بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

استفاده از مدل برنامه نویسی ناهمزمان و رویه های ذخیره شده با EF در MVC

مدرس : مهندس افشین رفوآ

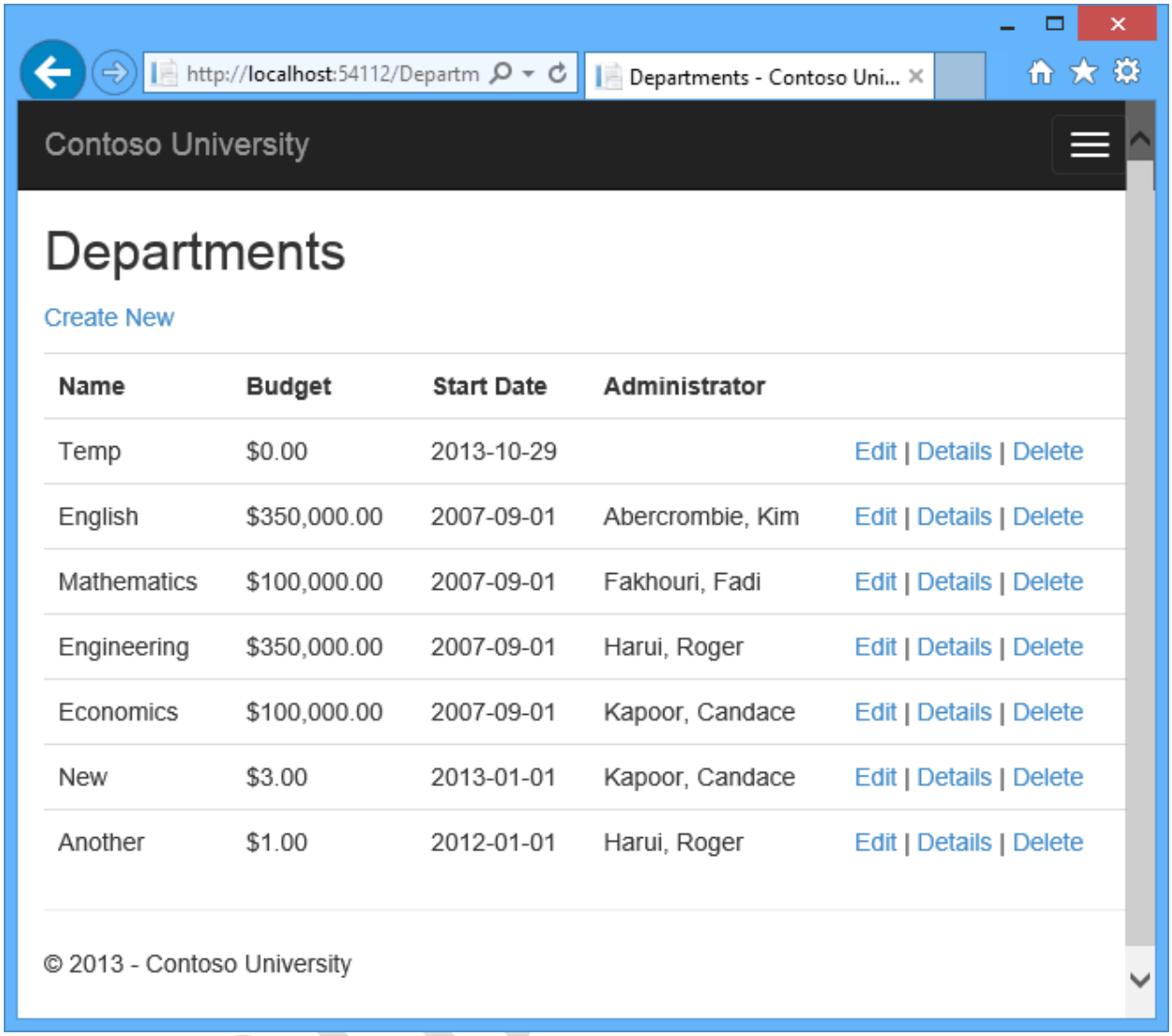
دوره آموزش MVC

استفاده از مدل برنامه نویسی ناهمزمان و رویه های ذخیره شده با EF در یک برنامه ی تحت وب ASP.NET MVC

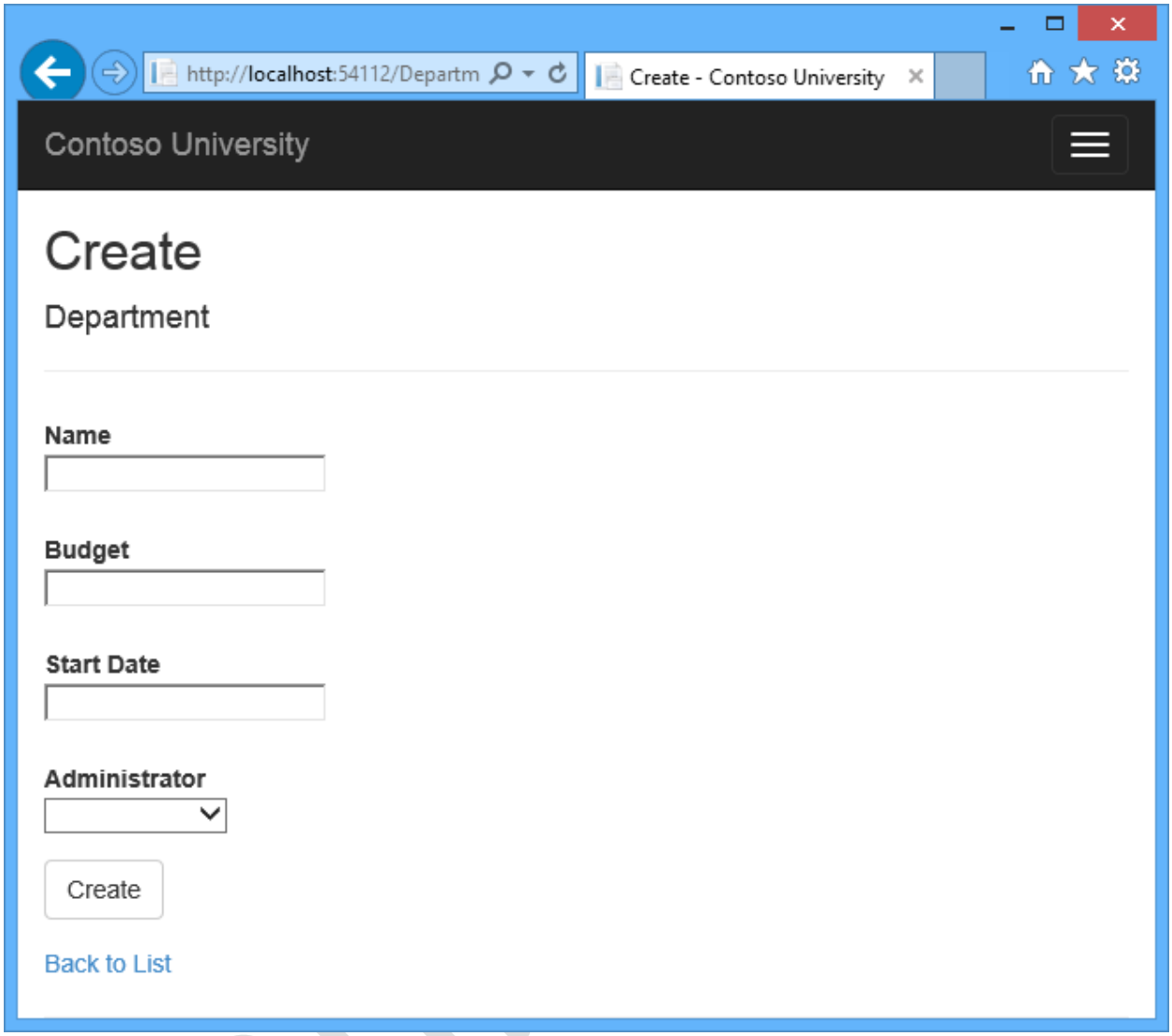
در مباحث قبلی با نحوه ی خواندن و بروز رسانی داده ها با استفاده از مدل برنامه نویسی هزمان (synch programming model) آشنا شدید. در این آموزش نحوه ی پیاده سازی مدل برنامه نویسی ناهمزمان را فراخواهید گرفت. Asynchronous code با استفاده ی بهینه از منابع سرور، کارایی برنامه را بالا می برد. در مقاله ی حاضر، چگونگی استفاده از stored procedure ها برای اجرای عملیات insert، update و delete بر روی یک موجودیت را خواهید آموخت.

در زیر تصاویر برخی از صفحاتی را که با آن ها کار خواهید کرد، مشاهده می کنید:

Tahlildaden


http://localhost:54112/Departm
Departments - Contoso Uni...
Contoso University
Departments
Create New
Name
Budget
Start Date
Administrator
Temp
$0.00
2013-10-29
Edit | Details | Delete
English
$350,000.00
2007-09-01
Abercrombie, Kim
Edit | Details | Delete
Mathematics
$100,000.00
2007-09-01
Fakhouri, Fadi
Edit | Details | Delete
Engineering
$350,000.00
2007-09-01
Harui, Roger
Edit | Details | Delete
Economics
$100,000.00
2007-09-01
Kapoor, Candace
Edit | Details | Delete
New
$3.00
2013-01-01
Kapoor, Candace
Edit | Details | Delete
Another
$1.00
2012-01-01
Harui, Roger
Edit | Details | Delete
© 2013 - Contoso University

Tar



چرا استفاده از asynchronous code توصیه می شود

تعداد thread هایی که در دسترس یک سرور وب قرار دارد محدود است و در مواقعی که بارگذاری سنگین در حال اجرا است، تمامی thread ها بکار گرفته می شوند. در صورت به وجود آمدن چنین شرایطی، سرویس دهنده تا زمانی که تمامی thread ها آزاد نشده اند، قادر به پردازش درخواست ها نخواهد بود. اما در خصوص synchronous code، بسیاری از thread ها با اینکه هیچ کار یا عملیات خاصی را انجام نمی دهند، باز مشغول بوده و قابل دسترس نمی باشند. دلیلش این است که thread ها منتظر هستند که عملیات ورودی/خروجی به اتمام برسد. در رابطه با asynchronous code، هنگامی که فرایندی منتظر اتمام عملیات ورودی/خروجی می

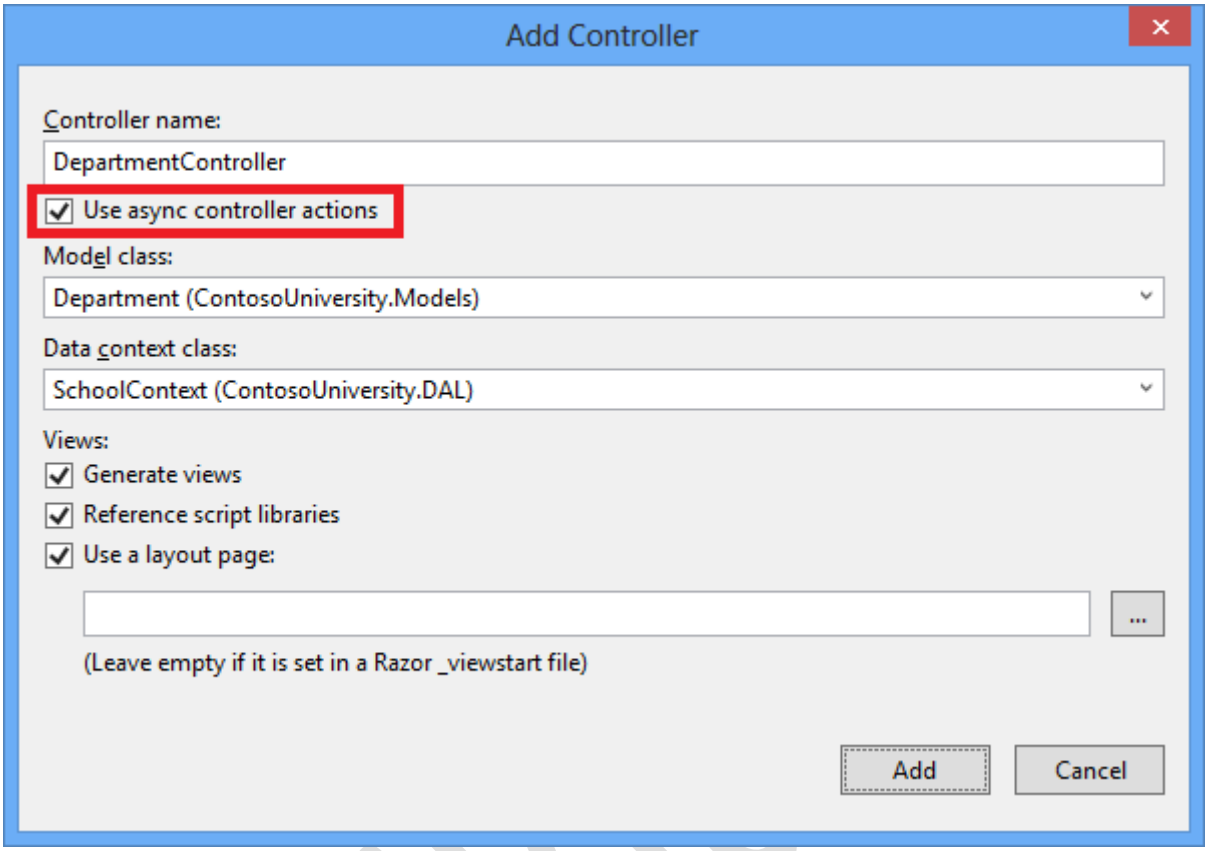
باشد، thread آن آزاد شده تا برای پردازش دیگر درخواست ها توسط سرور مورد استفاده قرار گیرد. در نتیجه، asynchronous code امکان و زمینه ی استفاده ی بهینه از منابع سرور را مهیا ساخته و همچنین سرور را قادر می سازد تا بدون هیچ گونه تاخیر ترافیک بیشتری را مدیریت کند.

در نسخه های قدیمی تر .NET.، کدنویسی و تست آن بسیار پیچیده، مستعد خطا بوده و همچنین خطایابی (debug) آن بسیار دشوار می باشد. در ویرایش .NET 4.5 کدنویسی، تست و اشکال زدایی آن به مراتب آسان تر می باشد، از این رو پیشنهاد می کنیم تا حد امکان کدهای ناهمزمان (asynchronous) بنویسید. اگرچه نوشتن کدهای ناهمزمان باعث ورود مقداری سربار (overhead) می شود، در مواقع کم ترافیک، افت کارایی بسیار ناچیز بوده و مشکل بزرگی رخ نمی دهد. این در حالی است که در شرایطی که ترافیک بالا است، افزایش بالقوه ی کارایی چشمگیر خواهد بود.

ایجاد controller ای به نام Department

یک Controller به نام Department مانند نمونه های قبلی ایجاد کنید، اما برای این نمونه checkbox، Use async controller actions را تیک دار نمایید.

Tahlildadeh



بخش های رنگی شده در کد زیر، نشان می دهد چه چیزهایی به synchronous code متد Index افزوده شده که آن را ناهمزمان (asynchronous) می کند:

```
public async Task<ActionResult> Index()
    {
      var departments = db.Departments.Include(d => d.Administrator);
      return View(await departments.ToListAsync());
    }
```

چهار تغییر به کد بالا اعمال شده که به query امکان می دهد به صورت ناهمزمان اجرا گردد:

1. متد مورد نظر با کلیدواژه ی asynch علامت گذاری شده که به مترجم یا کامپایلر می فهماند که باید callback هایی را برای بخش های بدنه ی متد ایجاد کرده و شی Task<ActionResult> که برگردانده می شود را به صورت خودکار ایجاد کند.

2. نوع (type) بازگشتی از ActionResult به Task<ActionResult> تبدیل شده است. نوع Task<T> نشانگر عملیات یا کارهای در حال انجام با نتیجه ی type T می باشد.

کلیدواژه ی await به فراخوانی web service اعمال شده است. کامپایلر با دیدن این کلیدواژه، در پشت پرده متد مورد نظر را به دو بخش تقسیم می کند. اولین بخش آن متد با عملیاتی که به صورت ناهمزمان راه اندازی (آغاز) شده پایان می یابد و اما دومین بخش آن در یک متد callback قرار داده می شود که پس از اتمام عملیات، فراخوانی می شود.

3. نسخه ی ناهمزمان (asynchronous) متد الحاقی (extension method) به نام ToList صدا زده شده است.

چرا دستور departments.ToList اصلاح شده اما دستور departments = db.Departments مورد تغییر قرار نگرفته است؟ باید گفت دلیلش این است که تنها آن دستوراتی که باعث می شوند query یا command به پایگاه داده ارسال شود، به صورت ناهمزمان اجرا می شوند. دستور departments = db.Departments یک query تنظیم می کند، اما query تا زمانی که متد ToList فراخوانی نشده، اجرا نمی شود. بنابراین تنها متد ToList به صورت ناهمزمان اجرا می شود.

از میان متدهای Details و همچنین توابع HttpGet Edit و Delete، متد Find است که باعث می شود query به پایگاه داده ارسال شود، از این رو متدی که به صورت ناهمزمان اجرا می شود، فقط متد Find است.

```
public async Task<ActionResult> Details(int? id)
    {
      if (id == null)
      {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
      }
      Department department = await db.Departments.FindAsync(id);
      if (department == null)
      {
        return HttpNotFound();
      }
      return View(department);
    }
```

در متدهای Create، HttpPost Edit و DeleteConfirmed، این فراخوانی متد SaveChanges است که سبب می شود یک دستور اجرا شود، نه دستورهایی نظیر db.Departments.Add(department) که تنها منجر به اصلاح موجودیت ها در حافظه می شوند.

```
    public async Task<ActionResult> Create(Department department)
{
```

```
  if (ModelState.IsValid)
  {
    db.Departments.Add(department);
  await db.SaveChangesAsync();
    return RedirectToAction("Index");
  }
```

Views\Department\Index.cshtml را باز کرده و template code را با کد زیر جایگزین کنید:

```
@model IEnumerable<ContosoUniversity.Models.Department>
@{
  ViewBag.Title = "Departments";
}
<h2>Departments</h2>
<p>
  @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
  <tr>
    <th>
      @Html.DisplayNameFor(model => model.Name)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Budget)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.StartDate)
    </th>
    <th>
      Administrator
    </th>
    <th></th>
  </tr>
  @foreach (var item in Model)
  {
    <tr>
      <td>
        @Html.DisplayFor(modelItem => item.Name)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.Budget)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.StartDate)
      </td>
      <td>
        @Html.DisplayFor(modelItem => item.Administrator.FullName)
      </td>
      <td>
        @Html.ActionLink("Edit", "Edit", new { id = item.DepartmentID }) |
        @Html.ActionLink("Details", "Details", new { id = item.DepartmentID }) |
```

Tahlildaden

```
            @Html.ActionLink("Delete", "Delete", new { id = item.DepartmentID })
        </td>
    </tr>
  }
</table>
```

این کد عنوان را از Index به Departments تغییر داده، اسم Administrator را به سمت راست انتقال می دهد و اسم کامل administrator را فراهم می کند.

در view های Edit، Details، Delete و Create عنوان (caption) فیلد InstructorID را به "Administrator" تغییر دهید، همان طوری که اسم فیلد department را در Course viewها به "Department" تغییر دادید.

در view های Create و Edit از کد زیر استفاده کنید:

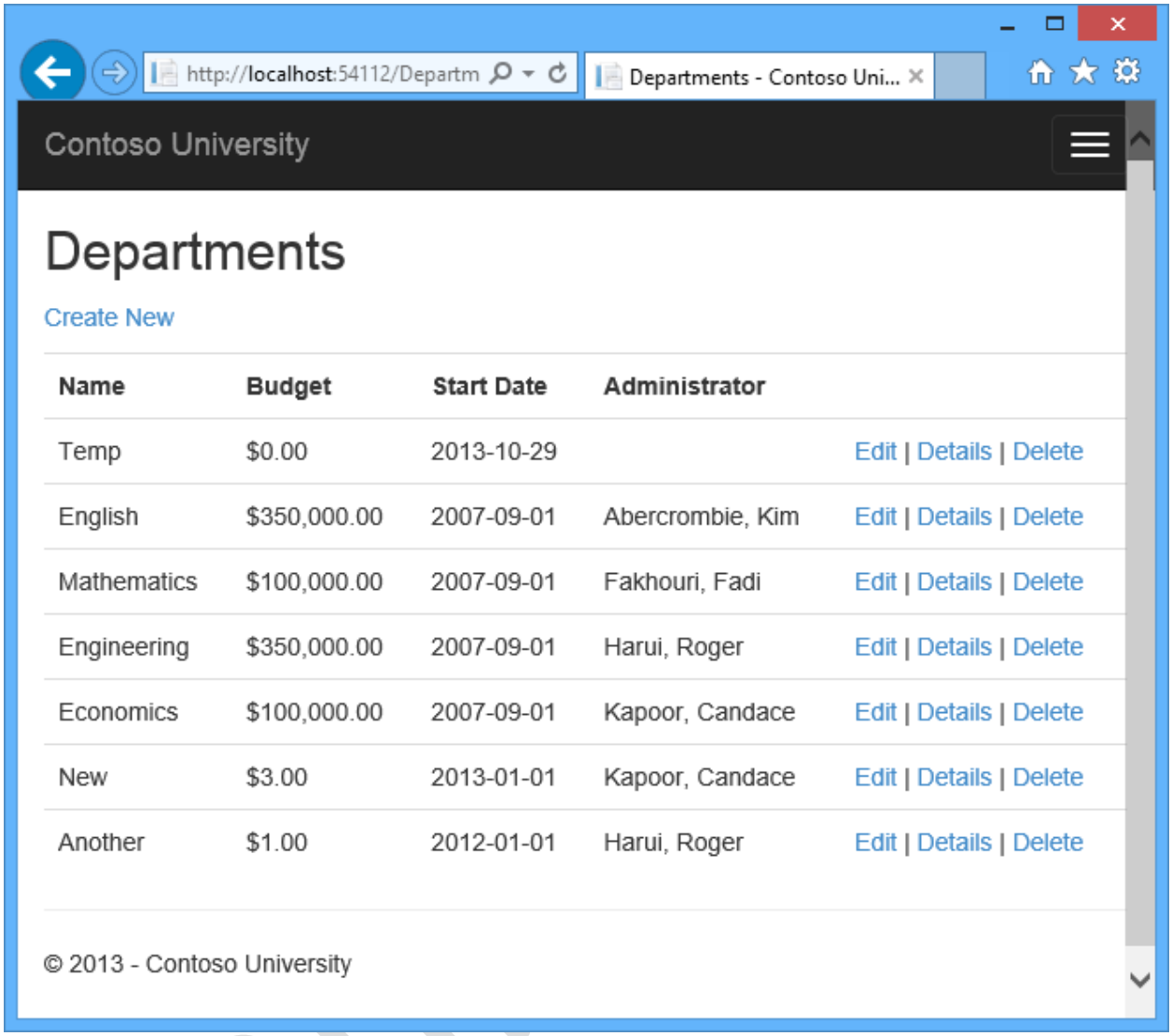
```
<label class="control-label col-md-2" for="InstructorID">Administrator</label>
```

در viewهای Create و Details کد زیر را بکار ببرید:

```
<dt>
  Administrator
</dt>
```

برنامه را اجرا کرده و تب Departments را کلیک کنید.

Tahlildadeh



همه چیز در این controller ها به طور مشابه عمل می کند، با این فرق که در این controller، تمامی queryها به صورت ناهمزمان (asynch) اجرا می شوند.

نکاتی که در استفاده از برنامه نویسی غیرموازی (asynch) با EF بایستی به آن توجه کرد:

1. استفاده از thread ها با async code توصیه نمی شود. به عبارتی دیگر، نبایست چندین عملیات را به طور موازی با استفاده از نمونه ی (context instance) یکسان انجام داد.

2. اگر می خواهید از مزایای بهبود کارایی async code استفاده ی بهینه داشته باشید، لازم است اطمینان حاصل کنید تمامی library package هایی که مورد استفاده قرار می دهید، در صورت

فراخوانی متدهای EF ای که باعث ارسال query ها به پایگاه داده می شوند، از async code استفاده کنند.

استفاده از Stored Procedure ها جهت درج، بروز آوری و حذف

برخی از DBA ها (مدیران پایگاه داده) ترجیح می دهند از stored procedure ها (رویه های ذخیره شده) برای دسترسی به پایگاه داده استفاده کنند. در ویرایش های پیشین EF می توانستید داده های مورد نیاز را با بکارگیری stored procedure ها بازیابی کنید. این کار به وسیله ی اجرای یک query خام صورت می گرفت. اما این امکان وجود نداشت که به EF دستور داد با استفاده از stored procedure عملیات بروز رسانی را انجام دهد. در نسخه ی نوین EF (ویرایش 6)، به راحتی می توان Code First گونه ای پیکربندی کرد که از stored procedure ها استفاده کند.

1. در فایل DAL\SchoolContext.cs، کد هایلایت شده را به متد OnModelCreating اضافه کنید.

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
      modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();
      modelBuilder.Entity<Course>()
        .HasMany(c => c.Instructors).WithMany(i => i.Courses)
        .Map(t => t.MapLeftKey("CourseID")
          .MapRightKey("InstructorID")
          .ToTable("CourseInstructor"));
      modelBuilder.Entity<Department>().MapToStoredProcedures();
    }
```

این کد به EF دستور می دهد با استفاده از stored procedure ها، عملیات درج، بروز رسانی و حذف را بر روی موجودیت Department انجام دهد.

2. در Package Manage Console، دستور زیر را وارد کنید:

```
add-migration DepartmentSP
```

فایل Migrations\<timestamp>_DepartmentSP.cs را باز کرده تا کد موجود در متد Up را مشاهده کنید. این متد stored procedure های Insert، Update و Delete را ایجاد می کند:

```
public override void Up()
    {
        CreateStoredProcedure(
    "dbo.Department_Insert",
    p => new
    {
        Name = p.String(maxLength: 50),
        Budget = p.Decimal(precision: 19, scale: 4, storeType: "money"),
        StartDate = p.DateTime(),
        InstructorID = p.Int(),
    },
    body:
        @"INSERT [dbo].[Department]([Name], [Budget], [StartDate], [InstructorID])
          VALUES (@Name, @Budget, @StartDate, @InstructorID)

          DECLARE @DepartmentID int
          SELECT @DepartmentID = [DepartmentID]
          FROM [dbo].[Department]
          WHERE @@ROWCOUNT > 0 AND [DepartmentID] = scope_identity()

          SELECT t0.[DepartmentID]
          FROM [dbo].[Department] AS t0
          WHERE @@ROWCOUNT > 0 AND t0.[DepartmentID] = @DepartmentID"
    );

        CreateStoredProcedure(
    "dbo.Department_Update",
    p => new
    {
        DepartmentID = p.Int(),
        Name = p.String(maxLength: 50),
        Budget = p.Decimal(precision: 19, scale: 4, storeType: "money"),
        StartDate = p.DateTime(),
        InstructorID = p.Int(),
    },
    body:
        @"UPDATE [dbo].[Department]
          SET [Name] = @Name, [Budget] = @Budget, [StartDate] = @StartDate, [InstructorID] = @InstructorID
          WHERE ([DepartmentID] = @DepartmentID)"
    );

        CreateStoredProcedure(
    "dbo.Department_Delete",
    p => new
    {
        DepartmentID = p.Int(),
    },
    body:
        @"DELETE [dbo].[Department]
          WHERE ([DepartmentID] = @DepartmentID)"
```

T.ahlaadeh

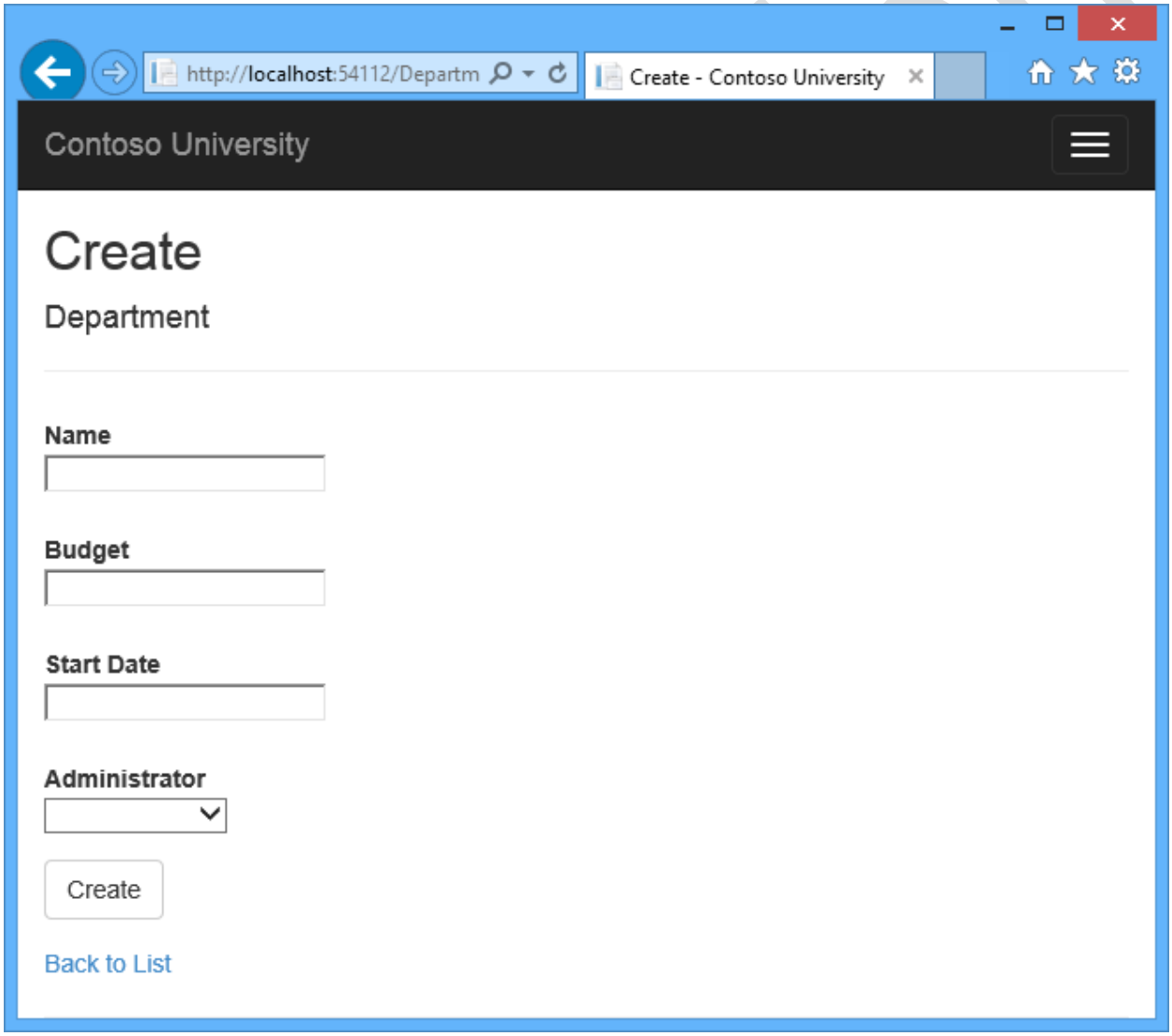
```
);

        }
```

3. در Package Manage Console، دستور زیر را وارد نمایید:

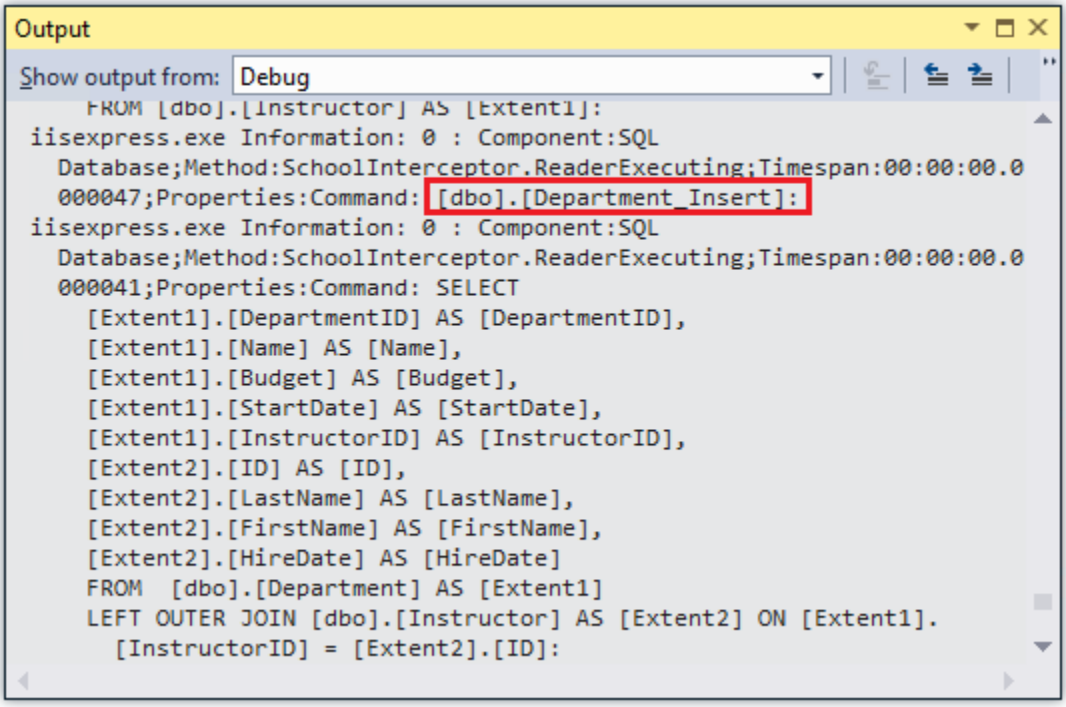
```
update-database
```

4. برنامه را در debug mode اجرا کرده، تب Departments را باز کنید، سپس Create New را کلیک نمایید.

5. اطلاعات لازم برای یک department جدید را وارد کرده و Create را کلیک نمایید.



6. در محیط Visual Studio، اگر به گزارشات (log) در پنجره ی Output توجه کنید، می بینید که یک stored procedure باعث درج یک سطر Department شده است.



Code First اسم های پیش فرض برای stored procedure ها ایجاد می کند. در صورت استفاده از یک پایگاه داده ی از پیش موجود، ممکن است لازم باشد اسم stored procedure ها را سفارشی تنظیم کنید تا بتوانید از آن stored procedure هایی که قبلا در پایگاه داده تعریف شده اند، استفاده نمایید.

اگر بخواهید کارهایی را که stored procedure های ایجاد شده قادر به انجام آن ها هستند، تنظیم نمایید، در آن صورت بایستی کد ارائه شده توسط scaffolding برای متد migrations Up را که آن stored procedure را ایجاد می کند، ویرایش کنید. در آن صورت تمام تغییرات ایجاد شده توسط شما، هر زمانی که آن migration اجرا می شود منعکس شده و همچنین هنگامی که migrations به صورت خودکار در production پس از نصب (deployment) اجرا می شود، به production database اعمال می گردد.

اگر می خواهید Stored procedure موجود را که قبلا در یک migration ایجاد شده، تغییر دهید، در آن صورت می بایست با استفاده از دستور Add-Migration یک migration خالی ایجاد کرده، سپس کدی بنویسید که متد AlterStoredProcedure را فراخوانی کند.

آنچه در این مبحث انجام شد:

در این درس با نوشتن کدی که به صورت ناهمزمان اجرا می شود، کارایی سرور را افزایش داده و همچنین با نحوه ی استفاده از stored procedure ها برای اجرای عملیات بروز آوری، درج و حذف آشنا شدید. در فصل بعدی خواهید آموخت زمانی که چند کاربر سعی می کنند سطری یکسان را همزمان ویرایش کنند، چگونه می توان از از دست رفت داده جلوگیری کرد.

Tahlildaden