

Mobile Application Development

Higher Diploma in Science in Computer Science

Produced
by

Eamonn de Leastar (edelestar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

<http://www.wit.ie>

<http://elearning.wit.ie>

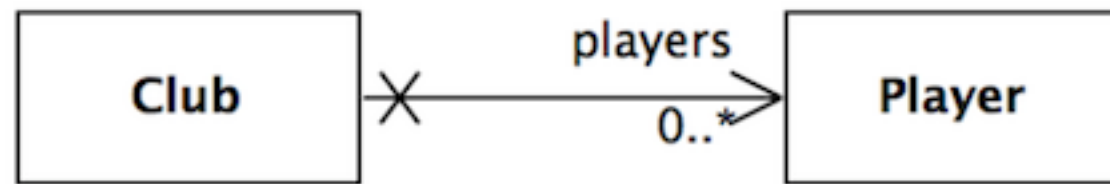


Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



Modeling & JPA 2

OneToMany



OneToMany - Unidirectional

```
public class Club extends Model
{
    public String name;

    @OneToMany(cascade=CascadeType.ALL)
    public List<Player> players;

    public Club(String name)
    {
        this.name = name;
        this.players = new ArrayList<Player>();
    }

    public String toString()
    {
        return name;
    }

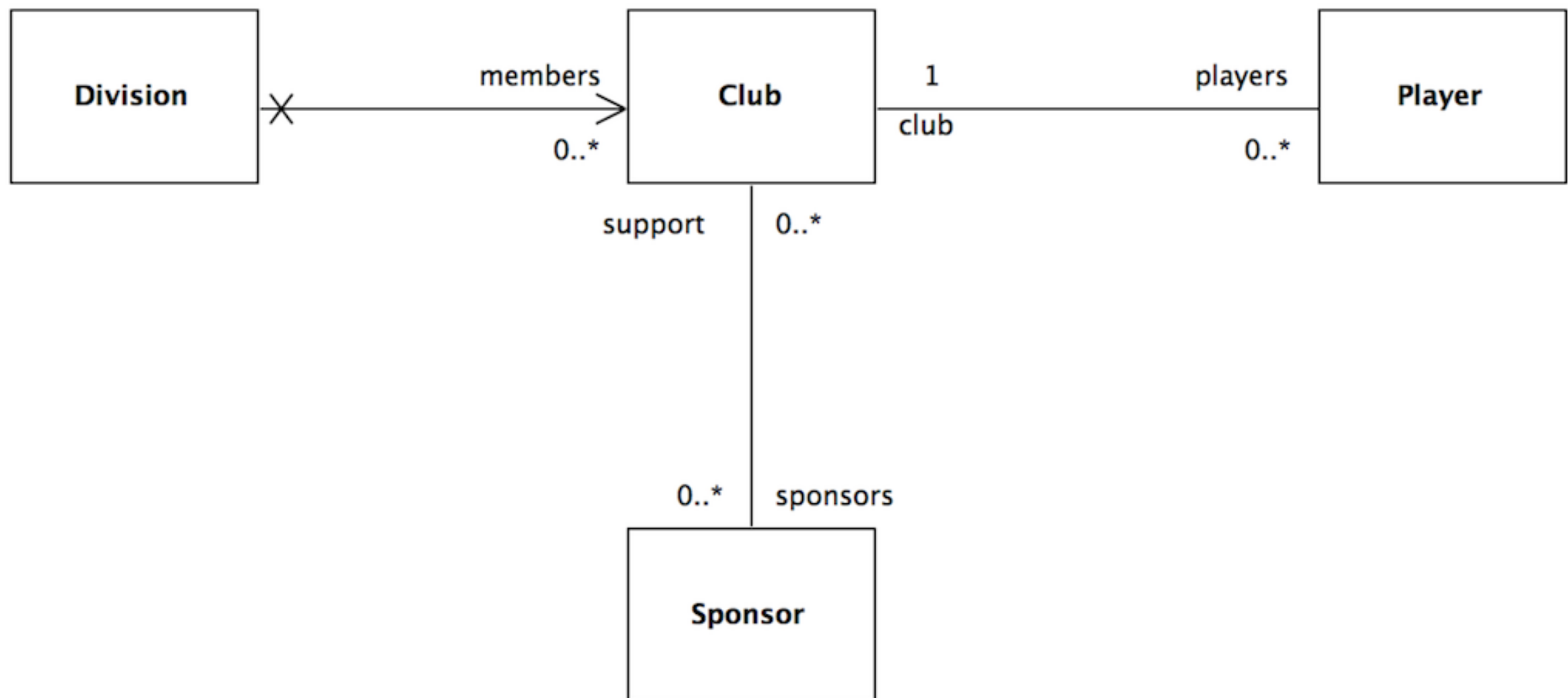
    public void addPlayer(Player player)
    {
        players.add(player);
    }
}
```

```
public class Player extends Model
{
    public String name;

    public Player(String name)
    {
        this.name = name;
    }

    public String toString()
    {
        return name;
    }
}
```

OneToMany, ManyToOne, ManyToMany



OneToMany

```
public class Division extends Model
{
    public String name;

    @OneToMany(cascade=CascadeType.ALL)
    public List<Club> members;

    public Division(String name)
    {
        this.name = name;
        members = new ArrayList<Club>();
    }

    public void addClub(Club club)
    {
        members.add(club);
    }

    public String toString()
    {
        return name;
    }

    public static Division findByName(String name)
    {
        return find("name", name).first();
    }
}
```

```
public class Club extends Model
{
    public String name;

    @OneToMany(mappedBy="club", cascade=CascadeType.ALL)
    public List<Player> players;

    @ManyToMany
    public List<Sponsor> sponsors;

    public Club(String name)
    {
        this.name = name;
        this.players = new ArrayList<Player>();
        this.sponsors = new ArrayList<Sponsor>();
    }

    public String toString()
    {
        return name;
    }

    public static Club findByName(String name)
    {
        return find("name", name).first();
    }

    public void addPlayer(Player player)
    {
        player.club = this;
        players.add(player);
    }

    public void addSponsor(Sponsor company)
    {
        sponsors.add(company);
    }

    public void removePlayer(Player player)
    {
        players.remove(player);
    }
}
```

ManyToOne

```
public class Club extends Model
{
    public String name;

    @OneToMany(mappedBy="club", cascade=CascadeType.ALL)
    public List<Player> players;

    //..
}
```

```
public class Player extends Model
{
    public String name;

    @ManyToOne
    public Club club;

    public Player(String name)
    {
        this.name = name;
    }

    public String toString()
    {
        return name;
    }

    public static Player findByName(String name)
    {
        return find("name", name).first();
    }
}
```

ManyToMany

```
public class Sponsor extends Model
{
    public String name;

    @ManyToMany (mappedBy="sponsors")
    public List<Club> support;

    public Sponsor(String name)
    {
        this.name = name;
        support = new ArrayList<Club>();
    }

    public void addSuport(Club club)
    {
        support.add(club);
    }

    public String toString()
    {
        return name;
    }
}
```

```
public class Club extends Model
{
    public String name;

    @OneToMany(mappedBy="club", cascade=CascadeType.ALL)
    public List<Player> players;

    @ManyToMany
    public List<Sponsor> sponsors;

    public Club(String name)
    {
        this.name = name;
        this.players = new ArrayList<Player>();
        this.sponsors = new ArrayList<Sponsor>();
    }

    public String toString()
    {
        return name;
    }

    public static Club findByName(String name)
    {
        return find("name", name).first();
    }

    public void addPlayer(Player player)
    {
        player.club = this;
        players.add(player);
    }

    public void addSponsor(Sponsor company)
    {
        sponsors.add(company);
    }

    public void removePlayer(Player player)
    {
        players.remove(player);
    }
}
```


Tests

- For more complex models, create fixtures in data.yml.
- These models can be loaded in unit tests

```
Club(dunmore):  
  name: dunmore  
  
Club(tramore):  
  name: tramore  
  
Club(fenor):  
  name: fenor  
  
Player(jim):  
  name: jim  
  club: dunmore  
  
Player(mary):  
  name: mary  
  club: dunmore  
  
Player(sam):  
  name: sam  
  club: tramore  
  
Player(john):  
  name: john  
  club: tramore  
  
Player(mike):  
  name: mike  
  club: fenor  
  
Player(linda):  
  name: john  
  club: fenor  
  
Division(senior):  
  name: senior  
  members:  
    - tramore  
    - dunmore  
  
Division(junior):  
  name: junior  
  members:  
    - fenor  
  
Sponsor(newsagent):  
  name: newsagent  
  
Sponsor(pub):  
  name: pub
```

data.yml

data.yml

```
Club(dunmore):
  name: dunmore

Club(tramore):
  name: tramore

Club(fenor):
  name: fenor

Player(jim):
  name: jim
  club: dunmore

Player(mary):
  name: mary
  club: dunmore

Player(sam):
  name: sam
  club: tramore

Player(john):
  name: john
  club: tramore

Player(mike):
  name: mike
  club: fenor

Player(linda):
  name: john
  club: fenor

Division(senior):
  name: senior
  members:
    - tramore
    - dunmore

Division(junior):
  name: junior
  members:
    - fenor

Sponsor(newsagent):
  name: newsagent

Sponsor(pub):
  name: pub
```

ComprehensiveTest

```
public class ComprehensiveTest extends UnitTest
{
    @Before
    public void setup()
    {
        Fixtures.loadModels("data.yml");
    }

    @After
    public void teardown()
    {
        Fixtures.deleteAllModels();
    }
}
```

Forward References

- In yaml files, representing many-to-many relationships cannot be easily represented.
- e.g:
 - dunmore->newsagent
 - newsagent->dunmore

```
Club(dunmore):  
  name: dunmore  
  
Player(jim):  
  name: jim  
  club: dunmore  
  
Player(mary):  
  name: mary  
  club: dunmore  
  
Division(junior):  
  name: junior  
  members:  
    - dunmore  
  
Sponsor(newsagent):  
  name: newsagent
```

Forward References - Workaround

- Load the data.yaml model without ManyToMany
- Establish the relationship after the fixture is loaded

```
public class ComprehensiveTest extends UnitTest
{
    public static void loadSponsorships()
    {
        Club    tramore    = Club.find("byName", "tramore").first();
        Club    dunmore    = Club.find("byName", "dunmore").first();
        Sponsor newsagent = Sponsor.find("byName", "newsagent").first();

        tramore.addSponsor(newsagent);
        dunmore.addSponsor(newsagent);

        newsagent.addSupport(tramore);
        newsagent.addSupport(dunmore);

        tramore.save();
        dunmore.save();
        newsagent.save();
    }

    @Before
    public void setup()
    {
        Fixtures.loadModels("data.yaml");
        loadSponsorships();
    }
}
```

Test Strategy

- For each relationship:
 - ‘short’ test - quick sanity check
 - ‘long’ test - full exercise of relationship, in both directions if present
 - ‘edit’ test - perform change on objects

```

Club(dunmore):
  name: dunmore

Club(tramore):
  name: tramore

Club(fenor):
  name: fenor

Player(jim):
  name: jim
  club: dunmore

Player(mary):
  name: mary
  club: dunmore

Player(sam):
  name: sam
  club: tramore

Player(john):
  name: john
  club: tramore

Player(mike):
  name: mike
  club: fenor

Player(linda):
  name: john
  club: fenor

Division(senior):
  name: senior
  members:
    - tramore
    - dunmore

Division(junior):
  name: junior
  members:
    - fenor

Sponsor(newsagent):
  name: newsagent

Sponsor(pub):
  name: pub

```

Test Data

```

public class ComprehensiveTest extends UnitTest
{
    public static void loadSponsorships()
    {
        Club    tramore    = Club.find("byName", "tramore").first();
        Club    dunmore    = Club.find("byName", "dunmore").first();
        Sponsor newsagent = Sponsor.find("byName", "newsagent").first();

        tramore.addSponsor(newsagent);
        dunmore.addSponsor(newsagent);

        newsagent.addSuport(tramore);
        newsagent.addSuport(dunmore);

        tramore.save();
        dunmore.save();
        newsagent.save();
    }

    @Before
    public void setup()
    {
        Fixtures.loadModels("data.yml");
        loadSponsorships();
    }
}

```

'Sanity' Tests

```
@Test
public void testPlayerClub()
{
    Club    dunmore = Club.find("byName", "dunmore").first();
    Player jim      = Player.find("byName", "jim").first();
    Player mary     = Player.find("byName", "mary").first();
    assertNotNull(mary);

    assertTrue (dunmore.players.contains(jim));
    assertTrue (dunmore.players.contains(mary));
}

@Test
public void testDivisionClub()
{
    Division senior = Division.find("byName", "senior").first();
    Club    dunmore = Club.find("byName", "dunmore").first();
    Club    tramore = Club.find("byName", "tramore").first();

    assertTrue (senior.members.contains(dunmore));
    assertTrue (senior.members.contains(tramore));
}

@Test
public void testClubSponsorShort()
{
    Sponsor newsagent = Sponsor.find("byName", "newsagent").first();
    Club    dunmore   = Club.find("byName", "dunmore").first();
    Club    tramore   = Club.find("byName", "tramore").first();

    assertTrue(newsagent.support.contains(dunmore));
    assertTrue(newsagent.support.contains(tramore));

    assertTrue(dunmore.sponsors.contains(newsagent));
    assertTrue(tramore.sponsors.contains(newsagent));
}
```

'Long' Tests

```
@Test
public void testPlayerClubLong()
{
    Player jim;
    Club    dunmore;

    jim = Player.find("byName", "jim").first();
    assertNotNull(jim);
    assertEquals(jim.name, "jim");

    dunmore = jim.club;
    assertEquals("dunmore", dunmore.name);

    dunmore = Club.find("byName", "dunmore").first();
    assertNotNull(dunmore);
    assertEquals("dunmore", dunmore.name);
    assertEquals(2, dunmore.players.size());

    Player p1 = dunmore.players.get(0);
    assertTrue (p1.name.equals("jim") || p1.name.equals("mary"));
    Player p2 = dunmore.players.get(1);
    assertTrue (p2.name.equals("jim") || p2.name.equals("mary"));
}

@Test
public void testDivisionClubLong()
{
    Division senior = Division.find("byName", "senior").first();
    assertNotNull(senior);
    assertEquals(2, senior.members.size());

    Club c1 = senior.members.get(0);
    Club c2 = senior.members.get(1);

    assertTrue (c1.name.equals("tramore") || c1.name.equals("dunmore"));
    assertTrue (c2.name.equals("tramore") || c2.name.equals("dunmore"));
}
```


'Edit' Tests

```
@Test
public void testEditPlayerClub()
{
    Club    dunmore = Club.find("byName", "dunmore").first();
    Player jim      = Player.find("byName", "jim").first();
    Player mary     = Player.find("byName", "mary").first();

    dunmore.players.remove(mary);
    mary.delete();
    dunmore.save();

    assertEquals (dunmore.players.size(), 1);
    assertTrue (dunmore.players.contains(jim));

    assertEquals(0, Player.find("byName", "mary").fetch().size());

    Player sara      = new Player("sara");
    dunmore.addPlayer(sara);
    dunmore.save();
    assertEquals (dunmore.players.size(), 2);
}

@Test
public void testEditClubSponsor()
{
    Sponsor newsagent = Sponsor.find("byName", "newsagent").first();
    Club    dunmore   = Club.find("byName", "dunmore").first();

    assertEquals(2, newsagent.support.size());

    newsagent.support.remove(dunmore);
    dunmore.sponsors.remove(newsagent);

    newsagent.save();
    dunmore.save();

    assertEquals(1, newsagent.support.size());
}
```

Schema App

| Divisions | Clubs | Players | Sponsors |
|--------------------|---------|---------|----------|
| <h2>Divisions</h2> | | | |
| Division | Club | | |
| senior | tramore | | |
| | dunmore | | |
| junior | fenor | | |

Clubs

| Club | Players | |
|---------|---------|--------|
| dunmore | jim | Delete |
| | mary | |
| tramore | sam | Delete |
| | john | |
| fenor | mike | |
| | linda | |

New Club

Create new Club

Enter name of club



Club name

Division

senior

Save

Cancel

Players

| Player | Club | | |
|--------|---------|--------|------|
| jim | dunmore | Delete | Edit |
| mary | dunmore | Delete | Edit |
| sam | tramore | Delete | Edit |
| john | tramore | | |
| mike | fenor | | |
| linda | fenor | | |

New Player

Create new Player

Enter name and select club of new player



Player name

Club

dunmore

Save

Cancel

Divisions

Clubs

Players

Sponsors

Sponsors

Sponsor

newsagent

pub

Division & Sponsors Controllers

| Divisions | Clubs | Players | Sponsors |
|------------------|---------|---------|----------|
| Divisions | | | |
| Division | Club | | |
| senior | tramore | | |
| | dunmore | | |
| junior | fenor | | |

```
public class DivisionController extends Controller
{
    public static void index()
    {
        List<Division> divisions = Division.findAll();

        render (divisions);
    }
}
```

| Divisions | Clubs | Players | Sponsors |
|-----------------|-------|---------|----------|
| Sponsors | | | |
| Sponsor | | | |
| newsagent | | | |
| pub | | | |

```
public class SponsorsController extends Controller
{
    public static void index()
    {
        List<Sponsor> sponsors = Sponsor.findAll();

        render (sponsors);
    }
}
```

Clubs Controller (1)

| Club | Players | |
|---------|---------------|--------|
| dunmore | jim mary | Delete |
| tramore | sam john | Delete |
| fenor | mike linda | Delete |

New Club

Create new Club

Enter name of club

Club name

Division: senior

Save Cancel

```
public class ClubsController extends Controller
{
    public static void index()
    {
        List<Club> clubs = Club.findAll();
        render (clubs);
    }

    public static void newClub()
    {
        List<Division> divisions = Division.findAll();
        render(divisions);
    }

    public static void createClub (String name, String division)
    {
        Logger.info("name: " + name + "Division: " + division);

        Club club = new Club(name);
        club.save();

        Division theDivision = Division.findByName(division);
        if (theDivision != null)
        {
            theDivision.addClub(club);
            theDivision.save();
        }
        index();
    }

    ...
}
```

Clubs Controller (2)

| Club | Players | |
|---------|---------------|--------|
| dunmore | jim mary | Delete |
| tramore | sam john | Delete |
| fenor | mike linda | Delete |

New Club

```
public class ClubsController extends Controller
{

    //...
    public static void delete(Long id)
    {
        Club club = Club.findById(id);
        if (club != null)
        {
            Logger.info("Trying to delete " + club.name);
            List<Division> divisions = Division.findAll();
            for (Division division : divisions)
            {
                if (division.members.contains(club))
                {
                    division.members.remove(club);
                    division.save();
                    Logger.info ("removing club from division");
                }
            }
            club.delete();
        }
        index();
    }

}
```


Players Controller (1)

Divisions

Clubs

Players

Sponsors

Players

| Player | Club | | |
|--------|---------|--------|------|
| jim | dunmore | Delete | Edit |
| mary | dunmore | Delete | Edit |
| sam | tramore | Delete | Edit |
| john | tramore | Delete | Edit |
| mike | fenor | Delete | Edit |
| linda | fenor | Delete | Edit |

New Player

Create new Player

Enter name and select club of new player

Player name

Club

dunmore

Save

Cancel

```
public class PlayersController extends Controller
{
    public static void index()
    {
        List<Player> players = Player.findAll();
        render (players);
    }

    public static void delete(Long id)
    {
        Player player = Player.findById(id);
        if (player != null)
        {
            player.club.removePlayer(player);
            player.club.save();
            player.delete();
        }
        index();
    }

    public static void newPlayer()
    {
        List<Club> clubs = Club.findAll();
        render(clubs);
    }

    public static void createPlayer(String name, String club)
    {
        Logger.info("Name: " + name + ": Club: " + club);

        Player player = new Player (name);
        Club theClub = Club.findByName(club);
        if (theClub != null)
        {
            theClub.addPlayer(player);
            theClub.save();
        }
        index();
    }
}
```

Players Controller (2)

| Player | Club | | |
|--------|---------|--------|------|
| jim | dunmore | Delete | Edit |
| mary | dunmore | Delete | Edit |
| sam | tramore | Delete | Edit |
| john | tramore | Delete | Edit |
| mike | fenor | Delete | Edit |
| linda | fenor | Delete | Edit |

New Player

Create new Player

Enter name and select club of new player

Player name

Club

dunmore

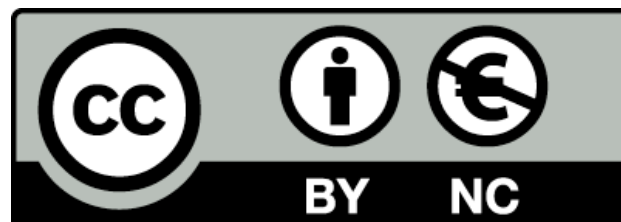
Save

Cancel

```
public class PlayersController extends Controller
{

    //...
    public static void changePlayer(Long id, String name, Long club)
    {
        Player player = Player.findById(id);
        if (player != null)
        {
            player.name = name;
            Club theClub = Club.findById(club);
            player.club = theClub;
            player.save();
        }
        index();
    }

    public static void editPlayer(Long id)
    {
        Player player = Player.findById(id);
        List<Club> clubs = Club.findAll();
        Integer clubIndex = clubs.indexOf(player.club);
        clubIndex++;
        render(player, clubs, clubIndex);
    }
}
```



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

