



Связь один ко многим. Практический пример

Последнее обновление: 31.10.2015



Воспользуемся теоретическим материалом из прошлой темы и создадим новое приложение, в котором будет реализована связь один-ко-многим. Для реализации подобной связи будем использовать lazy loading.

Создадим новый проект Windows Forms. Первым делом подключим через NuGet Entity Framework и добавим все наши модели. Итак, добавим следующие классы:

```
using System.Data.Entity;

public class Player
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Position { get; set; }
    public int Age { get; set; }

    public int? TeamId { get; set; }
    public virtual Team Team { get; set; }
}

public class Team
{
    public int Id { get; set; }
    public string Name { get; set; } // название команды
    public string Coach { get; set; } // тренер

    public virtual ICollection<Player> Players { get; set; }

    public Team()
    {
        Players = new List<Player>();
    }

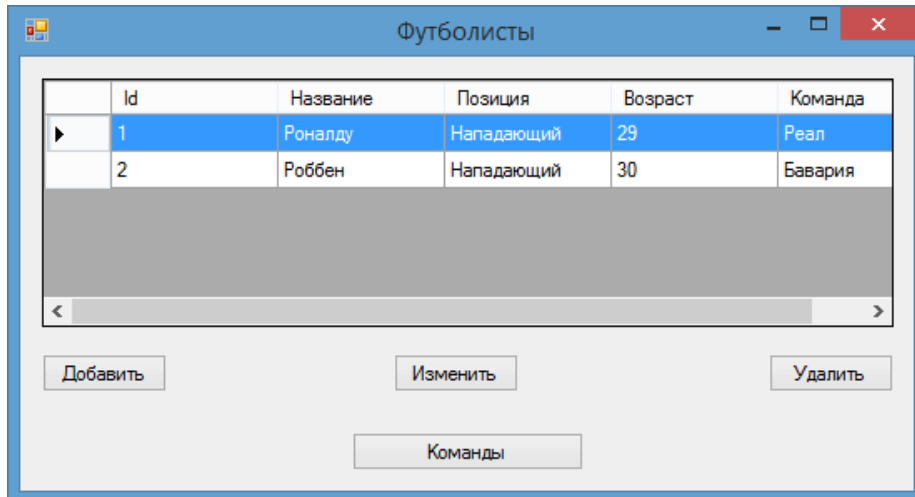
    public override string ToString()
    {
        return Name;
    }
}

class SoccerContext : DbContext
{
    public SoccerContext()
        :base("SoccerDb")
    {}

    public DbSet<Player> Players { get; set; }
    public DbSet<Team> Teams { get; set; }
}
```

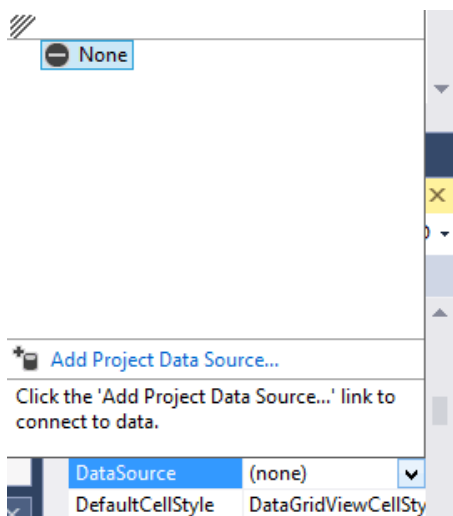
В нашем графическом приложении будет четыре формы: для списка футболистов, для списка команд, для добавления/редактирования футболиста и для добавления/изменения команды.

Пусть форма, которая уже есть по умолчанию, будет представлять футболистов и иметь следующий вид:

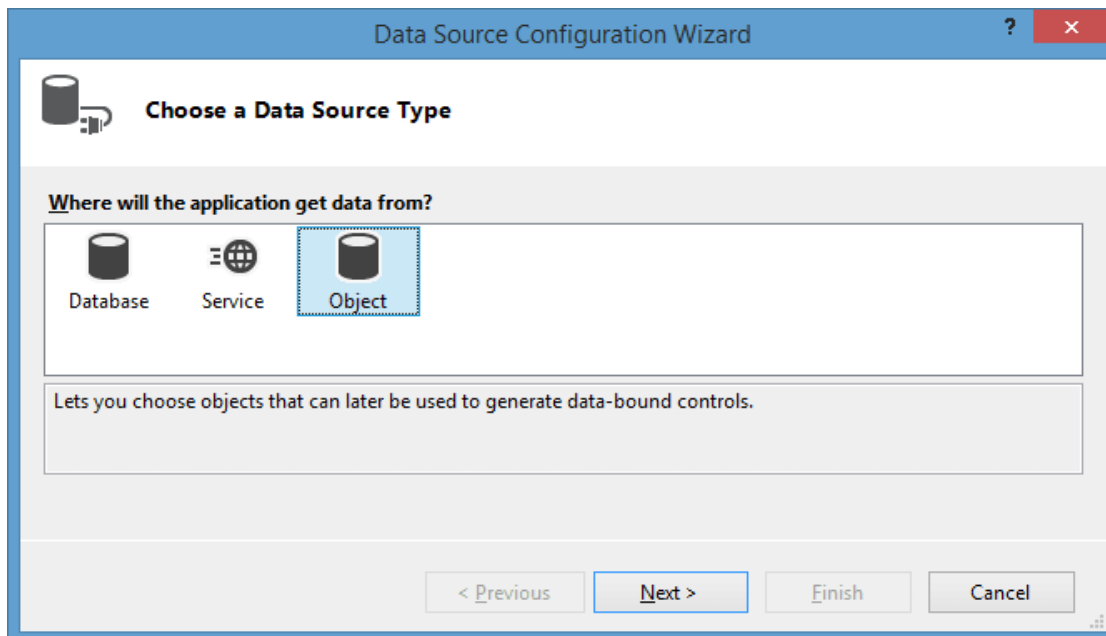


Здесь у нас элемент DataGridView для отображения данных, а также три кнопки для добавления/редактирования/удаления и одна кнопка для вызова окна с футбольными командами.

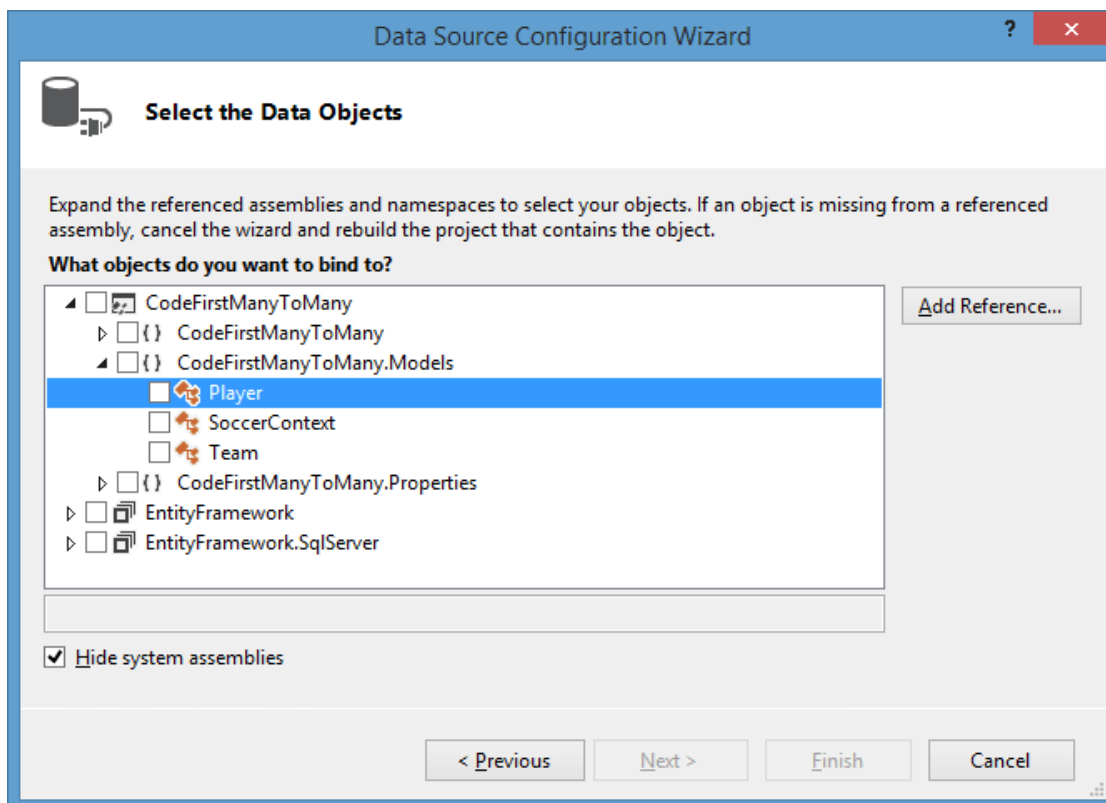
Итак, в предыдущих темах мы уже рассматривали привязку данных к DataGridView. При привязке для каждого свойства создается столбец. Однако свойство TeamId нам не нужно. Да и было бы неплохо, если бы в качестве заголовков отображались те названия, какие мы хотим, а не названия свойств. Поэтому выделим DataGridView и окне свойств найдем для него свойство **DataSource**. Нажмем, чтобы установить для него значение, и нам отобразится окно выбора источника данных:



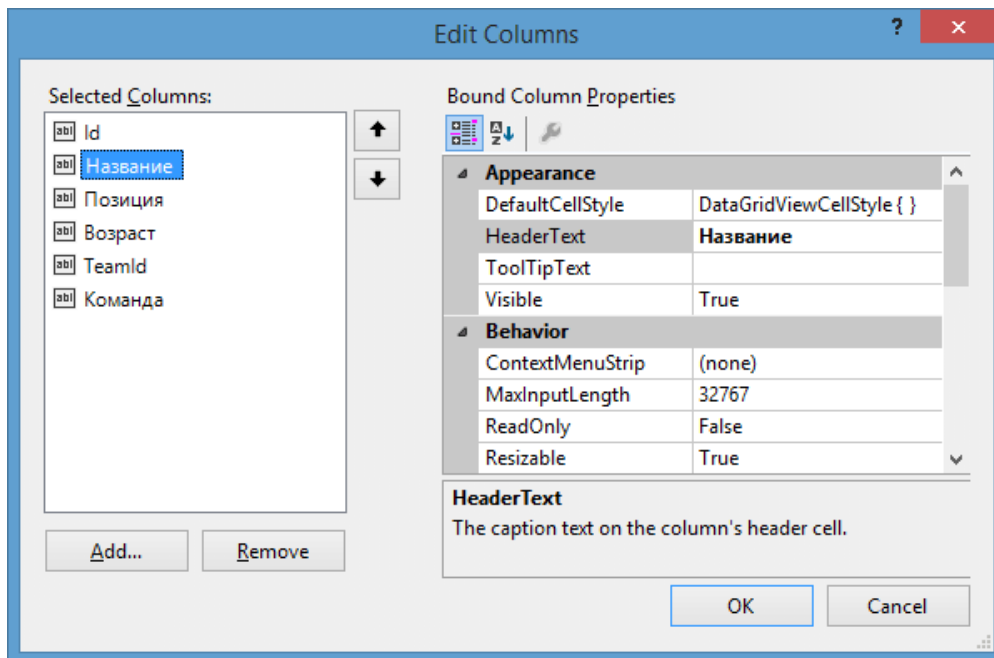
Нажмем на ссылку **Add Project Data Source....** После этого нам откроется окно мастера настройки источника данных, в котором нам надо выбрать Object:



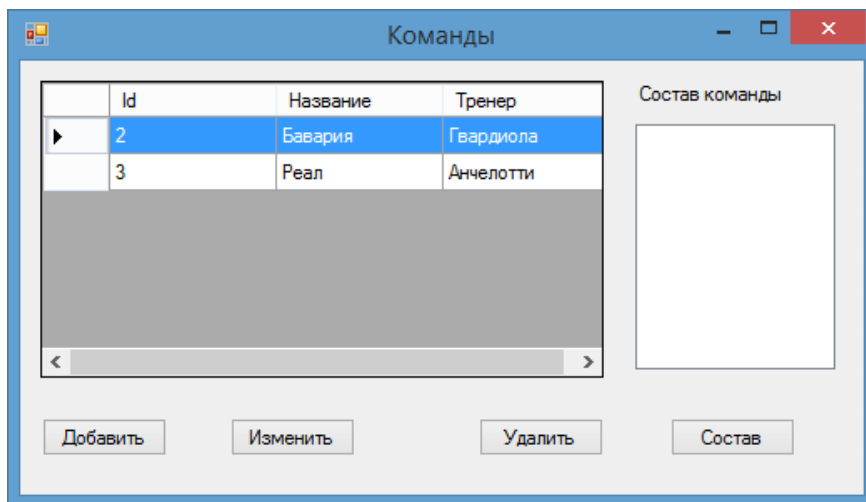
И затем на следующем шаге нам отобразится структура проекта, в которой в одном из узлов найдем наш класс Player:



После этого в DataGridView будут добавлены заголовки по именам свойств. Перейдем в свойство Columns у DataGridView. В свойстве HeaderText установим для всех заголовков предпочтительное название, которое будет отображаться, а столбец TeamId удалим.

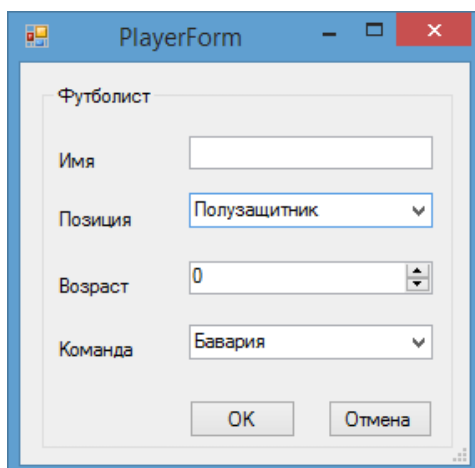


Подобным образом сделаем графический интерфейс и для формы с командами, назовем ее, к примеру, AllTeams:



Здесь также DataGridView для отображения списка команд, а также поле ListBox и кнопка 'Состав' для вывода в этом поле всех игроков выбранной команды.

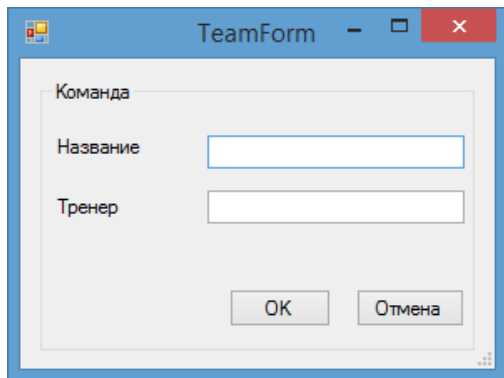
Добавим также дополнительные формы для создания и изменения игрока и команды. Форма для игрока, назовем ее PlayerForm:



Здесь текстовое поле для имени игрока, элемент NumericUpDown для указания возраста, и два элемента ComboBox.

Для кнопки 'ОК' у свойства **DialogResult** установим значение OK, а у кнопки 'Отмена' установим значение Cancel. И изменим у всех полей значение свойства **Modifiers** с Private на Protected Internal.

И форма для создания команды TeamForm:



Для кнопок и полей также настроим свойство DialogResult и Modifiers, как и у предыдущей формы.

Код главной формы с игроками:

```
public partial class Form1 : Form
{
    SoccerContext db;
    public Form1()
    {
        InitializeComponent();

        db = new SoccerContext();
        db.Players.Load();
        dataGridView1.DataSource = db.Players.Local.ToBindingList();
    }
    // добавление
    private void button1_Click(object sender, EventArgs e)
    {
        PlayerForm plForm = new PlayerForm();

        // из команд в бд формируем список
        List<Team> teams = db.Teams.ToList();
        plForm.comboBox2.DataSource = teams;
        plForm.comboBox2.ValueMember = "Id";
        plForm.comboBox2.DisplayMember = "Name";

        DialogResult result = plForm.ShowDialog(this);

        if (result == DialogResult.Cancel)
            return;

        Player player = new Player();
        player.Age = (int)plForm.numericUpDown1.Value;
        player.Name = plForm.textBox1.Text;
        player.Position = plForm.comboBox1.SelectedItem.ToString();
        player.Team = (Team)plForm.comboBox2.SelectedItem;

        db.Players.Add(player);
        db.SaveChanges();

        MessageBox.Show("Новый футболист добавлен");
    }

    // редактирование
    private void button2_Click(object sender, EventArgs e)
    {

```

```
if (dataGridView1.SelectedRows.Count > 0)
{
    int index = dataGridView1.SelectedRows[0].Index;
    int id = 0;
    bool converted = Int32.TryParse(dataGridView1[0, index].Value.ToString(), out id);
    if (converted == false)
        return;

    Player player = db.Players.Find(id);

    PlayerForm plForm = new PlayerForm();
    plForm.numericUpDown1.Value = player.Age;
    plForm.comboBox1.SelectedItem = player.Position;
    plForm.textBox1.Text = player.Name;

    List<Team> teams = db.Teams.ToList();
    plForm.comboBox2.DataSource = teams;
    plForm.comboBox2.ValueMember = "Id";
    plForm.comboBox2.DisplayMember = "Name";

    if(player.Team!=null)
        plForm.comboBox2.SelectedValue = player.Team.Id;

    DialogResult result = plForm.ShowDialog(this);

    if (result == DialogResult.Cancel)
        return;

    player.Age = (int)plForm.numericUpDown1.Value;
    player.Name = plForm.textBox1.Text;
    player.Position = plForm.comboBox1.SelectedItem.ToString();
    player.Team = (Team)plForm.comboBox2.SelectedItem;

    db.Entry(player).State = EntityState.Modified;
    db.SaveChanges();

    MessageBox.Show("Объект обновлен");
}
}
// удаление
private void button3_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        int index = dataGridView1.SelectedRows[0].Index;
        int id = 0;
        bool converted = Int32.TryParse(dataGridView1[0, index].Value.ToString(), out id);
        if (converted == false)
            return;

        Player player = db.Players.Find(id);
        db.Players.Remove(player);
        db.SaveChanges();

        MessageBox.Show("Объект удален");
    }
}
// открываем форму с командами
private void button4_Click(object sender, EventArgs e)
{
    AllTeams teams = new AllTeams();
    teams.Show();
}
```

```

    }
}

```

Тут тот же самый функционал, что рассматривался на примере простого приложения в одной из предыдущих тем. Только появляется дополнительное поле для выбора команды, в которое нам сначала надо загрузить все команды и настроить привязку:

```

List<Team> teams = db.Teams.ToList();
p1Form.comboBox2.DataSource = teams;
p1Form.comboBox2.ValueMember = "Id";
p1Form.comboBox2.DisplayMember = "Name";

```

А при редактировании мы устанавливаем для этого поля значение, равное TeamId: `p1Form.comboBox2.SelectedValue = player.Team.Id`. Здесь благодаря lazy loading мы можем получить связанную с игроком команду и обратиться к ее свойствам.

Код формы команд:

```

public partial class AllTeams : Form
{
    SoccerContext db;
    public AllTeams()
    {
        InitializeComponent();

        db = new SoccerContext();
        db.Teams.Load();
        dataGridView1.DataSource = db.Teams.Local.ToBindingList();
    }

    // добавление
    private void button1_Click(object sender, EventArgs e)
    {
        TeamForm tmForm = new TeamForm();
        DialogResult result = tmForm.ShowDialog(this);

        if (result == DialogResult.Cancel)
            return;

        Team team = new Team();
        team.Name = tmForm.textBox1.Text;
        team.Coach = tmForm.textBox2.Text;

        db.Teams.Add(team);
        db.SaveChanges();
        MessageBox.Show("Новый объект добавлен");
    }

    // просмотр списка игроков команды
    private void button4_Click(object sender, EventArgs e)
    {
        if (dataGridView1.SelectedRows.Count > 0)
        {
            int index = dataGridView1.SelectedRows[0].Index;
            int id = 0;
            bool converted = Int32.TryParse(dataGridView1[0, index].Value.ToString(), out id);
            if (converted == false)
                return;

            Team team = db.Teams.Find(id);
            listBox1.DataSource = team.Players.ToList();
            listBox1.DisplayMember = "Name";
        }
    }

    private void button3_Click(object sender, EventArgs e)

```

```
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        int index = dataGridView1.SelectedRows[0].Index;
        int id = 0;
        bool converted = Int32.TryParse(dataGridView1[0, index].Value.ToString(), out id);
        if (converted == false)
            return;

        Team team = db.Teams.Find(id);
        team.Players.Clear();
        db.Teams.Remove(team);
        db.SaveChanges();

        MessageBox.Show("Объект удален");
    }
}

// редактирование
private void button2_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count > 0)
    {
        int index = dataGridView1.SelectedRows[0].Index;
        int id = 0;
        bool converted = Int32.TryParse(dataGridView1[0, index].Value.ToString(), out id);
        if (converted == false)
            return;

        Team team = db.Teams.Find(id);

        TeamForm tmForm = new TeamForm();
        tmForm.textBox1.Text = team.Name;
        tmForm.textBox2.Text = team.Coach;

        DialogResult result = tmForm.ShowDialog(this);
        if (result == DialogResult.Cancel)
            return;

        team.Name = tmForm.textBox1.Text;
        team.Coach = tmForm.textBox2.Text;

        db.Entry(team).State = EntityState.Modified;
        db.SaveChanges();
        MessageBox.Show("Объект обновлен");
    }
}
}
```

На что здесь надо обратить внимание? Во-первых, здесь так же благодаря lazy loading мы можем получить связанных с командой игроков и загрузить их в ListBox: `listBox1.DataSource = team.Players.ToList();`

Во-вторых, при удалении мы предварительно очищаем данный список: `team.Players.Clear();`. Если бы мы вручную создавали базу данных, а потом через entity framework через database first или code first подключали бы к проекту, то мы могли бы не очищать список, установив при создании внешнего ключа в бд каскадное удаление или установку в null поля игрока при удалении связанной команды.

[Назад](#) [Содержание](#) [Вперед](#)



[Вконтакте](#) | [Twitter](#) | [Google+](#) | [Канал сайта на youtube](#) | [Помощь сайту](#)

Контакты для связи: metanit22@mail.ru

Copyright © metanit.com, 2012-2017. Все права защищены.