对于一个大的系统来说，异常处理必须要有一个总的策略和方针，并在组织内贯彻执行，否则就很难把异常处理好。

I. 从整个系统全局的角度来看

1. 进程内错误----用异常来报告错误（内部错误）；底层模块通过异常来向高层模块报告错误的发生。
2. 进程间错误----用ErrorCode, ErrorMessage, Action来向外部系统报告错误（业务错误），尤其是对外提供公共服务，除非是对外不公开的子系统，此时可以使用原生的异常来报告错误。一旦错误超出服务边界，最好使用ErrorCode, ErrorMessage, Action的方式向外部报告。

II. 从进程内不同模块的角度来看

1. 底层模块：DLL，一般不需要Catch异常；尽量多的使用Try--Finally；除非定义的自己的异常。
2. 高层模块：EXE等，尽量在高层模块中处理异常，报告错误。

III. 从不同的系统类型的角度来讲

1. 服务型：7* 24小时，通常情况下要确保系统的任何情况下都不要Crash，系统发生的任何Exception都要有Log，有些情况下可能还需要发邮件来通知。
2. UI型：友好的给出用户提示信息。必要的时候可以使用AOP来在全局唯一的点来处理Exception。

IV. 实际的异常处理规则

1. 异常类必须符合标准，必须可以序列化。

```
[Serializable]
public class ExampleException : ApplicationException
{
    public ExampleException()
        : base()
    {

    }

    public ExampleException(string message)
        : base(message)
    {

    }

    public ExampleException(string message, Exception inner) :
        base(message, inner)
    {

    }

    protected ExampleException(SerializationInfo info, StreamingContext conte
        base(info, context)
    {
```

```
    }
}
```

2. 不能丢掉任何一个异常，通常情况下要使用InnerException来保持原始的异常。

```
    try
    {
        //Read configuration from file
        ...
    }
    catch (FileNotFoundException ex)
    {
        throw new ExampleException("Can't find the configration file.", ex);
    }
```

3. Catch异常后如果不Throw的话，必须Log异常。
4. 记ExceptionLog时必须包含栈信息，也就是要使用ToString方法。