidireccional (cascade)**
  - Cuando se configura una relación de composición en la BBDD se suele configurar **cascade** y **orphanRemoval**
  - Se suele configurar **cascade=ALL** para que no haya que guardar los objetos de forma independiente
  - Se suele configurar **orphanRemoval=true** para que no pueda haber una parte sin si todo

https://vladmihalcea.com/orphanremoval-jpa-hibernate/

Generación de tablas desde entidades

code URJC

ejem9

- Relación 1:N bidireccional (cascade)

```
@Entity
public class Post {

      @Id
      @GeneratedValue(...)
      private long id;

      private String title;
      private String text;

      @OneToMany(mappedBy="post", cascade=CascadeType.ALL, orphanRemoval=true)
      private List<Comment> comments = new ArrayList<>();

      public void addComment(Comment comment) {
         comments.add(comment);
         comment.setPost(this);
    }

    public void removeComment(Comment comment) {
         comments.remove(comment);
         comment.setPost(null);
    }
}
```

Mantiene los atributos sincronizados en ambos extremos (Post y Comment)

Generación de tablas desde entidades

code URJC

ejem9

- Relación 1:N bidireccional (cascade)

```
@Entity
public class Comment {

    @Id
    @GeneratedValue(...)
    private long id;

    private String author;
    private String message;

    @ManyToOne
    private Post post;

}
```

Generación de tablas desde entidades

code URJC

ejem9

- Relación 1:N bidireccional (cascade)

```
create table comment (
  id bigint not null,
  message varchar(255),
  author varchar(255),
  post_id bigint,
  primary key (id)
)

create table post (
  id bigint not null,
  text varchar(255),
  title varchar(255),
  primary key (id)
)

alter table comment
  add constraint FK..
  foreign key (post_id)
  references post
```

Generación de tablas desde entidades

code URJC

ejem9

- **Relación 1:N bidireccional (cascade)**
  - Al actualizar un post, es necesario mantener la relación

```
@PostMapping("/posts/{id}/update")
public String updatePost(Model model, Post updatedPost, @PathVariable long id) {

    Post oldPost = postRepository.findById(id).orElseThrow();
    updatedPost.setId(id);

    // We assume that comments are not updated with edit operation
    oldPost.getComments().forEach(c -> updatedPost.addComment(c));

    postRepository.save(updatedPost);

    return "redirect:/posts/" + id;
}
```

Generación de tablas desde entidades

- **Relaciones entre entidades**
  - **1:1**
    - Unidireccional
    - Bidireccional
  - **1:N**
    - Unidireccional (1 → N y N → 1)
    - Bidireccional
  - **N:M**
    - Bidireccional

Generación de tablas desde entidades

code URJC

ejem10

- **Relación M:N**
  - Cuando existe una relación M:N entre entidades se usa la anotación **@ManyToMany**
  - Puede ser **unidireccional** o **bidireccional**
  - Cualquier sentido de la relación puede ser **la entidad principal** (la que se usa para guardar los datos)
  - También se puede usar **cascade**

Generación de tablas desde entidades

code URJC

ejem10

- Relación M:N bidireccional

```
@Entity
public class Team {

  @Id
  @GeneratedValue(...)
  private long id;

  private String name;
  private int ranking;

  @ManyToMany(mappedBy="teams")
  private List<Player> players;

}
```

```
@Entity
public class Player {

  @Id
  @GeneratedValue(...)
  private long id;

  private String name;
  private int goals;

  @ManyToMany
  private List<Team> teams;

}
```

Generación de tablas desde entidades

code URJC

ejem10

- Relación M:N bidireccional

```
create table player (
   id bigint not null,
   goals integer not null,
   name varchar(255),
   primary key (id)
)

create table team (
   id bigint not null,
   name varchar(255),
   ranking integer not null,
   primary key (id)
)

create table player_teams (
   players_id bigint not null,
   teams_id bigint not null
)
```

Como es @ManyToMany ya no está la restricción de que un player sólo pueda estar asociado a un team

```
alter table player_teams
   add constraint FK...
   foreign key (teams_id)
   references team

alter table player_teams
   add constraint FK...
   foreign key (players_id)
   references player
```

Generación de tablas desde entidades

code URJC

ejem10

- Relación M:N bidireccional

```
Team team1 = new Team("Selección", 1);
Team team2 = new Team("FC Barcelona", 1);
Team team3 = new Team("Atlético de Madrid", 2);

teamRepository.save(team1);
teamRepository.save(team2);
teamRepository.save(team3);

Player p1 = new Player("Torres", 10);
Player p2 = new Player("Iniesta", 10);

p1.getTeams().add(team1);
p1.getTeams().add(team3);

p2.getTeams().add(team1);
p2.getTeams().add(team2);

playerRepository.save(p1);
playerRepository.save(p2);
```

Generación de tablas desde entidades

- **Herencia entre clases**
  - Existen varias estrategias