

## **Комп'ютерний практикум 12**

### **Створення додатку MVC ASP.NET з використанням WebApi.**

#### **Мета**

Ознайомлення з можливостями WebApi практичне застосування їх в MVC ASP.NET.

#### **Зміст**

1. Підключення вже існуючої бази даних за допомогою Entity Framework.
2. Створення прошарку Api.
3. Налаштування взаємодії між Api прошарком та Entity Framework.
4. Налаштування роутінгу API контроллеру .
5. Реалізація CRUD операцій за допомогою WebAPI.

#### **Література**

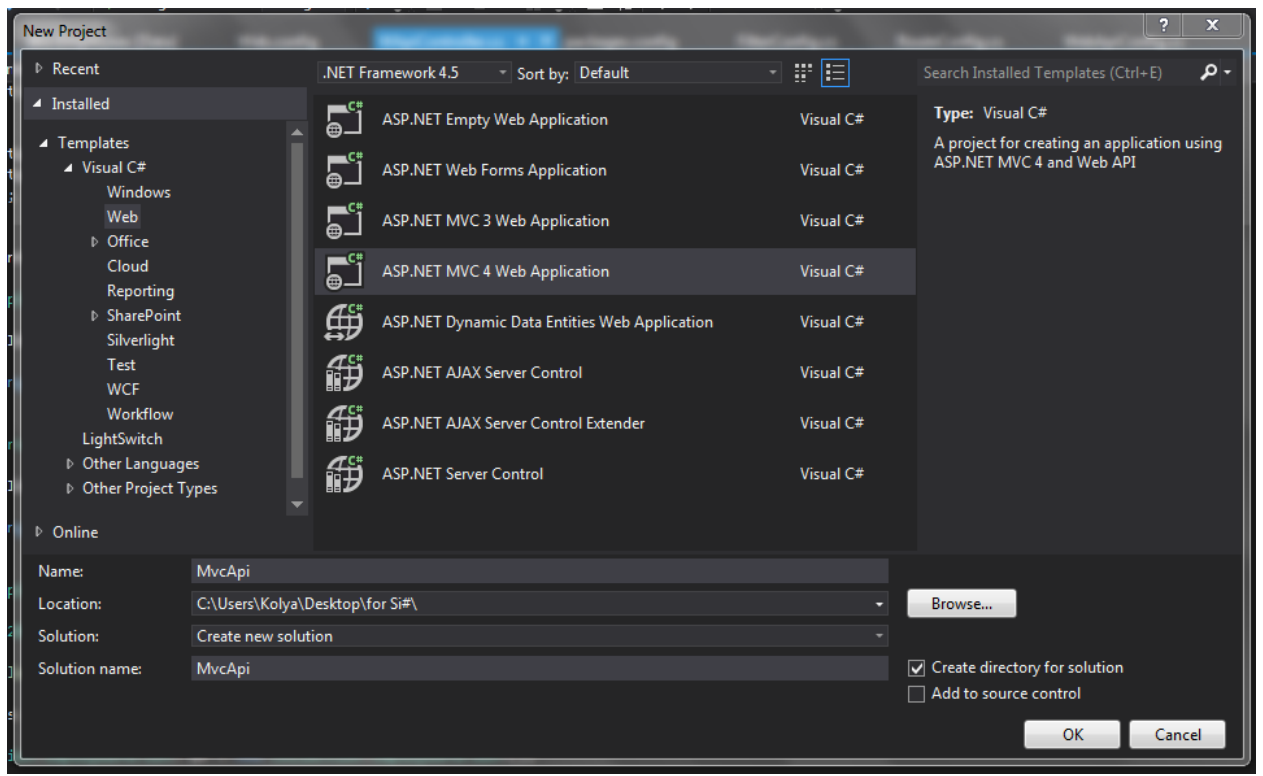
- <http://www.asp.net/aspnet/samples/aspnet-web-api>
- <http://www.asp.net/web-api>

#### **Частина 1**

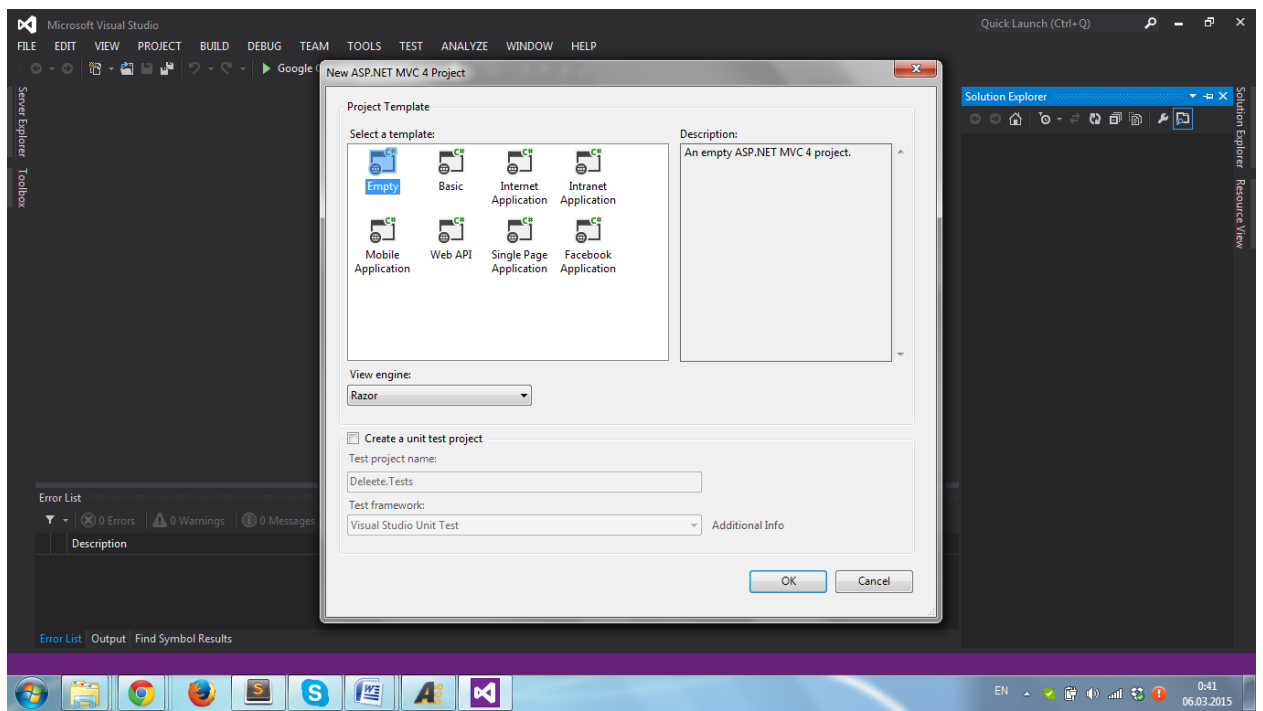
##### **Виконання:**

На основі створеної бази даних у лабораторному практикумі 11 реалізувати прошарок Api для отримання даних з БД (Перелік співробітників, спеціалізація співробітника) і повернення їх у форматі JSON.

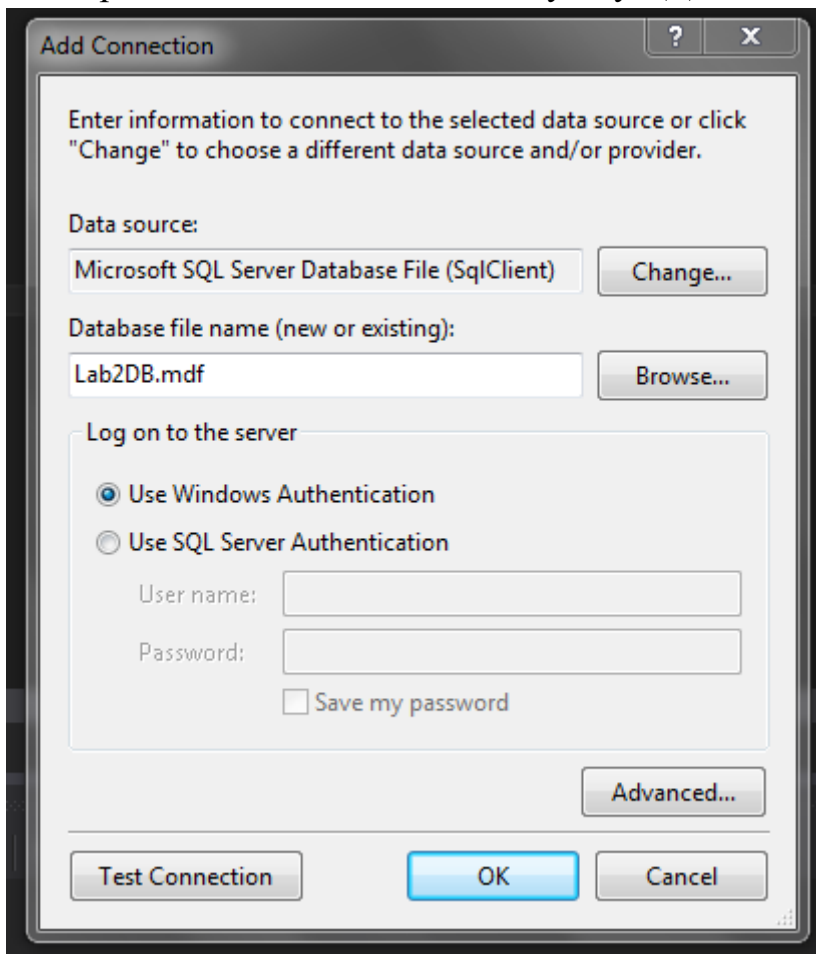
- 1.Сворити порожній проект ASP.NET MVC4



2. Обрати порожній проект. Для опису представлення буде використано синтаксис Razor.



3. У проект підключаємо вже існуючу БД.

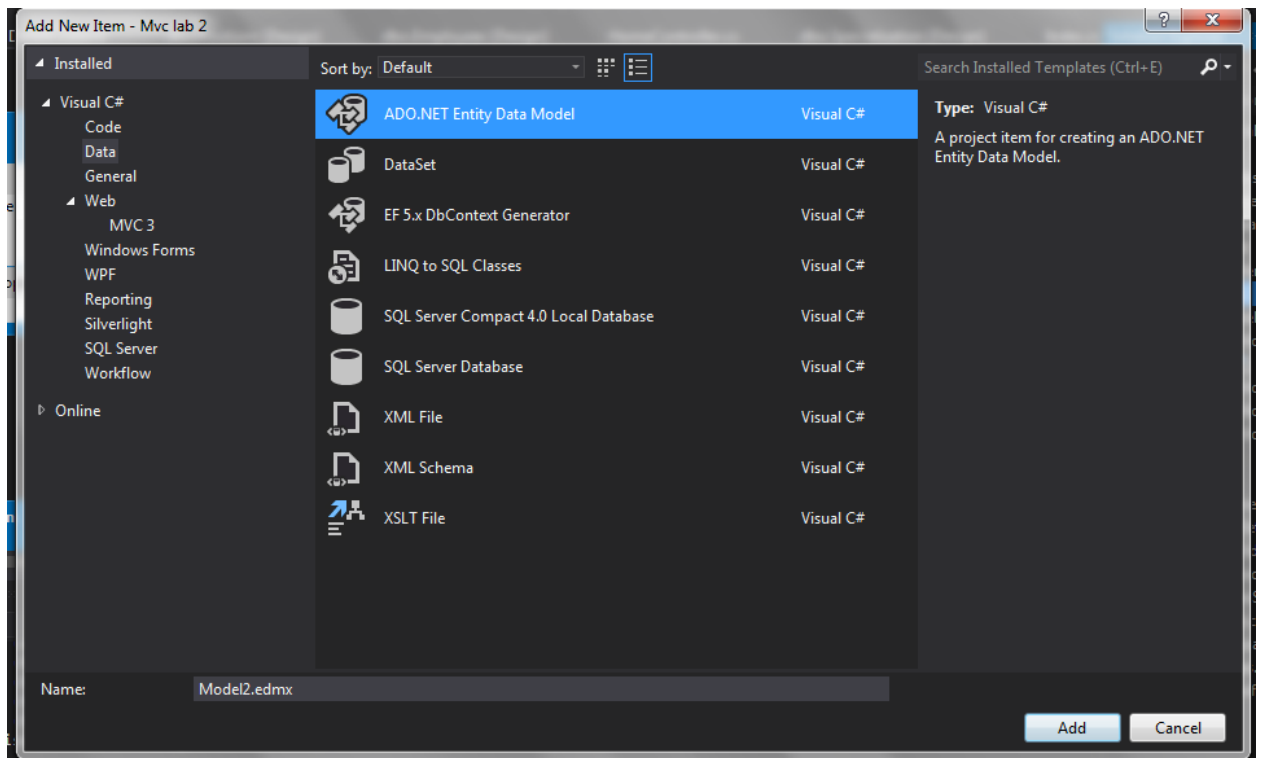


Обов'язково натискаємо кнопку Test Connection, для перевірки з'єднання.

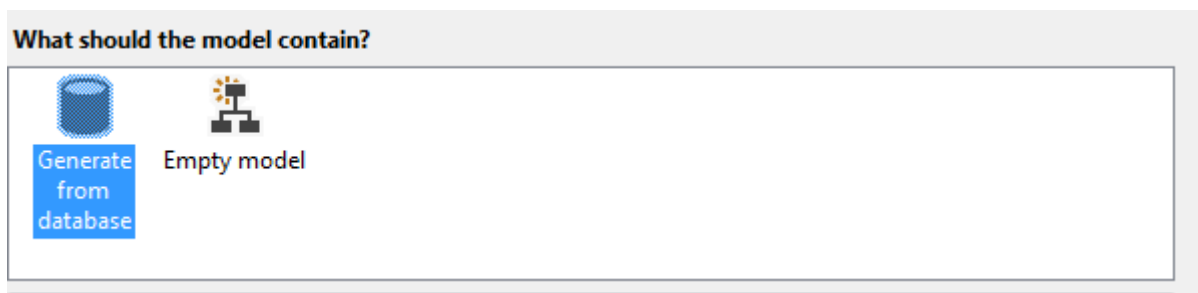
Після цього ваше з'єднання повинно з'явитися у вкладці Server Explorer.

Створена база даних не буде мати таблиць, тому їх треба створити, що і буде виконано на наступному кроці.

4. Для налагодження роботи між контролером та даними що зберігаються у БД необхідно створити модель, яка виступить цим зв'язуючим прошарком. Для цього на теці "Models" визиваємо контекстне меню і додаємо нову Entity Data Model.




Далі виконуємо наступні дії:



Обираємо наше підключення:

Entity Data Model Wizard

 **Choose Your Data Connection**

Which data connection should your application use to connect to the database?

Lab2DB.mdf New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Entity connection string:

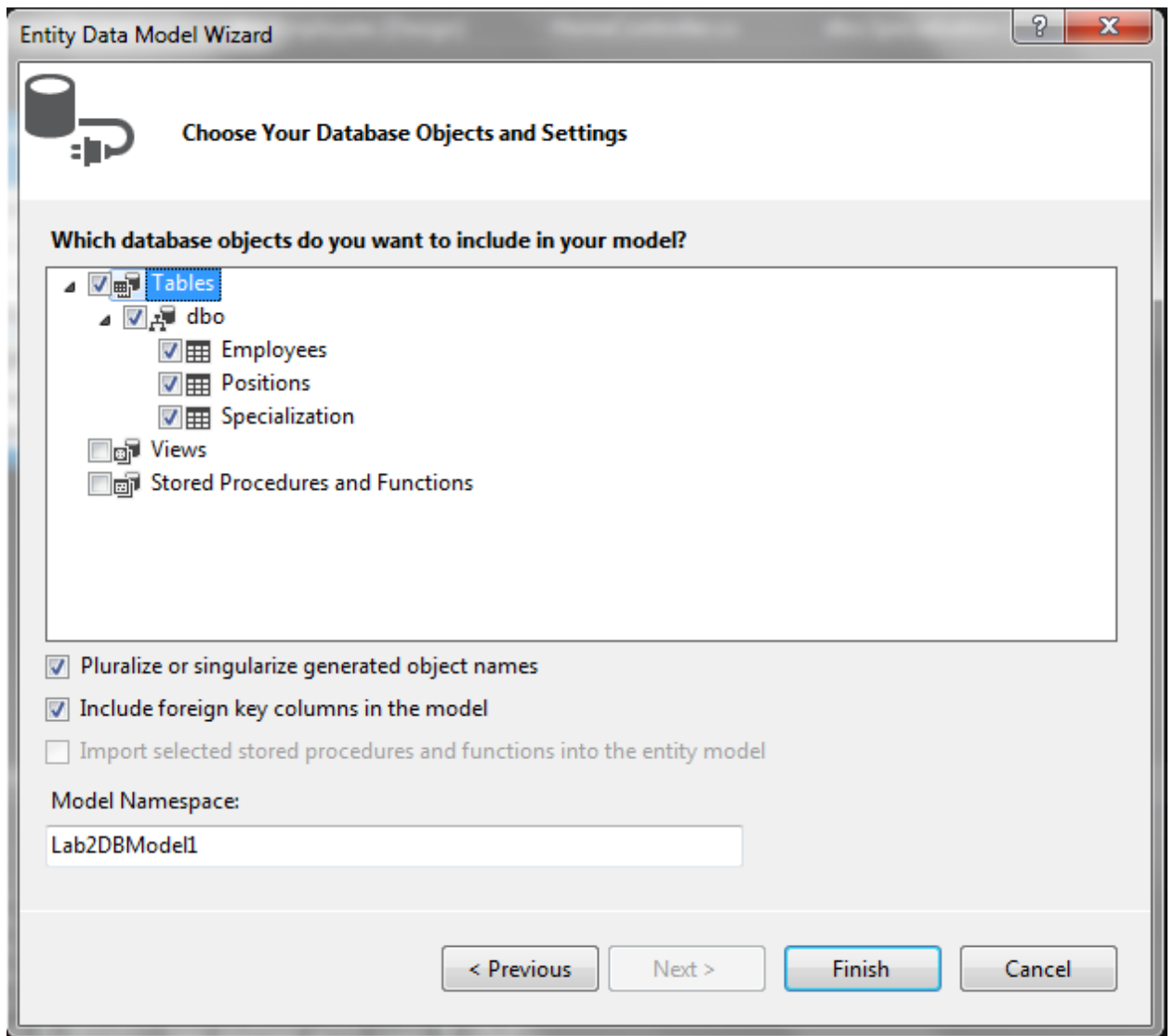
```
metadata=res://*/Models.Model2.csdl|res://*/Models.Model2.ssdl|
res://*/Models.Model2.msl;provider=System.Data.SqlClient;provider connection string='data source=
(LocalDB)\v11.0;attachdbfilename="C:\Users\Kolya\Desktop\for Si#\Mvc lab 2_bad\Mvc lab
2\App_Data\Lab2DB.mdf";integrated security=True;connect
timeout=30;MultipleActiveResultSets=True;App=EntityFramework'
```

☒ Save entity connection settings in Web.Config as:

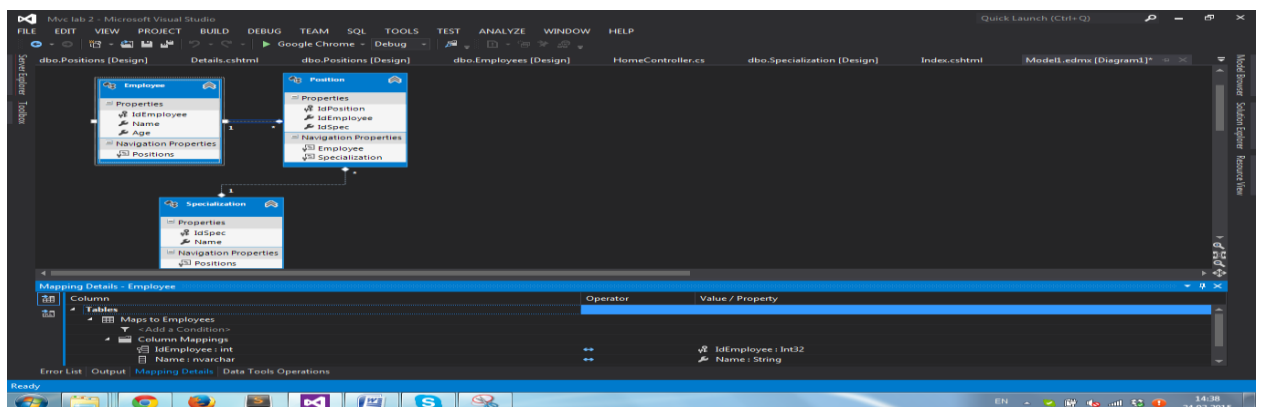
Lab2DBEntities1

< Previous Next > Finish Cancel

Обираємо всі таблиці, обираємо підключення зовнішніх ключів та генерацію одиничних відмінків у назвах класів.



Після натискання кнопки завершити, отримаємо наступну модель:



Обравши відповідну таблицю та натиснувши «Table mapping» можна побачити у який тип згенерується кожне поле таблиці.

Також було згенеровано класи .Context.cs, Employee.cs, Position.cs, Specialization.cs, які описують таблиці нашої БД і за допомогою яких ми будемо взаємодіяти з нею через контролер.

```
namespace Mvc_lab_2.Models
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;

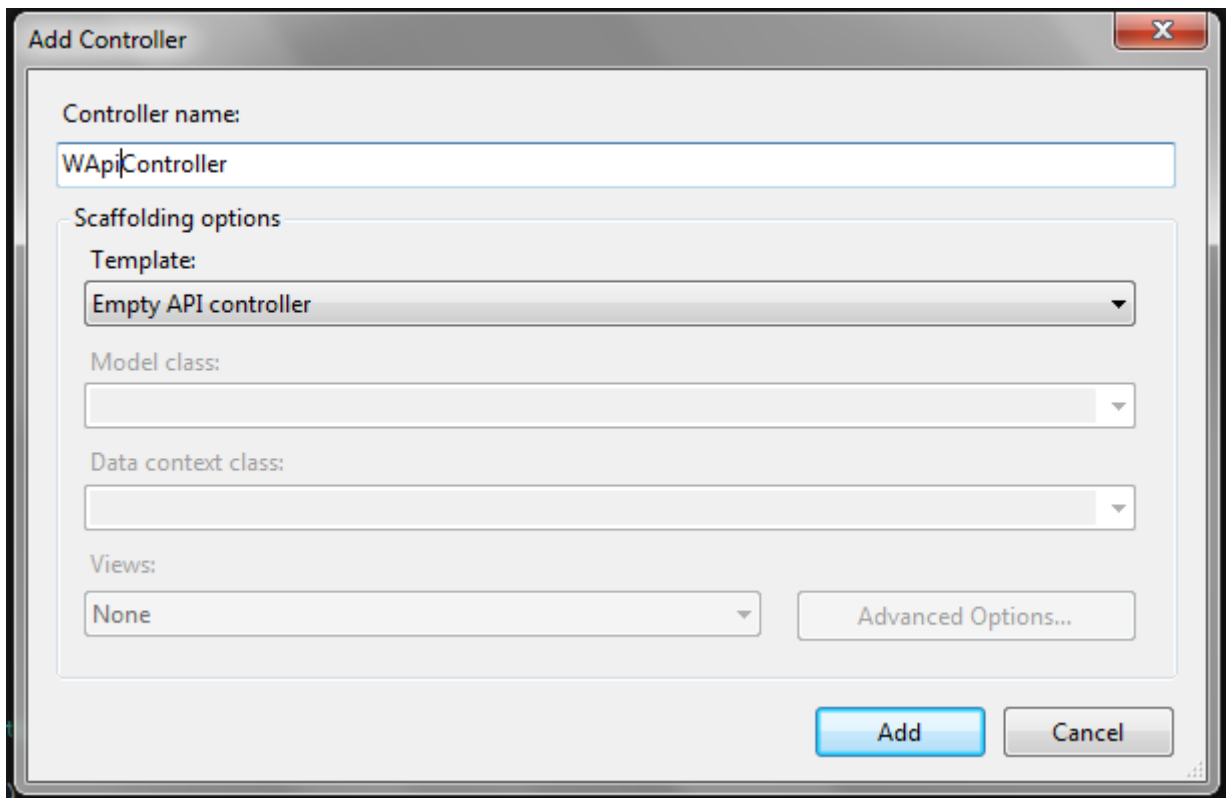
    public partial class Lab2DBEntities : DbContext
    {
        public Lab2DBEntities()
            : base("name=Lab2DBEntities")
        {
        }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }

        public DbSet<Employee> Employees { get; set; }
        public DbSet<Position> Positions { get; set; }
        public DbSet<Specialization> Specializations { get; set; }
    }
}
```

Обов'язково треба перевірити у файлі Web.config наявність відповідної до поточного з'єднання ConnectionString, якщо вона відсутня скопіювати її з минулих проектів.

5. Створимо новий контролер, назвемо його WApiController обравши тип контролера API:



Методи WApi контролеру доступні за наступною адресою «localhost:port/api/WApi». За замовчуванням методи даного контролеру будуть повертати данні у форматі XML, оскільки наше Api повинно повертати дані у вигляді JSON.

Для цього необхідно внести наступні зміни у файл App\_Start\WebApiConfig.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http.Headers;
using System.Web.Http;

namespace MvcApi
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.Routes.MapHttpRoute(
                name: "WApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );

            config.Formatters.JsonFormatter.SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/html"));
        }
    }
}
```



6. Додамо у наш Арі контролер клас EmployeePerson який описує відображувану Арі сутність співробітника та метод GetEmpr який буде повертати перелік співробітників:

```
namespace MvcApi.Controllers
{
    public class EmployeePerson
    {
        public int Id { get; set; }

        public string Name { get; set; }
    }
}
```

```
public ICollection<EmployeePerson> GetEmp()
{
    var emps = (from employees in db.Employees select employees).ToList();

    Collection<EmployeePerson> EP = new Collection<EmployeePerson>();

    foreach(Employee e in emps) {
        EP.Add(new EmployeePerson{Id = e.IdEmployee, Name = e.Name });
    }
    return EP;
}
```

7. Додамо у наш Арі контролер клас PersonSpecialization який описує відображувану Арі сутність спеціалізації співробітника та метод GetPos який буде повертати перелік співробітників:

```
public class PersonSpecialization
{
    public int Id { get; set; }

    public string Name { get; set; }
}
```

```
public ICollection<PersonSpecialization> GetPos(int id)
{
    var emps = (from position in db.Positions where position.IdEmployee == id select position.Specialization).ToList();

    Collection<PersonSpecialization> SP = new Collection<PersonSpecialization>();

    foreach (Specialization s in emps)
    {
        SP.Add(new PersonSpecialization { Id = s.IdSpec, Name = s.Name });
    }
    return SP;
}
```

8. Весь код WАрі контролера:

```
MvcApi.Controllers.WApiController

using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using MvcApi.Models;

namespace MvcApi.Controllers
{
    public class EmployeePerson...
    public class PersonSpecialization...
    public class WApiController : ApiController
    {
        private Lab2DBEntities db = new Lab2DBEntities();

        public ICollection<EmployeePerson> GetEmp()...
        public ICollection<PersonSpecialization> GetPos(int id)...)
    }
}
```

9. Виконаємо білд проекту, запустимо його та перевіримо роботу WApi контролеру, адреса контролера <http://localhost:57119/api/WApi>. За даною адресою виконається Арі-функція GetEmp.

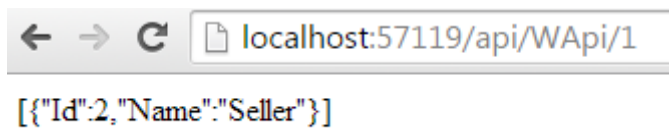
Отримаємо наступний результат:

```
localhost:57119/api/WApi
[{"Id":1,"Name":"Peter"}, {"Id":2,"Name":"Kate"}]
```

Контролер повернув нам перелік співробітників у JSON форматі.

10. Передамо WApi контролеру параметр. Адресна строка буде мати вигляд <http://localhost:57119/api/WApi/1>. За даною адресою виконається Арі-функція GetPos.

Отримаємо наступний результат:



Контролер повернув спеціалізацію співробітника у JSON форматі.

Додаток працює коректно.

### **Вимоги до виконання роботи:**

- Реалізувати додаток згідно варіанту
- Підготувати звіт до програми, який буде містити:
  - зміст завдання
  - текст додатку з коментарями
  - результати перевірки коректності роботи додатку на різних значеннях параметрів (у тому числі і помилкових)
  - формулювання призначення моделі, представлення та контролера.

### **Завдання:**

На основі створеної бази даних у минулому лабораторному практикумі реалізувати прошарок Арі для отримання даних з БД і повернення їх у форматі JSON.

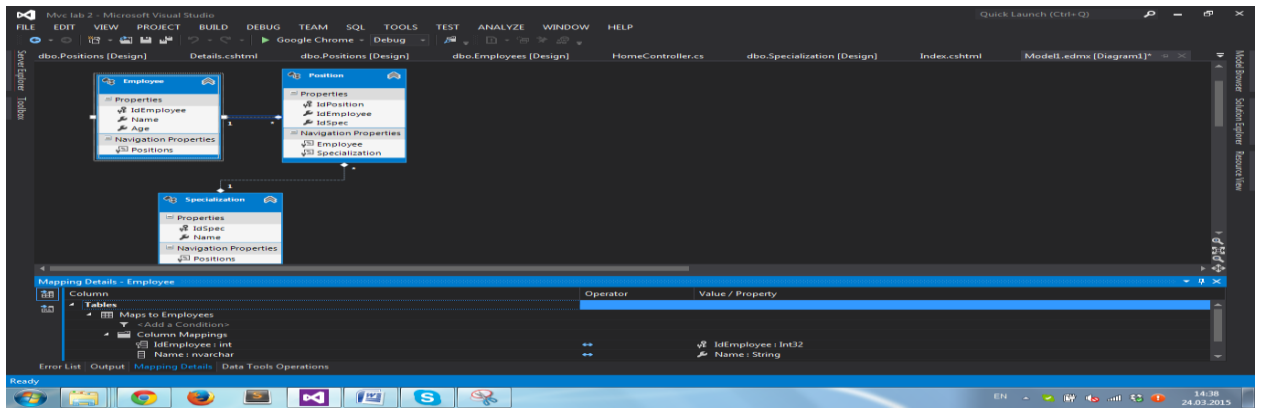
## **Частина 2**

### **Виконання:**

Реалізувати прошарок Арі для додавання, оновлення, видалення та отримання даних з БД (Перелік співробітників, спеціалізація співробітника) і повернення статусу виконаної операції у форматі JSON.

1. Відкриємо проект лабораторного практикуму. В цьому проекті було створено АРІ контролер, під'єднано Entity модель раніше створеної БД та реалізовано операції отримання даних з БД:

*Модель БД*



## API контроллер

```
MvcApi.Controllers.WApiController

using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using MvcApi.Models;

namespace MvcApi.Controllers
{
    public class EmployeePerson...
    public class PersonSpecialization...
    public class WApiController : ApiController
    {
        private Lab2DBEntities db = new Lab2DBEntities();

        public ICollection<EmployeePerson> GetEmp()...
        public ICollection<PersonSpecialization> GetPos(int id)...)
    }
}
```

## Результат у форматі JSON

localhost:57119/api/WApi

```
[{"Id":1,"Name":"Peter"}, {"Id":2,"Name":"Kate"}]
```

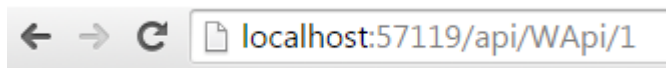
2. В данному проєкті API контролер сам обирає який Action викликати в залежності від отриманого шаблону параметрів. Конфігураційний файл при таких налаштуваннях має наступний вигляд:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http.Headers;
using System.Web.Http;

namespace MvcApi
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.Routes.MapHttpRoute(
                name: "WApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );

            config.Formatters.JsonFormatter.SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/html"));
        }
    }
}
```

URL:



Такий шаблон роутінгу не є досить гнучким та зручним. Зручніше було б власноруч обирати Action який треба викликати. В такому разі необхідно додати в існуючий шаблон додатковий прошарок «action», для цього треба внести певні зміни у конфігураційний файл роутінгу, після чого він матиме наступний вигляд:

```
config.Routes.MapHttpRoute(
    name: "WApi",
    routeTemplate: "api/{controller}/{action}/{id}",
    defaults: new { id = RouteParameter.Optional }
);

config.Formatters.JsonFormatter.SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/html"));
```

Після внесення змін у конфігураційний файл роутінгу, треба додати відповідні назви екшнів у контролері. Для цього треба над кожним екшином додати атрибут ActionName з відповідною назвою екшину.

Після внесення змін контролер матиме наступний вигляд:

```

namespace MvcApi.Controllers
{
    public class EmployeePerson...
    public class PersonSpecialization...
    public class WApiController : ApiController
    {
        private Lab2DBEntities db = new Lab2DBEntities();

        [HttpGet]
        [ActionName("GetEmp")]
        public ICollection<EmployeePerson> GetEmp()...

        [HttpGet]
        [ActionName("GetPos")]
        public ICollection<PersonSpecialization> GetPos(int id)...
    }
}

```

Новий шаблон роутінгу матиме наступний URL:

 [localhost:57119/api/WApi/GetEmp/1](http://localhost:57119/api/WApi/GetEmp/1)

3. Реалізуємо можливість створення нового запису у таблиці БД. Для цього додамо у наш контролер екшн типу POST з назвою CreateEmp який на вхід буде приймати об'єкт Employee. Цей екшн матиме наступний вигляд:

```

[HttpPost]
[ActionName("CreateEmp")]
public HttpResponseMessage CreateEmp(Employee emp)
{
    var response = Request.CreateResponse(HttpStatusCode.OK);

    try
    {
        db.Employees.Add(emp);
        db.SaveChanges();
        response.Content = new StringContent("{Id:"+emp.IdEmployee+",Name:"+emp.Name+",Age:"+emp.Age+}", Encoding.UTF8, "application/json");
    }
    catch (Exception ex)
    {
        response.Content = new StringContent("{Error:"+ex.Message+}", Encoding.UTF8, "application/json");
    }
    return response;
}

```

У разі вдалого створення нової сутності співробітника, екшн поверне нову сутність співробітника у форматі JSON, у разі помилки буде повернено текст помилки у форматі JSON.

```
[HttpPost]
[ActionName("UpdateEmp")]
public HttpResponseMessage UpdateEmp(Employee sEmp)
{
    var response = Request.CreateResponse(HttpStatusCode.OK);
    var emp = (from o in db.Employees where o.IdEmployee == sEmp.IdEmployee select o).First();

    try
    {
        db.Employees.Remove(emp);
        db.Employees.Add(sEmp);
        db.SaveChanges();
        response.Content = new StringContent("{Id:" + sEmp.IdEmployee + ",Name:" + sEmp.Name + ",Age:" + sEmp.Age + "}", Encoding.UTF8, "application/json");
    }
    catch (Exception ex)
    {
        response.Content = new StringContent("{Error:" + ex.Message + "}", Encoding.UTF8, "application/json");
    }
    return response;
}
```

4. Реалізуємо можливість оновлення вже існуючого запису у таблиці БД. Для цього додамо у наш контролер екшн типу POST з назвою UpdateEmp який на вхід буде приймати об'єкт Employee. Цей екшн матиме наступний вигляд:

У разі вдалого оновлення існуючої сутності співробітника, екшн поверне оновлену сутність співробітника у форматі JSON, у разі помилки буде повернено текст помилки у форматі JSON.

5. Реалізуємо можливість видалення існуючого запису з таблиці БД. Для цього додамо у наш контролер екшн типу POST з назвою DeleteEmp який на вхід буде приймати об'єкт Employee. Цей екшн матиме наступний вигляд:

```
[HttpPost]
[ActionName("DeleteEmp")]
public HttpResponseMessage DeleteEmp(Employee sEmp)
{
    var response = Request.CreateResponse(HttpStatusCode.OK);

    var emp = (from o in db.Employees where o.IdEmployee == sEmp.IdEmployee select o).First();

    try
    {
        db.Employees.Remove(emp);
        db.SaveChanges();
        response.Content = new StringContent("{Id:" + emp.IdEmployee + ",Name:" + emp.Name + ",Age:" + emp.Age + "}", Encoding.UTF8, "application/json");
    }
    catch (Exception ex)
    {
        response.Content = new StringContent("{Error:" + ex.Message + "}", Encoding.UTF8, "application/json");
    }
    return response;
}
```

У разі вдалого видалення сутності співробітника, екшн поверне видалену сутність співробітника у форматі JSON, у разі помилки буде повернено текст помилки у форматі JSON.

Після завершення внесення змін до нашого API контролеру необхідно виконати білд проекту та запустити його.

6. Для перевірки коректності роботи нашого Web-додатку варто скористатись плагіном Postman для зручної відправки POST запитів, та отримання результатів їх виконання.

Після запуску проекту, запусимо пагін Postman та зробимо GET запит на отримання переліку співробітників:

http://localhost:57119/api/WApi/GetEmp/ GET

Send Preview Add to collection

body Cookies (3) Headers (10) STATUS 200 OK TIME 3505 ms

Pretty Raw Preview

```
[{"Id":0,"Name":"Phil"}, {"Id":1,"Name":"Peter"}, {"Id":2,"Name":"Kate"}, {"Id":3,"Name":"Paula"}]
```

Отриманий результат є коректним, екшен працює правильно.

Зробимо POST запит на створення нової сутності співробітника:

http://localhost:57119/api/WApi/CreateEmp/ POST

form-data x-www-form-urlencoded raw

IdEmployee	4	✕
Name	Jimbo	✕
Key	Value	

Send Preview Add to collection

body Cookies (3) Headers (10) STATUS 200 OK TIME 101 ms

Pretty Raw Preview JSON XML

```
1 {Id:4, Name: Jimbo, Age: 0}
```

Отриманий результат є коректним, екшен працює правильно.

Зробимо POST запит на оновлення вже існуючої сутності співробітника:



http://localhost:57119/api/WApi/UpdateEmp POST

form-data x-www-form-urlencoded raw

IdEmployee	4	✕
Name	Jimbo	✕
Age	25	✕
Key	Value	

Send Preview Add to collection

body Cookies (3) Headers (10) STATUS 200 OK TIME 32 ms

Pretty Raw Preview JSON XML

1 {Id:4,Name:Jimbo,Age:25}

Отриманий результат є коректним, екшен працює правильно.

Зробимо POST запит на видалення існуючої сутності співробітника:

http://localhost:57119/api/WApi/DeleteEmp POST

form-data x-www-form-urlencoded raw

IdEmployee	3	✕
Name	Paula	✕
Key	Value	

Send Preview Add to collection

body Cookies (3) Headers (10) STATUS 200 OK TIME 30 ms

Pretty Raw Preview JSON XML

1 {Id:3,Name:Paula,Age:0}

Отриманий результат є коректним, екшен працює правильно.

Зробимо GET запит на отримання переліку співробітників, щоб переконатися у внесенні змін до нашої БД:



Всі екшени відпрацювали коректно, додаток працює коректно.

### Вимоги до виконання роботи:

- Реалізувати додаток згідно варіанту
- Підготувати звіт до програми, який буде містити:
  - зміст завдання
  - текст додатку з коментарями
  - результати перевірки коректності роботи додатку на різних значеннях параметрів (у тому числі і помилкових)
  - формулювання призначення моделі, представлення та контролера.

### Завдання:

Реалізувати прошарок Арі для додавання, оновлення, видалення та отримання даних з БД і повернення статусу виконаної операції у форматі JSON.