Telerik Academy

Entity Framework Performance

Databases

Telerik Software Academy
http://academy.telerik.com

Telerik Academy

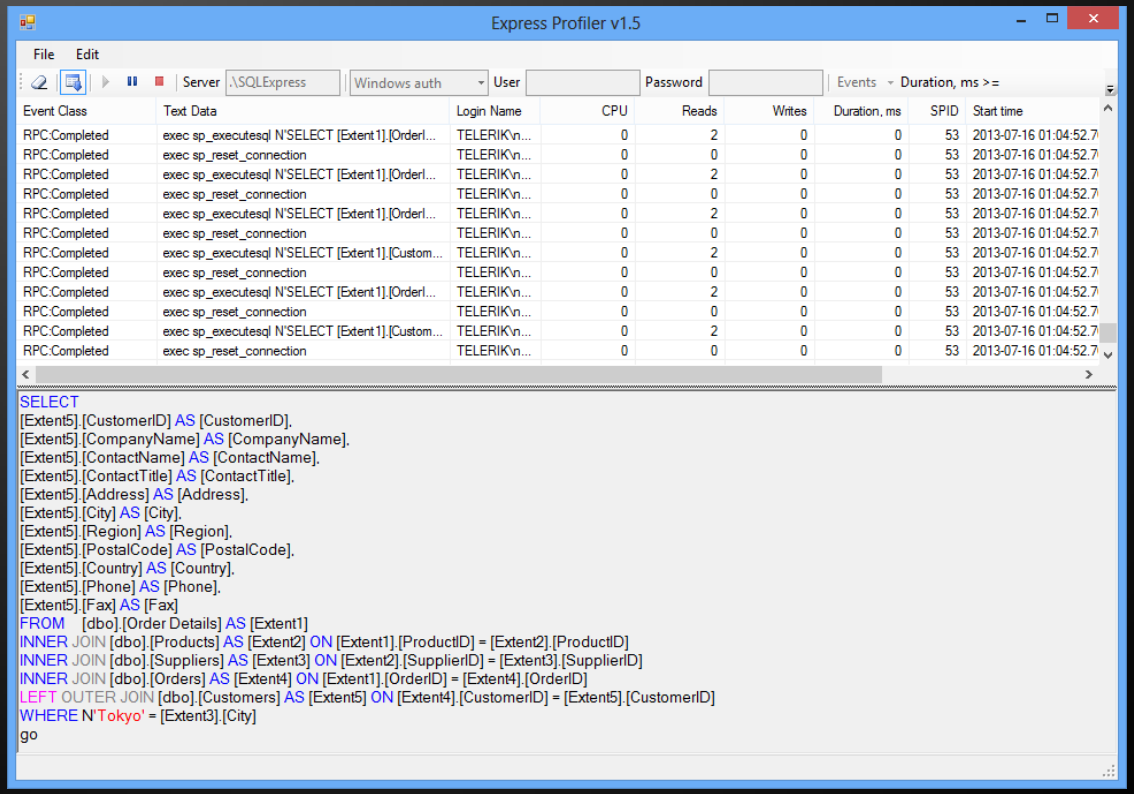
Table of Contents

- SQL Profilers
- The N+1 Query Problem
- Incorrect Use of ToList()
- Incorrect use of SELECT *
- Deleting objects faster with native SQL

Telerik Academy

SQL Profilers

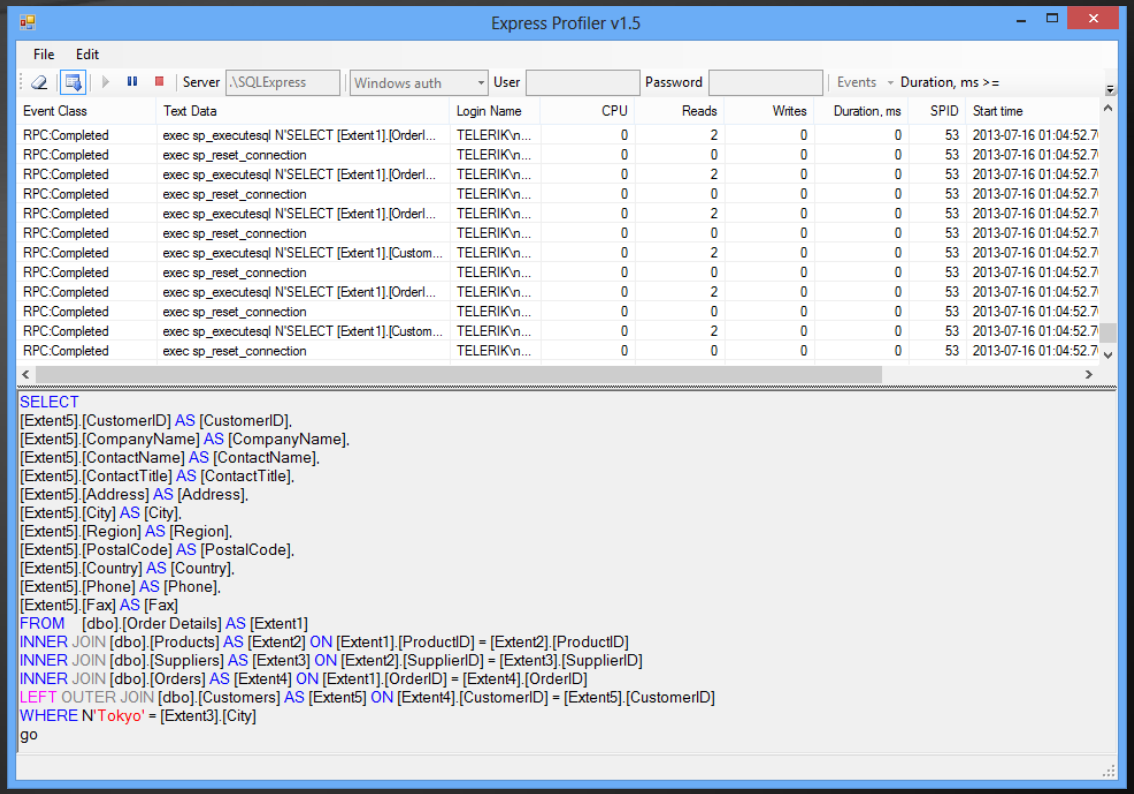
How to Trace All Executed SQL Commands?



What is SQL Profiler?

- SQL Profilers intercept the SQL queries executed at the server side
  - Powerful tools to diagnose the hidden Entity Framework queries
- SQL Server has "SQL Server Profiler" tool
  - Part of Enterprise / Developer edition (paid tool)
- A free SQL Profiler exists for SQL Server:
  - Express Profiler: http://expressprofiler.codeplex.com
  - Easy-to-use, open-source, lightweight, powerful, … and works!

Telerik Academy



Express Profiler

Live Demo

Telerik Academy

The N+1 Query Problem

What is the N+1 Query Problem and How to Avoid It?

The N+1 Query Problem

- What is the N+1 Query Problem?
  - Imagine a database that contains tables `Products`, `Suppliers` and `Categories`
    - Each product has a supplier and a category
  - We want to print each `Product` along with its `Supplier` and `Category`:

```
foreach (var product in context.Products)
{
  Console.WriteLine("Product: {0}; {1}; {2}",
    product.ProductName, product.Supplier.CompanyName,
    product.Category.CategoryName);
}
```

The N+1 Query Problem (2)

- This code will execute N+1 SQL queries:

```
foreach (var product in context.Products)
{
  Console.WriteLine("Product: {0}; {1}; {2}",
    product.ProductName, product.Supplier.CompanyName,
    product.Category.CategoryName);
}
```

One query to retrive the products

Additional N queries to retrieve the supplier for each product

Additional N queries to retrieve the category for each product

- Imagine we have 100 products in the database
  - That's ~ 201 SQL queries → very slow!
  - We could do the same with a single SQL query

Solution to the N+1 Query Problem

- Fortunately there is an easy way in EF to avoid the N+1 query problem:

Using `Include(…)` method only one SQL query with join is made to get the related entities

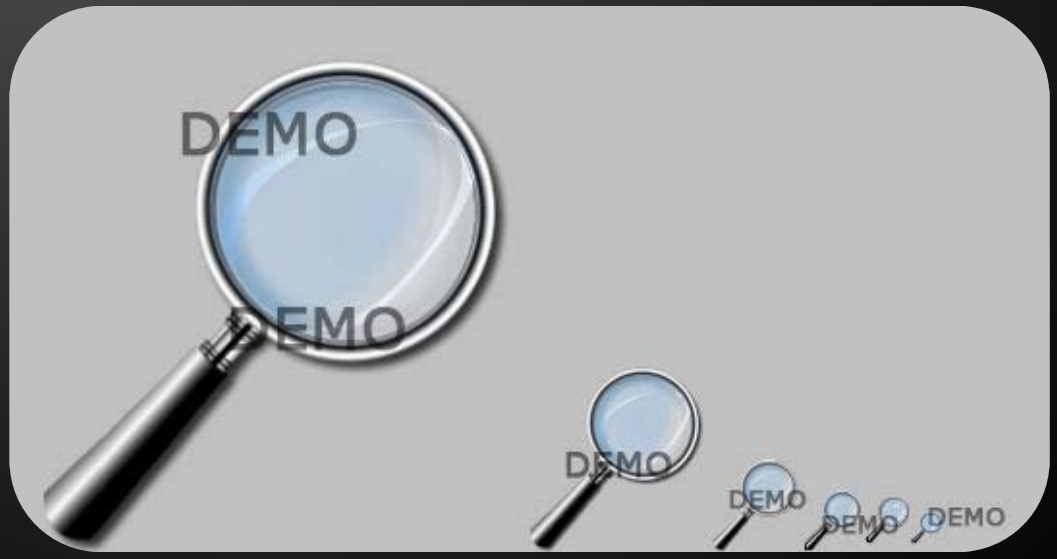
```
foreach (var product in context.Products.
  Include("Supplier").Include("Category"))
{
  Console.WriteLine("Product: {0}; {1}; {2}",
    product.ProductName, product.Supplier.CompanyName,
    product.Category.CategoryName);
}
```

No additional SQL queries are made here for the related entities

Telerik Academy

Solution to the N+1 Query Problem

Live Demo



Telerik Academy

Incorrect Use of ToList()

How ToList() Can Significantly Affect the Performance

Incorrect Use of ToList()

- In EF invoking `ToList()` executes the underlying SQL query in the database
  - Transforms `IQueryable<T>` to `List<T>`
  - Invoke `ToList()` as late as possible, after all filtering, joins and groupings
- Avoid such code:

```
List<Order_Detail> orderItemsFromTokyo =
  northwindEntities.Order_Details.ToList().
  Where(od => od.Product.Supplier.City == "Tokyo").ToList();
```

  - This will cause all order details to come from the database and to be filtered later in the memory

Incorrect Use of ToList()

Live Demo

Telerik Academy

Incorrect Use of SELECT *

Live Demo

Telerik Academy

Deleting Entities Faster with Native SQL Query

Deleting Entities

- Deleting entities (slower):

```
NorthwindEntities northwindEntities = new NorthwindEntities();
var category = northwindEntities.Categories.Find(46);
northwindEntities.Categories.Remove(category);
northwindEntities.SaveChanges();
```

  - Executes SELECT + DELETE commands

- Deleting entities with native SQL (faster):

```
NorthwindEntities northwindEntities = new NorthwindEntities();
northwindEntities.Database.ExecuteSqlCommand(
  "DELETE FROM Categories WHERE CategoryID = {0}", 46);
```

  - Executes a single DELETE command

Telerik Academy

Deleting Entities Faster with Native SQL Query

Live Demo

Telerik Academy

Entity Framework Performance

http://academy.telerik.com

Homework

1. Using Entity Framework write a SQL query to select all employees from the Telerik Academy database and later print their name, department and town. Try the both variants: with and without `.Include(…)`. Compare the number of executed SQL statements and the performance.
2. Using Entity Framework write a query that selects all employees from the Telerik Academy database, then invokes `ToList()`, then selects their addresses, then invokes `ToList()`, then selects their towns, then invokes `ToList()` and finally checks whether the town is "Sofia". Rewrite the same in more optimized way and compare the performance.

Telerik Academy

Free Trainings @ Telerik Academy

- C# Programming @ Telerik Academy
  - csharpfundamentals.telerik.com

- Telerik Software Academy
  - academy.telerik.com

- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy

- Telerik Software Academy Forums
  - forums.academy.telerik.com