



Programming the .NET Framework 4.5

Module 08 - Advanced Topics

In This Chapter

- ✦ Improving startup performance with NGEN
 - ✦ Advanced delegates and events
 - ✦ Advanced generics
 - ✦ Object cloning as serialization
 - ✦ Assembly loading problems and contexts
 - ✦ Code contracts
-

.NET Startup Performance

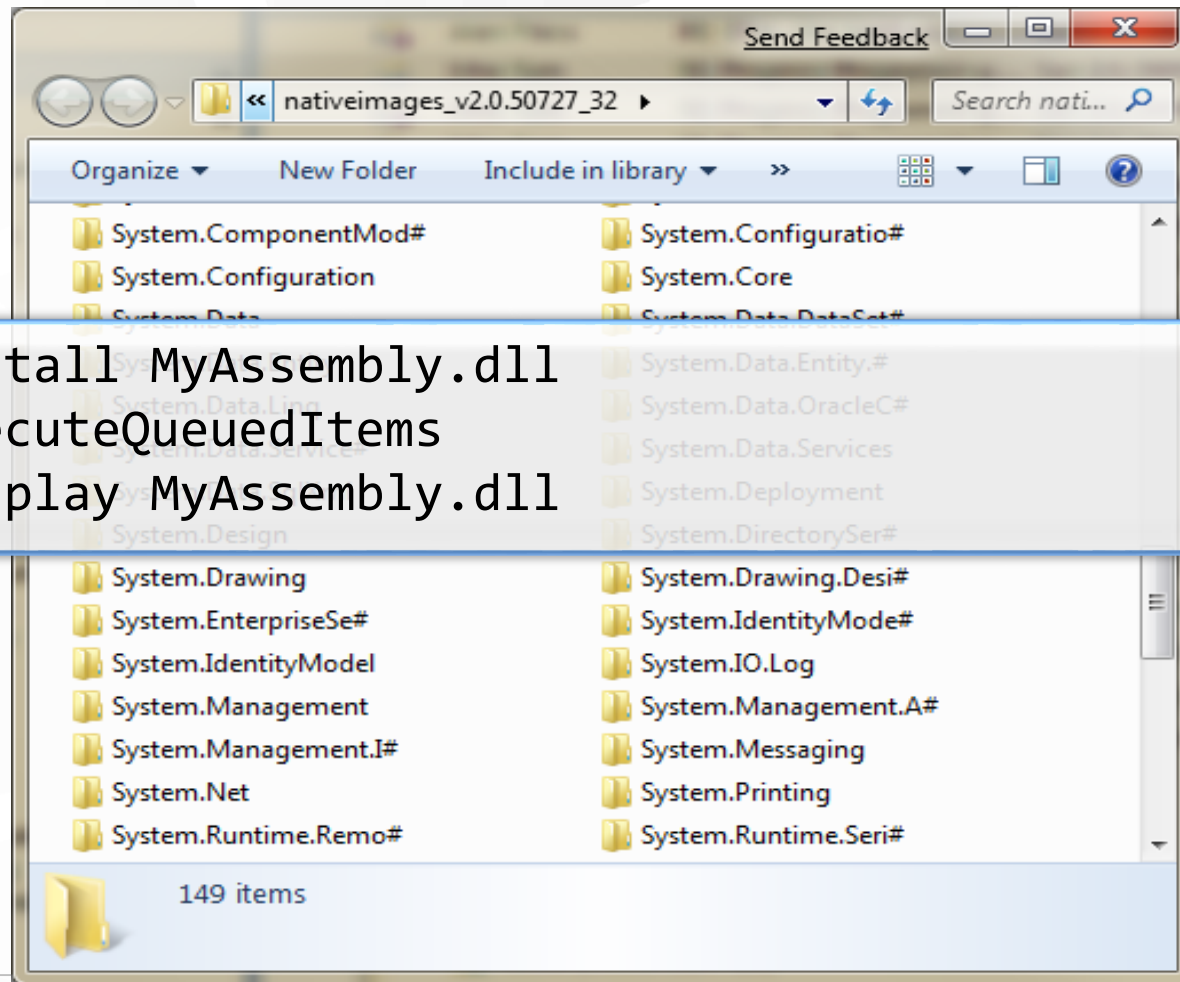
- ✦ What are the dominant factors in startup latency?
 - ✦ Cold startup – I/O
 - ✦ Warm startup – **JIT compilation**
-

Native Image Generator

- ✦ NGEN pre-compiles IL to native code
 - ✦ No JIT at runtime
 - ✦ IL images still required (metadata)
 - ✦ Automatically managed by NGEN service
-

NGEN Example

```
> ngen install MyAssembly.dll  
> ngen executeQueuedItems  
> ngen display MyAssembly.dll
```



Dynamically Binding to Delegates

- ✦ Delegate target unknown during compilation
- ✦ `Delegate.CreateDelegate`

```
1 logGenerator =  
2     (LogMethodDelegate)Delegate.CreateDelegate(  
3         typeof(LogMethodDelegate), @object, logMethod);
```

Dynamic Delegates and [LogMethod]

Demo



Dynamically Binding to Events

- ✦ Event source or handler unknown during compilation
- ✦ `EventArgs.AddEventHandler`

```
1 Delegate handler =  
2     Delegate.CreateDelegate(  
3         @event.EventHandlerType, @object,  
method);  
4 @event.AddEventHandler(null, handler);
```


Dynamic Events and [RegisterSystemEvent]

Demo



Event Registration Tricks

- ✦ Register with an anonymous method
 - ✦ Register with a lambda
 - ✦ Register with delegate { }
-

Invoking Events Asynchronously

- ✦ MulticastDelegate.GetInvocationList
- ✦ Invoke each handler asynchronously

```
1 public void Invoke(params object[] args)
2 {
3     foreach (Delegate del in
4         _del.GetInvocationList())
5     {
6         ThreadPool.QueueUserWorkItem(
7             delegate { del.DynamicInvoke(args);
8         });
9     }
10 }
```

AsyncInvoker and AsyncEventHandler

Demo



Generics and Reflection

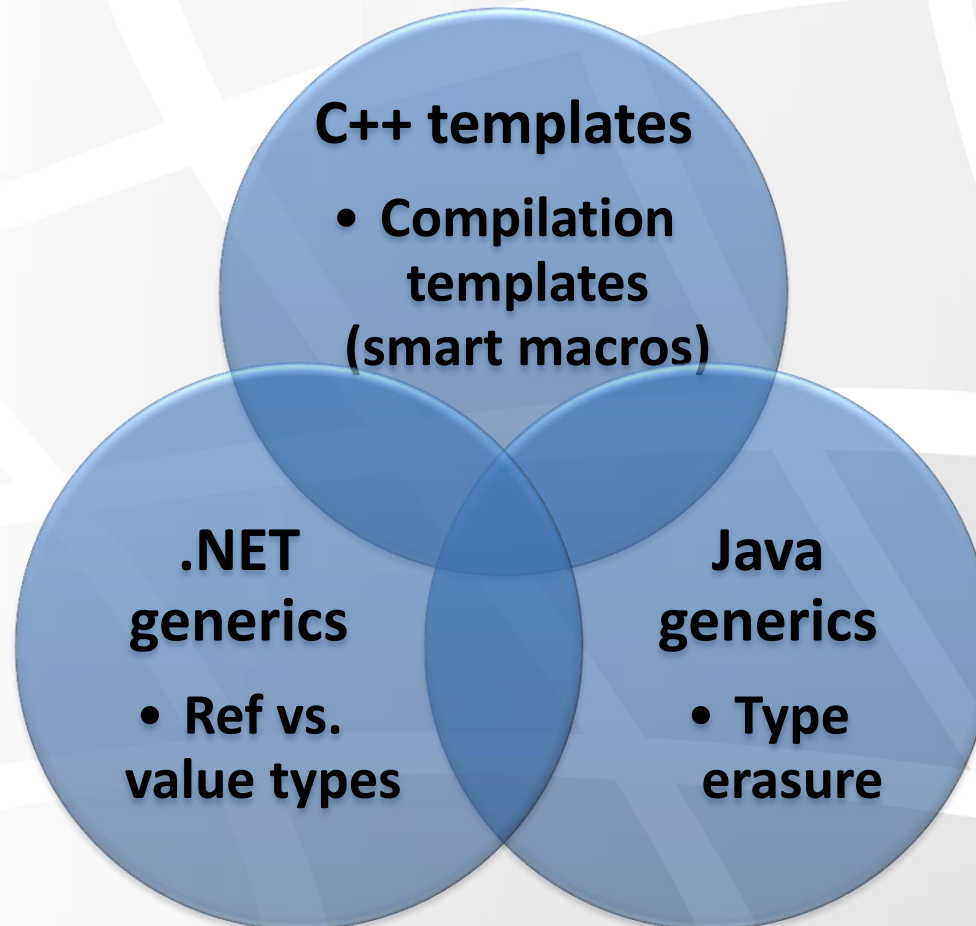
- ✦ Open generic type, closed generic type
 - ✦ `Type.MakeGenericType`,
`Type.GetGenericArguments` etc.
 - ✦ Dynamically create generic types (WCF channel, ...)
-

Dynamic Generics

Demo



Generics at Runtime



Object Cloning as Serialization

- ✦ Shallow clone: `Object.MemberwiseClone`
- ✦ Deep clone: `ICloneable.Clone`
- ✦ Cloning as serialization

```
1 public static T Clone<T>(this T obj)
2 {
3     using (MemoryStream stream = new MemoryStream())
4     {
5         _formatter.Serialize(stream, obj);
6         stream.Position = 0;
7         return (T)_formatter.Deserialize(stream);
8     }
9 }
```


Assembly Loading Diagnostics

- ✦ .NET assembly loading is not a straightforward process
 - ✦ It involves GAC, private probing, LoadFrom ...
 - ✦ Fuslogvw to the rescue
-

Assembly Load Contexts

- ✦ Two forms of assembly loading:
 - ✦ “Standard” load
 - ✦ JIT, `Assembly.Load`, implicit reference
 - ✦ “Load-from” load
 - ✦ `Assembly.LoadFrom`
-

Troubles Faced with Load Contexts

Load Context

- MainAssembly.exe
- Plugin.dll
 - SomeType

Load-From Context

- Plugin.dll
 - SomeType

.NET 3.5 SP1 Detailed Error

Unhandled Exception: System.InvalidCastException:

[A]Plugin.MyPlugin cannot be cast to [B]Plugin.MyPlugin.

Type A originates from 'Plugin, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null' in the context 'LoadFrom' at location '...\Plugin.dll'.

Type B originates from 'Plugin, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null' in the context 'Default' at location '...\Plugin.dll'.

fuslogvw and Assembly Load Contexts

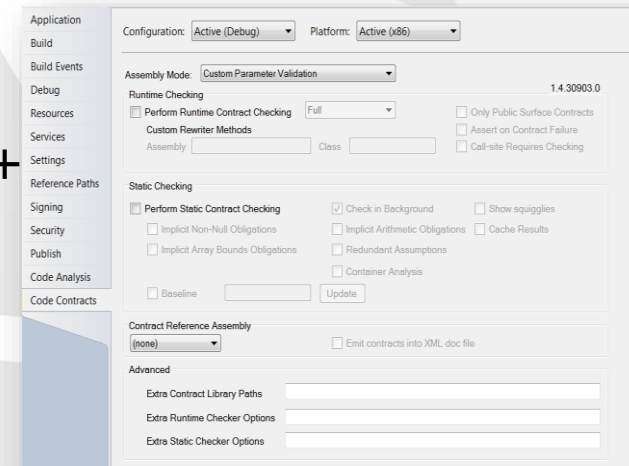
Demo



Code Contracts

- ✦ Express preconditions, postconditions, and object invariants
 - ✦ Checked at runtime (similar to assertions)
 - ✦ Used for static analysis, documentation generation

- ✦ Microsoft DevLabs project
 - ✦ Integrates with Visual Studio 2010+
 - ✦ Part of .NET 4.0+ *mscorlib*
 - ✦ System.Diagnostics.Contracts



Preconditions and Postconditions









```
1 public void Push(T item)
2 {
3     Contract.Requires(!IsFull);
4     Contract.Requires(item != null);
5     Contract.Ensures(Contract.OldValue(_top) + 1 == _top);
6     _items[++_top] = item;
7 }
8 public T Pop()
9 {
10    Contract.Requires(!IsEmpty);
11    Contract.Ensures(Contract.Result<T>() != null);
12    Contract.Ensures(Contract.OldValue(_top) - 1 == _top);
13    return _items[_top--];
14 }
```

Compilation and Runtime

- ✦ Code contracts can be analyzed statically
- ✦ Code contracts can be checked at runtime

Unhandled Exception:

System.Diagnostics.Contracts.__ContractsRuntime+ContractException:
Precondition failed: !IsFull

Error List	
 0 Errors  8 Warnings  1 Message	
	Description
 1	CodeContracts: requires is false: capacity >= 0
 2	+ location related to previous warning
 3	CodeContracts: ensures unproven: Contract.Result<T>() != null
 4	+ location related to previous warning
 5	CodeContracts: requires unproven: !IsFull

- ✦ Advanced uses not covered here
-

Static and Dynamic Contract Enforcement

Demo



Summary

- ✦ Improving startup performance with NGEN
 - ✦ Advanced delegates and events
 - ✦ Advanced generics
 - ✦ Object cloning as serialization
 - ✦ Assembly loading problems and contexts
 - ✦ Code contracts
-

Questions

