Översikt

- Läsa och skriva filer.
- Arbeta med Serializing och Deserializing.
- Arbeta med I/O genom Streams.

Lektion 1: Läsa och skriva filer

- Läsa och skriva information genom File Class.
- Manipulera fil.
- Manipulera katalog.
- Manipulera sökvägar.

Läsa och skriva information genom File Class

- Klassen File i System.IO ger dig tillgång till flera metoder som kan användas för att manipulera filer.
- Det finns metoder för att läsa:
  - ReadAllText, ReadAllLines och ReadAllBytes.
- Men även metoder för att skriva:
  - WriteAllText, WriteAllLine, WriteAllBytes, AppendAllText och AppendAllLine.

.eo/ec

Läsa och skriva information genom File Class (forts.)

- ReadAllText – läser in innehåll i fil till sträng.

```
string filePath = ”C:\\fourthCoffee\\settings.txt”;
string settings = File.ReadAllText(FilePath);
```

- ReadAllLines – läser rad för rad och lagrar detta i en area.

```
string filePath = ”C:\\fourthCoffee\\settings.txt”;
String[] settingsLineByLine = File.ReadAllLines(FilePath);
```

- ReadAllBytes – läser in filen i binär form och sparar denna i binärarea.

```
string filePath = ”C:\\fourthCoffee\\settings.txt”;
byte[] rawSettings = File.ReadAllBytes(FilePath);
```

.eo/ec

Läsa och skriva information genom File Class (forts.)

- WriteAllText – ger möjlighet att skriva innehåll i sträng till fil.

```
string filePath = ”C:\\fourthCoffee\\settings.txt”;
string settings = ”foretagsnamn=fourth coffee:”;
File.WriteAllText(filePath, settings);
```

- WriteAllLines – skriver innehåll i area till fil.

```
string filePath = ”C:\\fourthCoffee\\hosts.txt”;
string[] hosts = { ”192.168.0.10”, ”192.168.0.11” };
File.WriteAllLines(filePath, hosts);
```

- WriteAllBytes – skriver innehåll i binär area till fil.

```
string filePath = ”C:\\fourthCoffee\\settings.txt”;
byte[] rawSettings = { ”192.168.0.10”, ”192.168.0.11” };
File.WriteAllBytes(filePath, rawSettings);
```

.eo/ec

Läsa och skriva information genom File Class (forts.)

- AppendAllText – ger möjlighet att skriva innehåll i sträng i slutet av en existerande fil.

```
string filePath = ”C:\\fourthCoffee\\settings.txt”;
string settings = ”kontakt=Nisse Hult”;
File.AppendAllText(filePath, settings);
```

- AppendAllLines – ger möjlighet att skriva innehåll i strängarea i slutet av en existerande fil.

```
string filePath = ”C:\\fourthCoffee\\hosts.txt ”;
string[] newHosts = { ”192.168.0.10”, ”192.168.0.11” };
File.Append.AllLines(filePath, hosts);
```

.eo/eo

Manipulera fil

- Förutom att läsa och skriva till filer, behöver applikation möjlighet att interagera med filer som är lagrad i filsystemet.
- Funktionaliteten kan implementeras med klasserna File och FileInfo.
- Klassen File tillhandahåller statiska metoder som kan användas för att utför manipulering av fil:
  - Copy.
  - Delete.
  - Exists.
  - GetCreationTime.
- Används utan instasisering.

.eo/eo

Manipulera fil (forts.)

- Klassen FileInfo används för att manipulera fil i internminnet.
- Innan ingående metoder kan användas, måste klassen instansieras.

```
string FilePath = ”C:\\fourthCoffee\settings.txt”;
FileInfo settings = new FileInfo(filePath);
```

- Metoder som ingår:
  - CopyTo.
  - Delete.
  - Exists.
  - Extensions.
  - Length.

.eo/eo

Manipulera katalog

- Vanligt krav för applikation är att interagera och manipulera filsystemet.
- Klassbibliotek i .NET Framework tillhandahåller klasserna Directory och DirectoryInfo.
- På samma sätt som klassen File, tillhandahåller Directory statiska metoder:
  - CreateDirectory.
  - Delete.
  - Exists.
  - GetDirectories.
  - GetFiles.

.eo/ec

Manipulera katalog (forts.)

- Innan du kan använda de metoder som, DirectoryInfo tillhandahåller, måste du göra en instans av klassen.

```
string directoryPath = "C:\\fourthCoffee\tempData";
Directory.CreateDirectory(directoryPath);
```

- Ger åtkomst till metoderna:
  - Create.
  - Delete.
  - Exists.
  - FullName.
  - GetDirectories.
  - GetFiles.

.eo/ec

Manipulera sökvägar

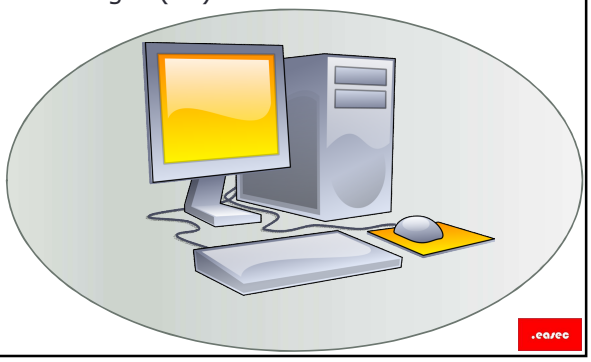
- .NET Framework finns klassen Path som innehåller ett antal verktyg för att skapa och hantera sökvägar i filsystemet.

```
string tempDirectoryPath = Path.GetTempPath();
```

- Innehåller ett antal statiska metoder:
  - HasExtensions.
  - GetExtensions.
  - GetTempFileName.

.eo/ec

Övning Manipulera filer, kataloger och sökvägar (8a)



.easec

Lektion 2: Arbeta med Serializing och Deserializing

- Vad är Serialization?
- Format.
- Skapa typ.
  - Skapa constructor.
  - Lägg till attribut.
  - Implementera interface och metod.
  - Definera publika medlemmar.
- Serializing och deserializing objekt till binär representation.
- Serializing och deserializing objekt till XML.
- Serializing och deserializing objekt till JSON.
- Skapa egen skräddarsydd serializer.

.easec

Vad är Serialization?

- Applikation typiskt bearbetar information, information läses in till internminnen från fil eller webbtjänst, skickas sedan vidare till andra komponenter för bearbetning.
- Komponenter som används för bearbetning, inte ens körs på samma maskin.
- Serialization är process för att konvertera informationen till en form som är bestående eller möjlig att transportera.

.easec

Format

- Det finns många olika format:
  - Binär.
  - XML.
  - JSON
  - Skräddarsytt format.

.easec

Skapa typ

- .NET Framework innehåller många klasser som funktioner där funktionen kan användas.
- Namespace:
  - System.
  - System.Runtime.Serialization.
- Arbetsgång:
  - Skapa constructor.
  - Lägg till attribut.
  - Implementera interface och metod.
  - Definiera publika medlemmar och eventuella medlemmar som inte skall använda funktionen.

.easec

Skapa constructor

- Första steget är att skapa constructor:

```
public class ServiceConfiguration
{
        public ServiceConfiguration()
        {
                ...
        }
}
```

.easec

Lägg till attribut

- Lägg därefter till attributet [Serializable].

```
[Serializable]
public class ServiceConfiguration
{
        public ServiceConfiguration()
        {
                ...
        }
}
```

.easec

Implementera interface och metod

- Implementera interface och metod som tillhandahålls av: System.Runtime.Serialization:

```
[Serializable]
public class ServiceConfiguration : ISerializable
{
        public ServiceConfiguration(SerializationInfo info,
StreamingContext context)
        {
                ...
        }
        public void GetObjectData(SerializationInfo info,
StreamingContext context)
        {
                ...
        }
}
```

.easec

Definera publika medlemmar

- Definiera publika medlemmar och eventuella medlemmar som inte skall använda funktionen.

```
[Serializable]
public class ServiceConfiguration : ISerializable
{
        [NoSerialized]
        private Guid _internalId;
        public string ConfigName { get; set; }
        public string DatabaseHostName { get; set; }
        ...
        public ServiceConfiguration()
        {
        }
        public ServiceConfiguration(SerializationInfo info,
StreamingContext context)
        {
```

.easec

Serializing och deserializing objekt till binär representation

- .NET Framework tillhandahåller klassen BinaryFormatter i System.Runtime.Serialization.Formatters.Binary.
- Används för serialization och deserialization av objekt till binär representation.

.eo/ec

Serializing och deserializing objekt till binär representation -exempel

- Serializing:

```
// Skapa objekt som du vill använda funktionen på.
ServiceConfiguration config = ServiceConfiguration.Default;
// Skapa formaterare som skall användas för funktionen.
IFormatter formatter = new BinaryFormatter();
// Skapa stream där information kommer att buffras.
FileStream buffer = File.Create("C:\\fourthcoffee\\config.txt");
// Kalla på metod.
formatter.Serialize(buffer, config);
// Stäng stream.
buffer.Close();
// Serializing konverterar informationen, skriver inte information
// till fil.
```

.eo/ec

Serializing och deserializing objekt till binär representation -exempel

- Deserializing:

```
// Skapar formatter som skall användas
Iformatter formatter = new BinaryFormatter();
// Skapa stream, där informationen kommer att buffras
FileStream buffer = File.OpenRead("C:\\fourthcoffe\\config.txt");
// Kalla på metod för deserializing
ServiceConfiguration config = formatter.Deserialize(buffer) as
ServiceConfiguration;
// Stäng stream
buffer.Close();
```

.eo/ec

Serializing och deserializing objekt till XML

- .NET Framework tillhandahåller klassen SoapFormatter i System.Runtime.Serialization.Formatters.Soap.
- Används för serialization och deserialization av objekt till XML.

.easec

Serializing och deserializing objekt till XML -exempel

- Serializing:

```
// Skapa objekt som du vill använda funktionen på.
ServiceConfiguration config = ServiceConfiguration.Default;
// Skapa formaterare som skall användas för funktionen.
IFormatter formatter = new SoapFormatter();
// Skapa stream där information kommer att buffras.
FileStream buffer = File.Create("C:\\fourthcoffee\\config.xml");
// Kalla på metod.
formatter.Serialize(buffer, config);
// Stäng stream.
buffer.Close();
// Serializing konverterar informationen, skriver inte information
// till fil.
```

.easec

Serializing och deserializing objekt till XML -exempel

- Deserializing:

```
// Skapar formatter som skall användas
Iformatter formatter = new SoapFormatter();
// Skapa stream, där informationen kommer att buffras
FileStream buffer = File.OpenRead("C:\\fourthcoffe\\config.xml");
// Kalla på metod för deserializing
ServiceConfiguration config = formatter.Deserialize(buffer) as
ServiceConfiguration;
// Stäng stream
buffer.Close();
```

.easec

Serializing och deserializing objekt till JSON

- .NET Framework tillhandahåller klassen DataContractJsonSerializer i System.Runtime.Serialization.Json.
- Stegen är lite annorlunda, klassen är underklass till XmlObjectSerializer och inte en implementering av interface IFormatter.

.easec

Serializing och deserializing objekt till JSON -exempel

- Serializing:

```
// Skapa objekt som du vill använda funktionen på.
ServiceConfiguration config = ServiceConfiguration.Default;
// Skapa objekt som skall användas för funktionen.
DataContractJsonSerializer jsonSerializer = new
DataContractJsonSerializer(config.GetType();
// Skapa stream där information kommer att buffras.
FileStream buffer = File.Create(”C:\\fourthcoffee\\config.txt”);
// Kalla på metod.
jsonSerializer.WriteObject(buffer, config);
// Stäng stream.
buffer.Close();
```

.easec

Serializing och deserializing objekt till JSON -exempel

- Deserializing:

```
// Skapa objekt som du vill använda funktionen på.
DataContractJsonSerializer jsonSerializer = new
DataContractJsonSerializer(typeof(ServiceConfiguration));
// Skapa stream, där informationen kommer att buffras
FileStream buffer = File.OpenRead(”C:\\fourthcoffe\\config.txt”);
// Kalla på metod för deserializing
ServiceConfiguration config = jsonSerializer.ReadObject(buffer) as
ServiceConfiguration;
// Stäng stream
buffer.Close();
```

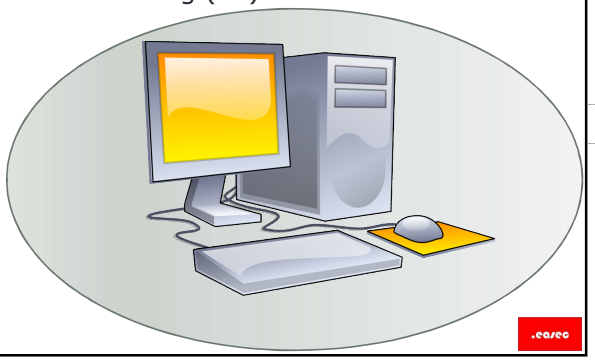
.easec

Skapa egen skräddarsydd serializer

- .NET Framework tillhandahåller interface Iformatter i System.Runtime.Serialization.
- Du följer sedan samma mönster som för BinaryFormatter och SoapFormatter klasserna.

.easec

Övning Arbeta med Serializing och Deserializing (8b)



.easec

Lektion 3: Arbeta med I/O genom Streams

- Vad är Streams?
- Typer av Streams i .NET Framework.
- Läsa och skriva binärinformation genom att använda Streams.
- StreamReader och StreamWriter.

.easec

Vad är Streams?

- Stream är en sekvens av information i bytes.
- Denna sekvens kan komma fråm filsystemet, nätverksanslutning eller internminne.
- .NET Framework tillhandahåller ett antal klasser för detta syfte, bl a:
  - Abstract klass Stream.
  - FileStream.
  - MemoryStream.
- Återfinns i System.IO.

.eo/ec

Typer av Streams i .NET Framework

- .NET Framework tillhandahåller många klasser för Stream, som ger möjlighet till att läsa och skriva information till olika typer av källor.
- Klasser som tillhandahåller tillgång till informationskällor:

Klasser	Beskrivning
FileStream	Exponerar stream till fil i filsystemet
MemoryStream	Exponerar stream till plats i internminnet
NetworkStream	Exponerar stream till nätverksplats

.eo/ec

Typer av Streams i .NET Framework (forts.)

- .NET Framework tillhandahåller många klasser för Stream, som ger möjlighet till att läsa och skriva information till olika typer av källor.
- Klasser som tillhandahåller möjlighet att läsa och skriva till eller från streams:

Klasser	Beskrivning
StreamReader	Läser textinformation från stream
StreamWriter	Skriver textinformation till stream
BinaryReader	Läser binärinformation från stream
BinaryWriter	Läser binärinformation till stream

.eo/ec

Läsa och skriva binärinformation genom att använda Streams

- Du kan använda klasserna BinaryReader och BinaryWriter för att använda streams.
- Varför arbeta med binär information:
  - Skriva information mycket snabbare.
  - Binär information tar upp mindre lagringsplats.
  - Är inte läsbar av människa.

Läsa och skriva binärinformation genom att använda Streams (forts.)

- Initiera klasserna BinaryReader och BinaryWriter:

```
string filePath = "C:\\fourthcoffee\\settings.txt";
FileStream file = new FileStream(filePath);
...
BinaryReader reader = new BinaryReader(file);
...
BinaryWriter writer = new BinaryWriter(file);
```

StreamReader och StreamWriter

- Du kan använda klasserna StreamReader och StreamWriter för att hantera textinformation istället för binärinformation.
- Processen för att läsa och skriva information med StreamReader och StreamWriter liknar processen för binärinformation.

StreamReader och StreamWriter (forts.)

- Initiera klasserna StreamReader och StreamWriter:

```
string destinationFilePath = "C:\\fourthcoffee\\settings.txt";
FileStream file = new FileStream(destinationFilePath);
...
StreamReader reader = new StreamReader(file);
...
StreamWriter writer = new StreamWriter(file);
```

.easec

Övning StreamReader och StreamWriter (8c)

.easec

Repetitionsfrågor

.easec