Instantiating and Manipulating Objects

Kevin Dockx
Architect

@KevinDockx https://www.kevindockx.com

Coming Up

Invoking constructors

Working with objects

- Interfaces
- Dynamics

Manipulating objects

- Properties and fields
- Methods

Coming Up

Reflection behind the scenes

Writing a self-configuring network monitor

Demo

Introducing the demo applications

Demo

Invoking constructors

Demo

Invoking a constructor dynamically by name

Demo

Working with an object through interfaces

Demo

Working with an object through dynamics

Demo

Getting and setting properties and fields

Demo

Invoking methods

Reflection Behind the Scenes

System.Reflection.RuntimeMethodInfo.Invoke, which calls into System.Reflection.RuntimeMethodInfo.UnsafeInvokeInternal

System.RuntimeMethodHandle.PerformSecurityCheck, which calls into System.GC.KeepAlive

System.RuntimeMethodHandle.InvokeMethod

Reflection Behind the Scenes

Getting the info object

- Type metadata parsing

Actual method invocation

- Argument validity checks
- Error handling

Security checks

- Restricted methods
- Dynamic code access security permissions

Writing a Self-configuring Network Monitor

Dynamic activation and invocation at runtime is reasonably common

- Relationships between the different components are determined at runtime
- Decreases the amount of tight coupling

```json
{
  "NetworkMonitorSettings": {
    "WarningService": "ReflectionSample.MailService",
    "MethodToExecute": "SendMail",
    "PropertyBag": {
      "Address": "kevin.dockx@gmail.com",
      "Subject": "Warning"
}}}
```

Writing a Self-configuring Network Monitor

The service to use, method to use and parameter list for this method are configured in a config file

```
{
  "NetworkMonitorSettings": {
    "WarningService": "ReflectionSample.MailService",
    "MethodToExecute": "SendMail",
    "PropertyBag": {
      "Address": "kevin.dockx@gmail.com",
      "Subject": "Warning"
}}}
```

Writing a Self-configuring Network Monitor

The service to use, method to use and parameter list for this method are configured in a config file

```
{
  "NetworkMonitorSettings": {
    "WarningService": "ReflectionSample.MailService",
    "MethodToExecute": "SendMail",
    "PropertyBag": {
      "Address": "kevin.dockx@gmail.com",
      "Subject": "Warning"
}}}
```

Writing a Self-configuring Network Monitor

The service to use, method to use and parameter list for this method are configured in a config file

```json
{
  "NetworkMonitorSettings": {
    "WarningService": "ReflectionSample.MailService",
    "MethodToExecute": "SendMail",
    "PropertyBag": {
      "Address": "kevin.dockx@gmail.com",
      "Subject": "Warning"
}}}
```

Writing a Self-configuring Network Monitor

The service to use, method to use and parameter list for this method are configured in a config file

Demo

Writing a self-configuring network monitor

Summary

Creating instances of classes

- GetContructor(s)
- Invoke (on ConstructorInfo)
- Activator.CreateInstance

Working with objects

- Work on the interface
- Work with dynamics

Summary

Getting and setting properties, fields, and executing methods follow likewise principles

- Base functionality contained in MemberInfo class

Summary

Reflection can be expensive

- Create the info object (involves parsing)
- Invoke the method (involves argument validation and error handling)
- Perform security checks