The System.Exception Base Class

All exceptions ultimately derive from the System.Exception base class, which in turn derives from

System.Object. Here is the crux of this class (note that some of these members are virtual and may thus

be overridden by derived classes):

public class Exception : ISerializable, \_Exception

{

// Public constructors

public Exception(string message, Exception innerException);

public Exception(string message);

public Exception();

// Methods

public virtual Exception GetBaseException();

public virtual void GetObjectData(SerializationInfo info,

StreamingContext context);

// Properties

public virtual IDictionary Data { get; }

public virtual string HelpLink { get; set; }

public Exception InnerException { get; }

public virtual string Message { get; }

public virtual string Source { get; set; }

public virtual string StackTrace { get; }

public MethodBase TargetSite { get; }

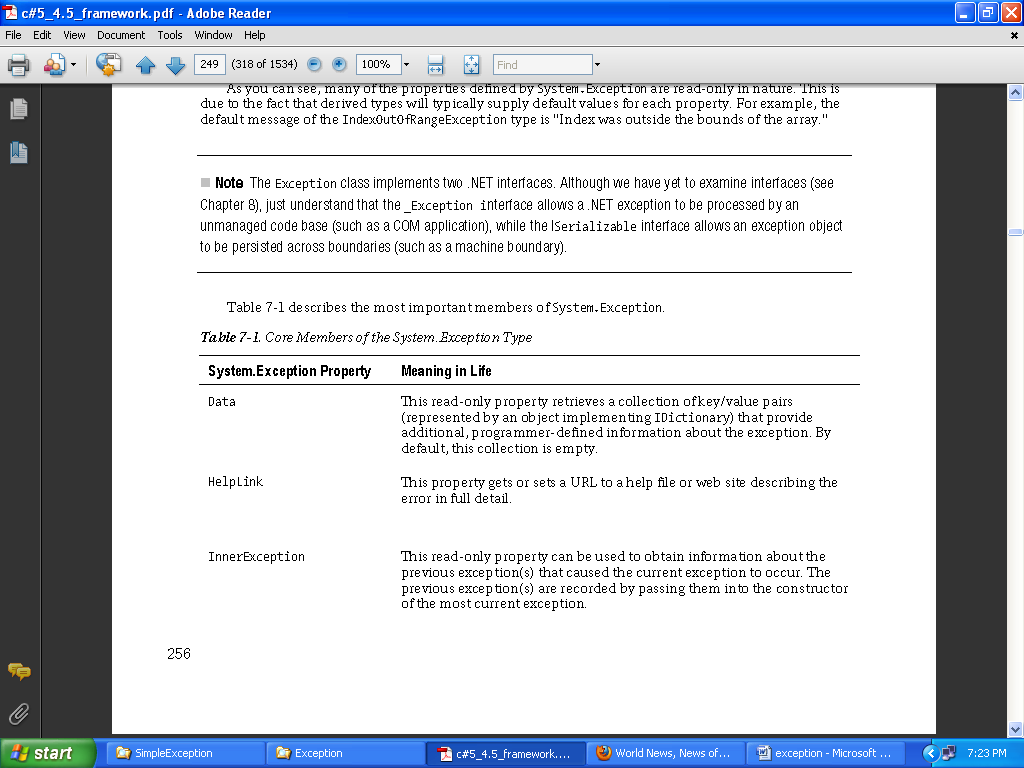
...

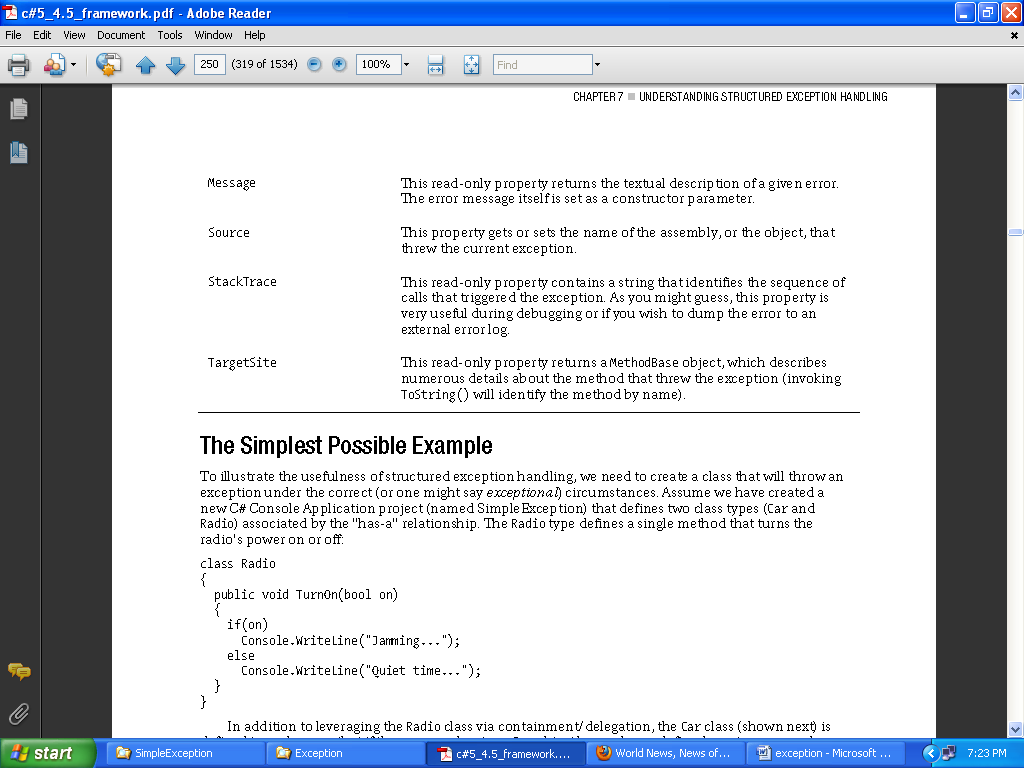
}

As you can see, many of the properties defined by System.Exception are read-only in nature. This is

due to the fact that derived types will typically supply default values for each property. For example, the

default message of the IndexOutOfRangeException type is “Index was outside the bounds of the array.”





**System.Exception Base Class**

**System.SystemException**

For example, the System namespace defines core exception objects such as ArgumentOutOfRangeException,

IndexOutOfRangeException,

StackOverflowException, and so forth.

Other namespaces define exceptions

that reflect the behavior of that namespace. For example, System.Drawing.Printing defines printing

exceptions, System.IO defines input/output-based exceptions, System.Data defines database-centric

exceptions, and so forth.

As a rule, all custom exception classes should be defined as public classes (recall, the default access

modifier of a non-nested type is internal). The reason is that exceptions are often passed outside of assembly boundaries, and should therefore be accessible to the calling code base.

Application-Level Exceptions (System.ApplicationException)

Functionally, the only purpose of System.ApplicationException is to identify the source

of the error. When you handle an exception deriving from System.ApplicationException, you can

assume the exception was raised by the code base of the executing application, rather than by the .NET

base class libraries or .NET runtime engine.