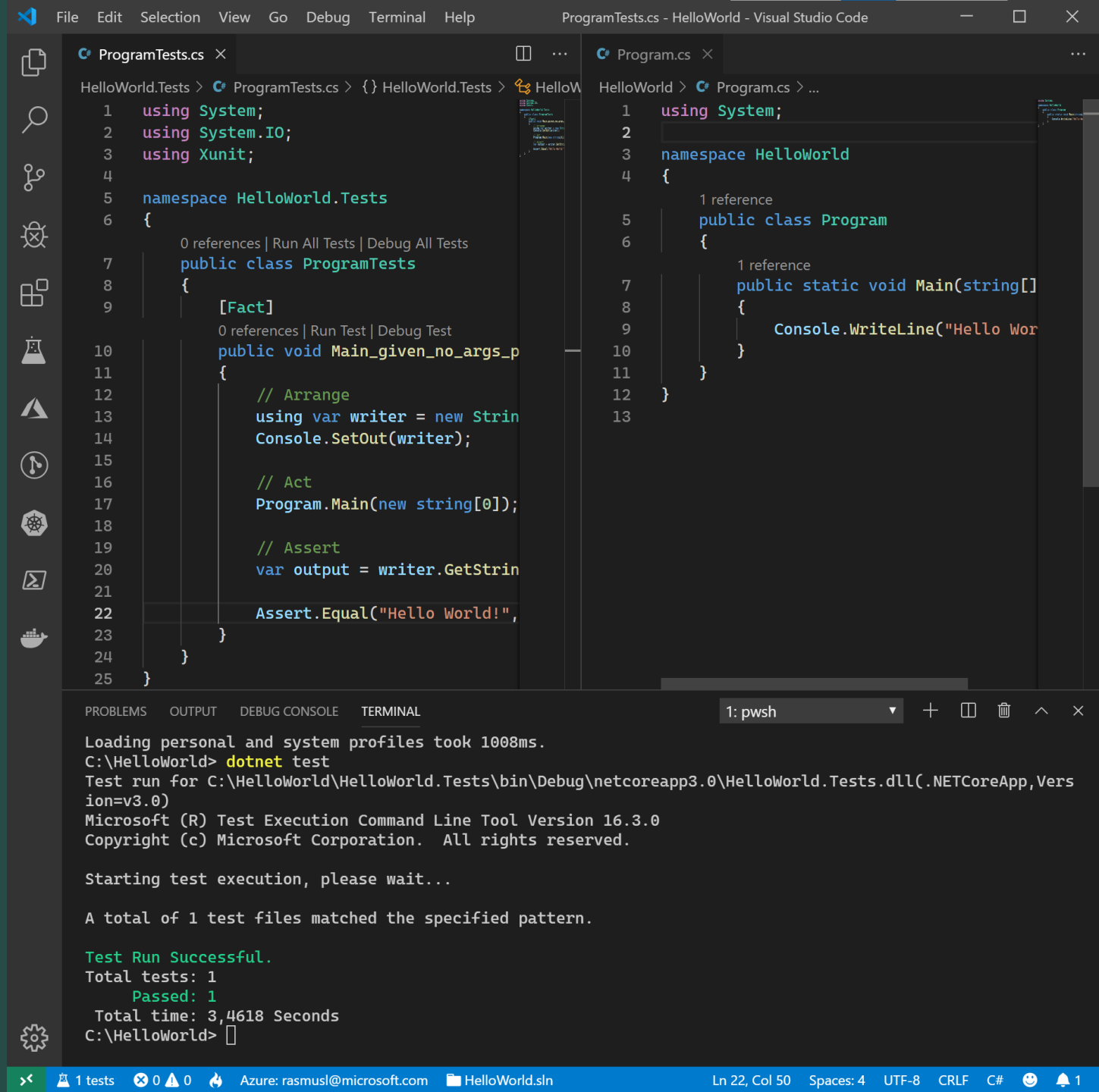
C#
Data Access and Entity Framework Core

Rasmus Lystrøm
Associate Professor
ITU
rnie@itu.dk



Agenda

Databases (SQL Server)

Secrets

Old school SQL

SQL Injection

The IDisposable interface

Entity Framework Core

Lazy vs. Eager Loading

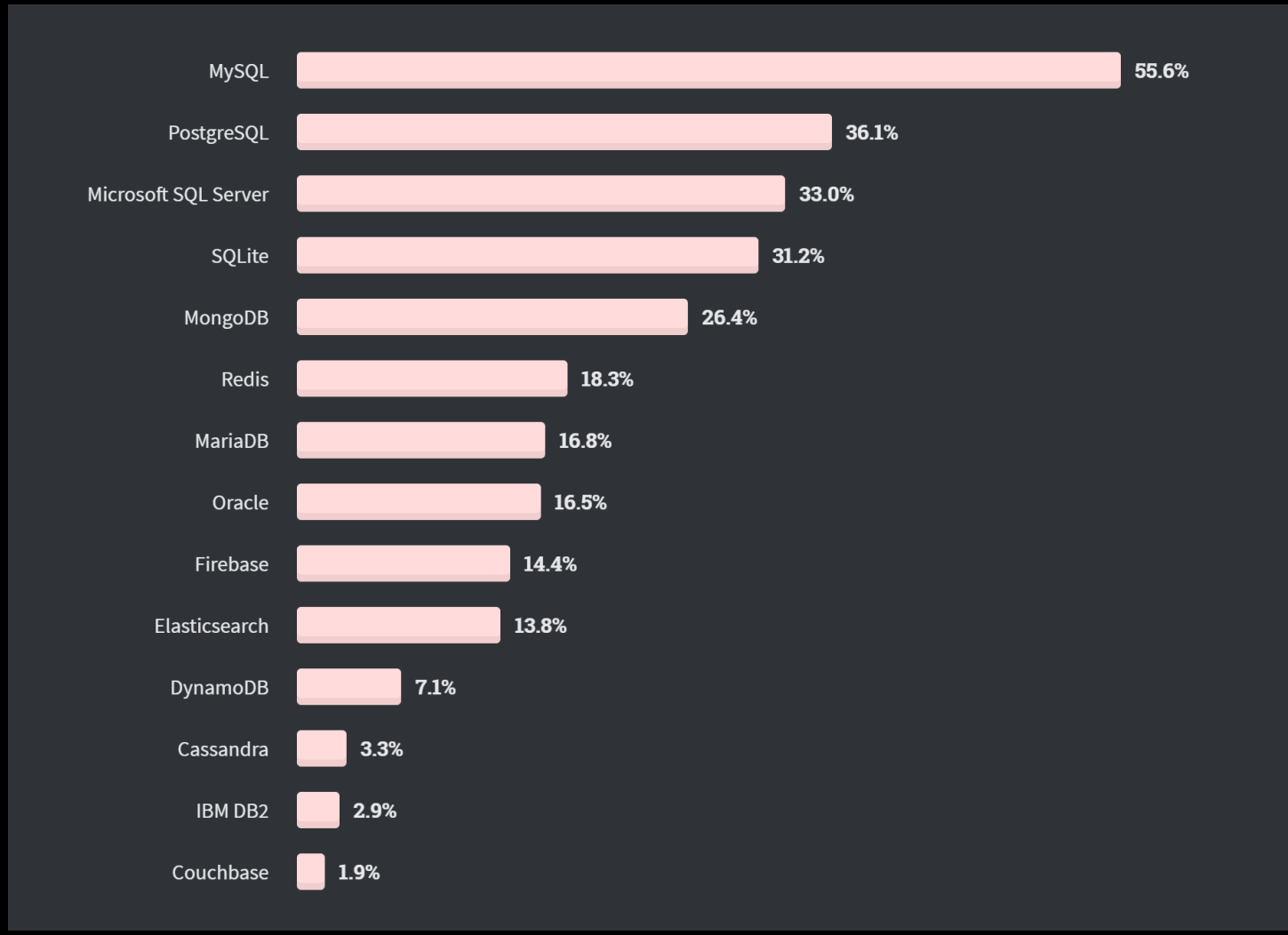
Databases

Databases

Relational (SQL)	Document (NoSQL)
Microsoft SQL Server	Azure Cosmos DB
Oracle Database	Amazon DynamoDB
IBM Db2	MongoDB
MySQL	Couchbase
MariaDB	Redis
PostgreSQL	Elasticsearch
SQLite	

Most popular databases

https://insights.stackoverflow.com/survey/2020#most-popular-technologies



SQL Server

Windows only: SQL Server Express LocalDB

https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-express-localdb

All (prefered): SQL Server in a container

Get docker:

https://docs.docker.com/get-docker/

Run container:

https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker

SQL Server

```
docker pull mcr.microsoft.com/mssql/server:2019-latest

$password = New-Guid

docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=$password" `
    -p 1433:1433 `
    -d mcr.microsoft.com/mssql/server:2019-latest
```

Microsoft

SQL Server Demo

SQL Server Docker Container

Secrets

Secrets

```
dotnet user-secrets init
dotnet user-secrets set "ConnectionStrings:ConnectionString" "..."
dotnet add package Microsoft.Extensions.Configuration.UserSecrets
```

Secrets

```
using Microsoft.Extensions.Configuration;

...

var configuration = new ConfigurationBuilder()
    .AddUserSecrets(typeof(Program).Assembly)
    .Build();

var connectionString = configuration.GetConnectionString("ConnectionString");
```

Old School SQL

Old School SQL

```
dotnet add package System.Data.SqlClient
```

```
var cmdText = "SELECT * FROM Animals";
using var connection = new SqlConnection(connectionString);
connection.Open();
using var command = new SqlCommand(cmdText, connection);
using var reader = command.ExecuteReader();
while (reader.Read())
{
    ...
}
```

SQL Injection

SQL Injection

A **SQL injection** attack consists of insertion or “injection” of a SQL query via the input data from the client to the application.

A successful SQL injection exploit can:

- read sensitive data from the database,
- modify database data (Insert/Update/Delete),
- execute administration operations on the database (such as shutdown the DBMS),
- or worse

https://owasp.org/www-community/attacks/SQL_Injection

IDisposable

IDisposable

```
try
{
    ...
}
finally
{
    resource.Dispose();
}
```

IDisposable

```
using (var resource1 = new Resource())
{
    ...
}
```

IDisposable

```
using var resource = new Resource();

...
```

Entity Framework Core

Entity Framework Core

```
dotnet tool install --global dotnet-ef
dotnet add package Microsoft.EntityFrameworkCore.Design
dotnet ef migrations add InitialCreate
dotnet ef database update
```

Lazy Loading

https://docs.microsoft.com/en-us/ef/core/querying/related-data/lazy

Lazy Loading

```
dotnet add package Microsoft.EntityFrameworkCore.Proxies

protected override void OnConfiguring2(DbContextOptionsBuilder optionsBuilder)
    => optionsBuilder.UseLazyLoadingProxies()
                     .UseSqlServer(...);
```