Exception Handling

هنگامی که در سی‌شارپ یا هر زبان دیگری کدنویسی می‌کنید، احتمالاً با مواردی مواجه خواهید شد که در حالت خاصی، برنامه شما دچار ایراد گردد. چطور؟

فرض کنید که برای مثال شما یک برنامه ماشین‌حسابی را نوشته‌اید. قطعاً باید این مورد را در نظر گرفته باشید که کاربر هنگامی که با این برنامه در حال کار کردن است، نباید دچار ایرادی گردد که باعث شود برنامه شما هنگ کرده و خاموش گردد. برای مثال، اگر کاربر بیاید عددی را بر صفر تقسیم کند، خوب مسلماً سی‌شارپ کنترل برنامه را از دست خواهد داد و برنامه لغو خواهد شد.

در چنین حالتی شما باید این نوع عملیات را در نظر گرفته باشید تا از بروز چنین خطاهایی جلوگیری کنید. سی‌شارپ دستوری با نام **Try … Catch** را در خود جای داده است که باعث می‌شود از خطاهای احتمالی جلوگیری کند و برنامه شما لغو نشود.

معرفی دستور Try … Catch

دستور **Try … Catch** بدین شکل عمل می‌کند که شما به سی‌شارپ می‌گویید که یک دستوری را امتحان کن **(Try**کن)، اگر بدون خطا بود که اجرا کن، اگر دارای خطا بود، آن را کنترل کن **Catch)** کن). قالب این دستور **(Syntax)**به شکل زیر است:

```
try
{
}
catch
{
}
```

در دستور زیر ما سعی می‌کنیم که یک فایل متنی را در **RichTextBox** ای با نام **rtb** لود کنیم:

```
try
{
```

```
rtb.LoadFile("C:/test.txt");
}
catch (System.Exception excep)
{
MessageBox.Show(excep.Message);
}
```

کد اصلی‌ای را که ما می‌خواهیم آن را در هنگام لود برنامه اجرا کنیم را همیشه در بین دو براکت مربوط به **Try** قرار می‌دهیم. ولی اگر که فایلی با نام **text.txt** وجود نداشت، دوست داریم که برنامه از کار نیفتد و پیغام **"File Not Found"**را داشته باشیم. ما می‌توانیم که این کار را در دستور **Catch** انجام دهیم. دقت کنید که در مقابل **Catch** دستور زیر است:

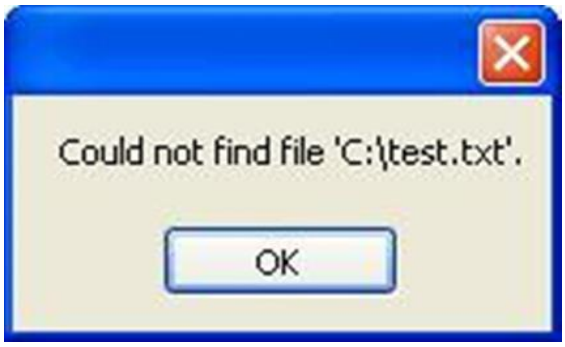
System.Exception excep

Exceptionیک شیءای در سی‌شارپ است که خطاها را کنترل می‌کند. این شیء در **Namespace** مربوط به **System**وجود دارد. پس می‌نویسیم**System.Exception** :

پس از ایجاد یک فاصله در مقابل آن**(Space)** ، نام متغیری برای آن تعریف می‌کنیم؛ که در اینجا آن را **Excep** نامیده‌ایم.

اگر در هنگام لود فایل خطایی ایجاد شد، سی‌شارپ خطای ایجاد شده را در متغیر **excep** ذخیره می‌کند. شما وقتی که از مشخصه **message** برای **excep** استفاده می‌کنید، می‌توانید خطای ایجاد شده را به‌صورت یک **String** داشته باشید که آن را می‌توان در یک **MessageBox** نمایش داد.

وقتی که برنامه خود را اجرا کنید؛ اگر فایل ذکر شده وجود نداشته باشد، پیغام زیر برای شما نمایش داده خواهد شد:



اگر نوع خطایی که ممکن است بوجود بیاید را از قبل میدانید، می‌توانید کدهای خودتان را به‌شکل زیر درآورید:

```
catch (System.IO.FileNotFoundException)
{
MessageBox.Show(“File Not Found!”);
}
```

برای اینکه بفهمید که خطایی که قرار است ایجاد شود از چه نوع است می‌توانید از کد زیر استفاده کنید:

```
catch (System.Exception excep)
{
MessaegBox.Show(excep.GetType().ToString());
}
```

که توسط یک**MessageBox** ، نوع خطای ایجاد شده را برای شما می‌نویسد.

اگر که می‌خواهید از دردسرهای فوق خلاص شوید و خیلی کار را ساده پیش ببرید، کافیست کد زیر را جایگزین کنید:

```
try
{
rtb.LoadFile(“C:/test.txt”);
}
Catch
{
MessageBox.Show(“An error occurred”);
}
```

شما همچنان می‌توانید چندین **catch** قرار دهید:

```
try
{
rtb.LoadFile(“C:/test.txt”);
}
catch
{
MessageBox.Show(“An error occurred”);
}
```

```
catch
{
MessageBox.Show(“Couldn’t find the file”);
}
catch
{
MessageBox.Show(“Or maybe it was something else.”);
}
```

در نهایت، برای دستور Try .. Catch یک دستور نهایی با نام Finally نیز وجود دارد که می‌توانید کارهایی را که لازم است انجام دهد را در آن قرار دهید. فرض کنید که شما فایلی را باز کرده‌اید که نیاز دارد که آن را ببینید؛ در نتیجه می‌بایست آن را در Finally قرار دهید.

```
try
{
rtb.LoadFile(“C:/test.txt”);
}
catch (System.Exception excep)
{
MessageBox.Show(excep.Message);
}
Finally
{
//Close the file here
}
```

کدهای داخل Finally حتماً یکبار اجرا خواهند شد.