ASP.NET WEB API 2.1

EF Core

Content

- Querying Data
- Saving Data

Querying Data

- Basic Queries
- LINQ Extension Methods
- Loading Related Data
- Tracking vs. No-Tracking Queries
- Raw SQL Queries
- Asynchronous Queries
- 101 LINQ Examples

Basic Queries

- Loading all data

```
using (var context = new BloggingContext())
{
    var blogs = context.Blogs.ToList();
}
```

- Loading a single entity

```
using (var context = new BloggingContext())
{
    var blog = context.Blogs
        .Single(b => b.BlogId == 1);
}
```

- Filtering

```
using (var context = new BloggingContext())
{
    var blogs = context.Blogs
        .Where(b => b.Url.Contains("dotnet"))
        .ToList();
}
```

LINQ Extension Methods

- First()
- FirstOrDefault()
- Single()
- SingleOrDefault()
- ToList()
- Count()
- Min()
- Max()
- Sum()
- Last()
- LastOrDefault()
- Average()
- Find()
- GroupBy()
- OrderBy()
- Distinct()

Loading Related Data

```
using (var context = new BloggingContext())
{
    var blogs = context.Blogs
        .Include(blog => blog.Posts)
        .ToList();
}
```

```
using (var context = new BloggingContext())
{
    var blogs = context.Blogs
        .Include(blog => blog.Posts)
        .Include(blog => blog.Owner)
        .ToList();
}
```

```
using (var context = new BloggingContext())
{
    var blogs = context.Blogs
        .Include(blog => blog.Posts)
            .ThenInclude(post => post.Author)
        .ToList();
}
```

Tracking vs. No-Tracking Queries

- Tracking queries

```
using (var context = new BloggingContext())
{
    var blog = context.Blogs.SingleOrDefault(b => b.BlogId == 1);
    blog.Rating = 5;
    context.SaveChanges();
}
```

- No-tracking queries

```
using (var context = new BloggingContext())
{
    var blogs = context.Blogs
        .AsNoTracking()
        .ToList();
}
```

Raw SQL Queries

- Limitations
- Basic raw SQL queries
- Passing parameters
- Composing with LINQ

Limitations

- The SQL query must return data for all properties of the entity or query type.
- The column names in the result set must match the column names that properties are mapped to.
- The SQL query cannot contain related data.
- SELECT statements passed to this method should generally be composable.
- SQL statements other than SELECT are recognized automatically as non-composable

Basic raw SQL queries

```
var blogs = context.Blogs
    .FromSql("SELECT * FROM dbo.Blogs")
    .ToList();
```

```
var blogs = context.Blogs
    .FromSql("EXECUTE dbo.GetMostPopularBlogs")
    .ToList();
```

Passing parameters

```
var user = "johndoe";

var blogs = context.Blogs
    .FromSql("EXECUTE dbo.GetMostPopularBlogsForUser {0}", user)
    .ToList();
```

```
var user = "johndoe";

var blogs = context.Blogs
    .FromSql($"EXECUTE dbo.GetMostPopularBlogsForUser {user}")
    .ToList();
```

```
var user = new SqlParameter("user", "johndoe");

var blogs = context.Blogs
    .FromSql("EXECUTE dbo.GetMostPopularBlogsForUser @user", user)
    .ToList();
```

Composing with LINQ

```
var searchTerm = ".NET";

var blogs = context.Blogs
    .FromSql($"SELECT * FROM dbo.SearchBlogs({searchTerm})")
    .Where(b => b.Rating > 3)
    .OrderByDescending(b => b.Rating)
    .ToList();
```

```
var searchTerm = ".NET";

var blogs = context.Blogs
    .FromSql($"SELECT * FROM dbo.SearchBlogs({searchTerm})")
    .Include(b => b.Posts)
    .ToList();
```

101 LINQ Examples

- https://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b

Saving Data

- Basic Save
- Custom Update
- Using Transactions
- Asynchronous Saving

Basic Save

- Adding Data

```
using (var context = new BloggingContext())
{
    var blog = new Blog { Url = "http://sample.com" };
    context.Blogs.Add(blog);
    context.SaveChanges();
}
```

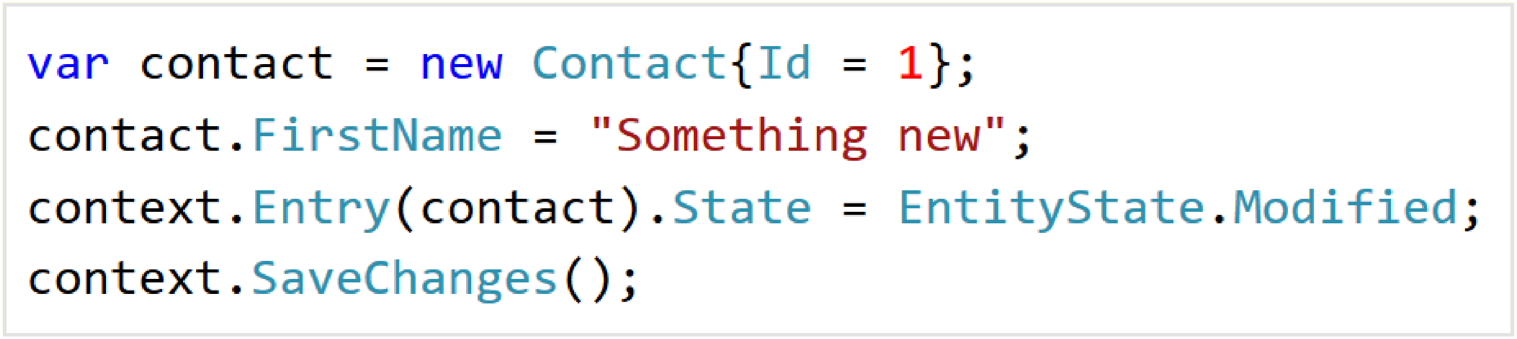
- Updating Data

```
using (var context = new BloggingContext())
{
    var blog = context.Blogs.First();
    blog.Url = "http://sample.com/blog";
    context.SaveChanges();
}
```

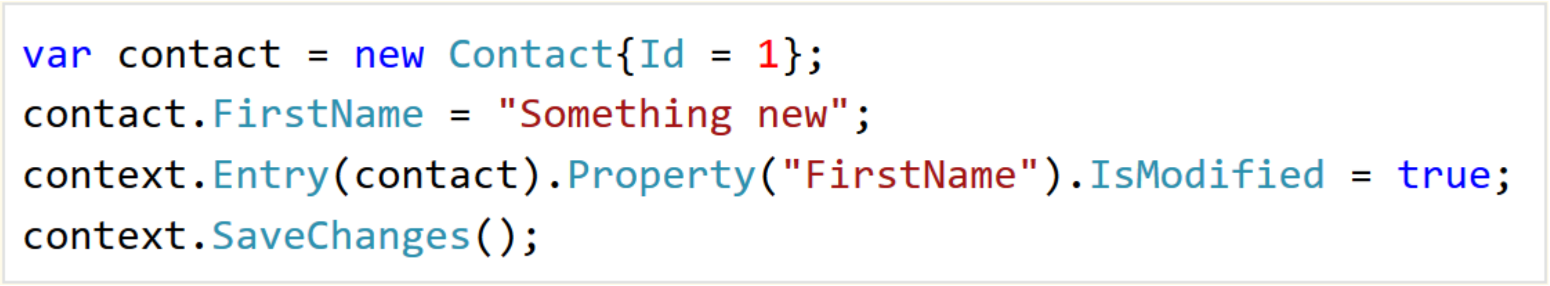
- Deleting Data

```
using (var context = new BloggingContext())
{
    var blog = context.Blogs.First();
    context.Blogs.Remove(blog);
    context.SaveChanges();
}
```

Custom Update



```csharp
var contact = new Contact{Id = 1};
contact.FirstName = "Something new";
context.Entry(contact).State = EntityState.Modified;
context.SaveChanges();
```



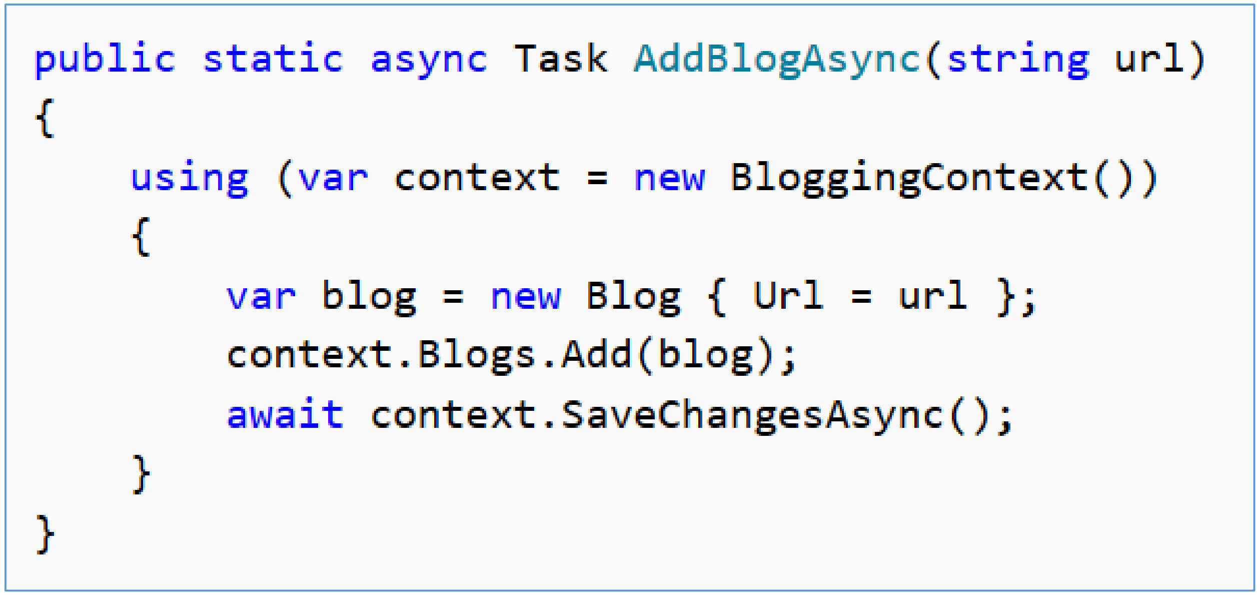
```csharp
var contact = new Contact{Id = 1};
contact.FirstName = "Something new";
context.Entry(contact).Property("FirstName").IsModified = true;
context.SaveChanges();
```

Using Transactions

```
using (var context = new BloggingContext())
{
    using (var transaction = context.Database.BeginTransaction())
    {
        try
        {
            context.Blogs.Add(new Blog { Url = "http://blogs.msdn.com/dotnet" });
            context.SaveChanges();

            context.Blogs.Add(new Blog { Url = "http://blogs.msdn.com/visualstudio" });
            context.SaveChanges();

            var blogs = context.Blogs
                .OrderBy(b => b.Url)
                .ToList();

            // Commit transaction if all commands succeed, transaction will auto-rollback
            // when disposed if either commands fails
            transaction.Commit();
        }
        catch (Exception)
        {
            // TODO: Handle failure
        }
    }
}
```

Asynchronous Saving

Asynchronous saving avoids blocking a thread while the changes are written to the database.



```
public static async Task AddBlogAsync(string url)
{
    using (var context = new BloggingContext())
    {
        var blog = new Blog { Url = url };
        context.Blogs.Add(blog);
        await context.SaveChangesAsync();
    }
}
```

THE END