Översikt

- Skapa och arbeta med Dynamic Object.
- Hantera livscykel för objekt och kontrollera ohanterade resurser.

.eo/ee

Lektion 1: Skapa och arbeta med Dynamic Objects

- Introduktion.
- Vad är Dynamic Objects?.
- Hanterat programmeringsspråk.
- Ohanterat programmeringsspråk.
- Dynamic Language Runtime.
- Skapa Dynamic Object.

.eo/ee

Introduktion

- Mjukvarusystem kan vara komplexa och involvera applikationer som är implementerade i olika teknologier.
- Några applikationer kan vara hanterade .NET Framework applikationer, andra applikationer kan vara ohanterade C++ applikationer.
- Du kan behöva använda funktionalitet från dessa olika typer av applikationer, eller använda funktionalitet som exponeras genom Component Object Model (COM).

Vad är Dynamic Objects?

- C# är av typen strongly typed static language.
- När du skapar en variabel, kan du bara kalla på metoder och få tillgång till medlemmar som denna typ definierat.
- Om du försöker kalla på metod som typ inte implementerar, kommer din programkod inte att kompileras.
- Detta är bra, för kompilator kontrollerar och hittar fel som gör så att din kod inte fungerar.

Vad är Dynamic Objects? (forts.)

- .NET Framework tillhandahåller dynamiska objekt, dessa kontrolleras inte förrän programkoden körs.
- Detta ger möjlighet att skriva programkod snabbt, utan att behöva definiera varje medlem innan.
- Dynamiska språk oftast lite långsammare, efterom optimering av programkod inte görs av kompilatorn.
- Exempel:
  - IronPython, IronRuby.

Hanterat programmeringsspråk

- Visual C# är av typen hanterat (eng: managed) programmeringsspråk.
- Detta betyder att den programkod du skriver, exekveras av Common Language Runtime (CLR).
- CLR tillhandahåller ett antal fördelar:
  - Hantering av minne.
  - Code Access Security (CAS).
  - Undantagshantering.

Ohanterat programmeringsspråk

- C++ är av typen ohanterat programmeringsspråk.
- Hanteras utanför CLR och exekveringen är snabbare och mer flexibel.
- När applikationer byggs genom att använda .NET Framework, är det vanligt att återanvända äldre programkod, t ex i C++, dynamiska objekt förenklar den kod som krävs för att hantera ohanterad programmeringsspråk.

Dynamic Language Runtime

- .NET Framework tillhandahåller Dynamic Language Runtime (DLR), som innehåller de nödvändiga tjänsterna och komponenterna för att stödja dynamiska programmeringsspråk och närma sig de ohanterade språken.
- Ansvarar för kopplingen mellan exekverande programkod och dynamiska objekt.
- Binders, som är språkberoende, sköter ihopkopplingen.

Skapa Dynamic Object

- Infrastruktur för DLR tillhandahåller nyckelordet dynamic.
- Ger möjlighet att deklarera dynamiska objekt.
- När nyckelordet används, kommer följande att ske:
  - Variabel definieras av typen objekt. Vilket värde som helst kan tilldelas, liksom försök att anropa vilken metod som helst. DLR tillsammans med binders kommer att kontrollera typ och att medlem existerar.
  - Instruerar kompilator att inte utför kontroll på typ.
  - Stänger av IntelliSence.

.easec

Skapa Dynamic Object -exempel

```
using Microsoft.Office.Interop.Word;
...
dynamic word = new Application();
```

- Efter det att du har instansierat dynamiskt objekt, genom dynamic, kan du få tillgång till egenskaper och metoder på samma sätt som du skulle ha gjort i C#.

.easec

Skapa Dynamic Object -exempel

- Du arbetar på samma sätt som i C#, exempelvis:

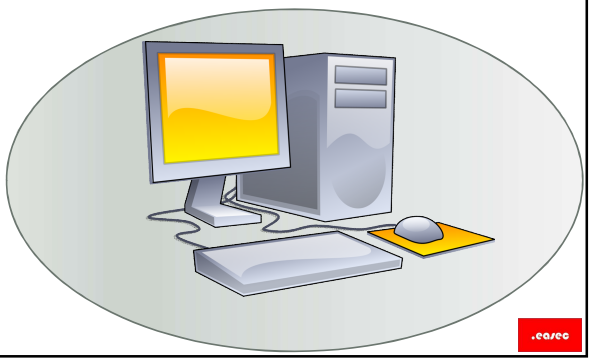
```
// Startar Microsoft Word.
dynamic word = new Application();
...
// Skapa ett nytt dokument
dynamic doc = word.Documents.Add();
doc.Activate();
```

- Skicka parameter till metod:

```
string filePath = "C:\\FourthCoffee\\Documents\\Schedule.docx";
...
...
dynamic word = new Application();
dynamic doc = word.Documents.Open(filePath);
doc.SaveAs(filePath);
```

.easec

Demonstration



Lektion 2: Hantera livscykel för objekt och kontrollera ohanterade resurser

- Objektets livscykel.
- CLR.
- Dispose Pattern.
- Hantera livscykel för objektet.

Objektets livscykel

- Ett objekts livscykel har ett antal steg, startar med skapandet av ett objekt och slutar när vi tar bort det.
- För att skapa ett objekt används nyckelordet *new*.

Objektets livscykel (forts.)

- När CLR exekverar kod för att skapa objektet, utförs följande:
  - Allokerar block i minnet, stort nog för hela objeket.
  - Initierar block av minne för objektet.
- När ett objekt förstörs:
  - Resurser kommer att släppas.
  - Minne kommer att återlämnas.

.easec

CLR

- CLR hanterar tilldelningen av minne för alla hanterade objekt, när ohanterade objekt används, får du skriva programkod för detta.
- När du är klar med objektet, kan du använda *dispose* och resurser kommer att släppas, t ex koppling till databas eller filhantering.
- CLR kommer att använda sig av Garbage collector, för att:
  - Frigöra resurser.
  - Återbörda minnet som har blivit tilldelat.
- Körs automatiskt för varje thread.
- När GC körs, stannar andra threads av.

.easec

Dispose Pattern

- Dispose Pattern är ett designmönster som kommer att frigöra resurser som objektet använder sig av.
- .NET Framework tillhandahåller IDisposable, som ligger i System.
- Denna definierar metod Dispose.
- StreamWriter, implementerar detta interface.
- Interfacet kan användas även för egna klasser som refererar till ej hanterade typer.

.easec

Hantera livscykel för objekt

- Det är inte tillräckligt att implementera IDisposable interfacet, du behöver även anropa metoden Dispose.
- Detta görs när du är klar med objektet.

```
var word = new ManagedWord();
...
word.Dispose();
```

- Att kalla på metoden direkt efter programkod som har använt den, är accepterat. Men tänk om ett undantag sker, kommer inte metoden att köras.

.easec

Hantera livscykel för object (forts.)

- Kalla på metoden i blocket finally, i ett block av try/catch/finally.

```
var word = new default(ManagedWord);
try
{
        word = new ManagedWord();
        // programkod för att använda hanterat objekt.
}
catch
{
        // programkod för att hantera eventuella fel.
}
finally
{
        if(word!=null)
                word.Dispose();
}
```

.easec

Repetitionsfrågor

.easec