Module 15

"Serialization"

TEKNOLOGISK INSTITUT

Agenda

- Serialization
- XML Serialization
- JSON Serialization
- Custom Serialization

Introducing Serialization

- Often objects will need to be persisted and stored to e.g. files
- *Serialization*
  - is the process of generating a stream of bytes from object graphs representing objects
- *Deserialization*
  - reconstructs objects from the serialized representation
- Occasionally, these processes are refered to as ”dehydrating” and ”hydrating”.

.NET Serialization

- .NET provides a multitude of built-in support for serialization

- ObjectManager automatically
  - Enumerates and traverses object graphs
  - Detects cycles in objects graphs being serialized
  - Creates stream for members

- Formatters
  - Convert object state to/from streams of bytes

Built-in Formatters

- `BinaryFormatter`
  - In `System.Runtime.Serialization.Formatters.Binary` namespace
- `SoapFormatter`
  - In `System.Runtime.Serialization.Formatters.Soap` namespace
  - Must be explicitly referenced

- Both formatters
  - implement `IFormatter`
  - transform entire object state!
    - `IFormatter.Serialize()`
    - `IFormatter.Deserialize()`

- Use same formatter for both (de)serialization directions

Serializable Classes

- Classes must be marked with the `[Serializable]` attribute

```
[Serializable]
class ShoppingCartItem
{
   public int productId;
   public decimal price;
   public int quantity;
   public decimal total;
}
```

- All members are then automatically serialized

Non-serialized Members

- You can exclude members from serialization using the `[NonSerialized]` attribute

```
[Serializable]
class ShoppingCartItem
{
   public int productId;
   public decimal price;
   public int quantity;

   [NonSerialized]
   public decimal total;
}
```

- Often use will need to exclude members such as
  - Computed members
  - Database connections
  - Events and delegates
  - …

IDeserializationCallback

- Occasionally, it is necessary to postprocess deserialized objects
- Implement **IDeserializationCallback** to do this manually

```
[Serializable]
class ShoppingCartItem : IDeserializationCallback
{
    ...
    [NonSerialized]
    public decimal total;

    public void OnDeserialization( object sender )
    {
        CalculateTotal();
    }
}
```

Agenda

- Serialization
- **XML Serialization**
- JSON Serialization
- Custom Serialization

TEKNOLOGISK INSTITUT

XML Serialization

- `XmlSerializer` class serializes objects to pure XML
  - No SOAP wrapping
  - Serializes only public members

- Note: Class must have a default constructor!

- Create an `XmlSerializer` object for the specific type to be (de)serialized
  - `XmlSerializer.Serialize()`
  - `XmlSerializer.Deserialize()`

Serializing to XML

```
public class ShoppingCartItem
{
   public int productId;
   public decimal price;
   public int quantity;
   public decimal total;

   public ShoppingCartItem()
   {
   }
}
```

```
XmlSerializer xs = new XmlSerializer( typeof( ShoppingCartItem ) );
xs.Serialize( fs, item );
```

Controlling XML Serialization

TEKNOLOGISK INSTITUT

- Attributes for controlling the generated XML
  - `[XmlIgnore]`
    - Exclude properties from the serialization process
  - `[XmlElement]`
    - Serialize as an XML element
      `<element>value</element>`)
  - `[XmlAttribute]`
    - Serialize as an XML attribute
      `<class attribute="value"></class>`
  - `[XmlArrayAttribute]`
    - Serialize as array
  - `[XmlArrayItemAttribute]`
    - Controls serialization of array members

IXmlSerializable

- XML serialization can be customized if desired

```
public class ShoppingCartItem : IXmlSerializable
{
   ...

   public XmlSchema GetSchema() { ... }
   public void ReadXml( XmlReader reader)
   {
      productId = int.Parse(reader.ReadString());
   }
   public void WriteXml(XmlWriter writer)
   {
      writer.WriteString(productId.ToString());
   }
}
```

Agenda

- Serialization
- XML Serialization
- **JSON Serialization**
- Custom Serialization

JSON Serialization

- `DataContractJsonSerializer` class serializes objects to JSON
  - JSON = JavaScript Object Notation
  - State-of-the-art!
  - Serializes only `[DataMember]` members in `[DataContract]` classes

```
{"price":19.95,"productId":1,"quantity":2,"total":39.90}
```

- Create an `DataContractJsonSerializer` object for the specific type to be (de)serialized
  - `DataContractJsonSerializer.WriteObject()`
  - `DataContractJsonSerializer.ReadObject()`

- In `System.Runtime.Serialization.Json` namespace

Serializing to JSON

```
[DataContract]
public class ShoppingCartItem
{
   [DataMember]
   public int productId;
   [DataMember]
   public decimal price;
   [DataMember]
   public int quantity;
   [DataMember]
   public decimal total;
}
```

```
DataContractJsonSerializer js
   = new DataContractJsonSerializer( typeof( ShoppingCartItem ) );
js.WriteObject( fs, item );
```

Agenda

- Serialization
- XML Serialization
- JSON Serialization
- **Custom Serialization**

ISerializable

- The **`ISerializable`** interface can supply custom serialization for formatters

- Serialization is handled by
  - **`ISerializable.GetObjectData()`** method

- Deserialization is performed in specialized constructor

Implementing **ISerializable**

TEKNOLOGISK
INSTITUT

```
public class ShoppingCartItem : ISerializable
{  ...
   protected ShoppingCartItem( SerializationInfo info,
                               StreamingContext context)
   {
      productId = info.GetInt32("ProductID");
      ...
      CalculateTotal();
   }

   public void GetObjectData( SerializationInfo info,
                              StreamingContext context)
   {
      info.AddValue("ProductID", productId);
      ...
   }
}
```

Serialization Events

- Serialization events
  - [OnSerializing]
  - [OnSerialized]
  - [OnDeserializing]
  - [OnDeserialized]

```
[OnDeserialized]
void CalculateTotal( StreamingContext sc )
{
   total = price * quantity;
}
```

- Ordering as above
  - What about `IDeserializationCallback.OnDeserialization`?

Versioning Serialization

TEKNOLOGISK INSTITUT

- The [OptionalField] attribute allows newer versions of a class to be deserialized from older versions

```
[Serializable]
class ShoppingCartItem
{
   public int productId;
   public decimal price;
   public int quantity;
   private decimal total;

   [OptionalField(VersionAdded = 2)]
   public int carriedSinceYear;
}
```

Implementing Custom Formatters

- Implement **`IFormatter`** interface to create custom formatter

```
public interface IFormatter
{
   SerializationBinder Binder { get; set; }
   StreamingContext Context { get; set; }
   ISurrogateSelector SurrogateSelector { get; set; }

   object Deserialize( Stream serializationStream );
   void Serialize( Stream serializationStream, object graph );
}
```

- `FormatterServices` helper class
- `Formatter`
  - abstract base class
  - Implements **`IFormatter`**

Summary

- Serialization
- XML Serialization
- JSON Serialization
- Custom Serialization

Question

- "You are defining a serializable class named **`CalcUtil`** that contains several child objects. **`CalcUtil`** contains a method named **`InitChildren()`** which performs actions on these child objects. You need to ensure that the **`InitChildren()`** method is executed after the **`CalcUtil`** object and all its child objects are recreated. Which two actions should you take?"
  (Each correct answer presents part of the solution. Choose two.)

a) Specify that **`CalcUtil`** inherits from **`ObjectManager`**.
[✓ b) Specify that **`CalcUtil`** implements **`IDeserializationCallback`**.]
c) Apply the **`OnDeserializing`** attribute to **`InitChildren()`**.
d) Apply the **`OnSerialized`** attribute to **`InitChildren()`**.
[✓ e) Create an **`OnDeserialization()`** method invoking **`InitChildren()`**.]
f) Create a **`GetObjectData()`** method that invokes **`InitChildren()`**

TEKNOLOGISK
INSTITUT