

## Hoofdstuk 7 : Debugging en Exceptionhandling

## Debugging en Exceptionhandling



# Debugging en Exceptionhandling

1. Inleiding
2. Debugging
3. Exception handling
4. IntelliTrace
5. Referenties

## 1. Inleiding

### ► The Excuse Manager



#### Excuse Manager

##### Description

- [Edit](#) My dog has a headache
- [Edit](#) My friend snoored the whole night. I couldn't sleep
- [Edit](#) Very bad file

#### Excuse Manager : Edit My dog has a headache

##### Excuse

###### Description

My dog has a headache

###### Results

My boss didn't believe me.

###### Last Used

15/02/2012 01:00:00

[Save](#)

Met JQuery kunnen we dit vervangen door een DatePicker (nog een beetje geduld)

# 1. Inleiding

## ► The Excuse Manager



But the program isn't working.....!

# 1. Inleiding

## ► Mogelijke fouten

- Syntax fouten
- Logische fouten
- Run time fouten



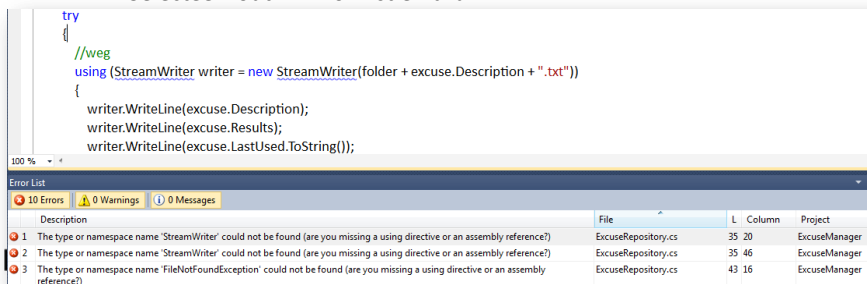
Oplossen alvorens we programma kunnen runnen



Opsporen tijdens de uitvoering van programma : debugging

# 1. Inleiding

- Syntax fouten
  - Fouten tegen de programmeertaal
  - Ontdekt at design-time of bij compilatie
  - Worden onderlijnd weergegeven + tooltip (met muis erover)
  - Kan je ook bekijken in Error List (View > Error List)
  - Voor meer info over fout :
    - Dubbelklik op fout in Error List : je gaat naar de lijn met de fout
    - Selecteer fout in Error List en druk F1

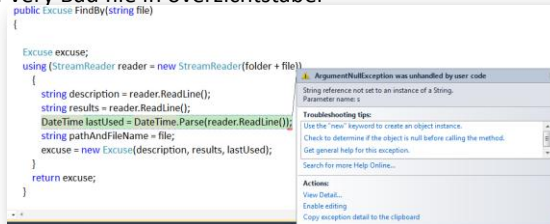


# 1. Inleiding

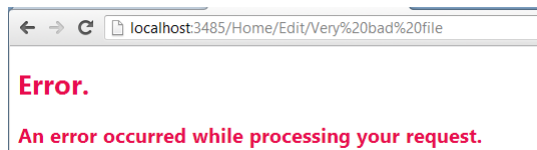
- Logische fouten
  - Programma werkt niet zoals verwacht
    - In voorbeeld : Index pagina toont voor de excuses een \
    - Unit testing kan helpen om het probleem op te lossen
  - Foutopsporing via **debuggen**
    - Tijdelijk onderbreken van uitvoering programma om
      - De programma flow te bekijken : bepalen welke methodes werden aangeroepen
      - Lijn per lijn door de code te gaan
      - De waarden van variabelen, properties en uitdrukkingen te bekijken en eventueel aan te passen
      - Methodes uit te testen
      - ...
  - Starten Debug proces : Debug > Start Debugging of F5

# 1. Inleiding

- Run time fouten
  - De uitvoering van het programma wordt gestopt door een fout -> Probeer dit via foutafhandeling op te vangen!
  - Vb. Bij selectie van Very Bad file in overzichtstabel
    - In debug mode



- Anders wordt omgeleid naar Error page



HoGent

# 2. Debugging

- ▶ Code lijn per lijn uitvoeren om te zien wat er gebeurt
- ▶ Maak gebruik van breakpoints
  - Een breakpoint = een regel waarbij de uitvoering van het programma tijdelijk onderbroken wordt
  - Klik de Margin Indicator bar naast de lijn met code of selecteer de code en druk F9 (rode bol)
  - Voorbeeld
    - De fout in de Index pagina, nl. alle excuses beginnen met \ : er zal waarschijnlijk iets misgaan bij het ophalen van de excuses. Dus...
    - Plaats breakpoint 1ste lijn in methode Index in HomeController
    - Start debugging : F5
    - Programma uitvoering stopt bij die regel (gele pijl). Deze lijn moet nog worden uitgevoerd!
    - Om het verloop van de code verder te bestuderen. Gebruik de Debug Menu/Toolbar

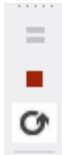
HoGent

Pag. 10

## 2. Debugging

- ▶ Commando's beschikbaar in break mode (Debug menu/Toolbar)
  - Start/Continue (F5, Debug > Start) : programma (verder) uitvoeren tot volgende breakpoint of tot einde
  - Break all : onderbreek programma onmiddellijk en ga naar instructie die wordt uitgevoerd
  - Stop : stop de uitvoering en exit de debugger (Shift + F5, Debug > Stop Debugging)
  - Restart : stop uitvoering en start programma opnieuw (Ctrl+Shift+F5, Debug > Restart)
  - Toon volgende instructie : ga naar de volgende instructie die zal worden uitgevoerd
  - Step into (F11, Debug > Step into) : voer volgende lijn code uit. Is de volgende lijn een methode, voer eerste instructie methode uit.
  - Step over (F10, Debug > Step Over) : voer volgende lijn code uit. Is dit een methode, voer de methode volledig uit als zijnde 1 instructie
  - Step out (Shift+F11, Debug > Step out) : voer de rest van de instructies in de huidige methode uit en onderbreek als dit gedaan is.

Continue



## 2. Debugging

- Voorbeeld
  - Eénmaal in break mode

```
public ActionResult Index()
{
    string[] excuses = excuseRepos.FindAll();
    return View(excuses);
}
```

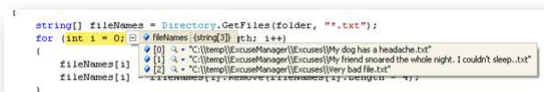
- Druk F11, om de lijn code uit te voeren. Vermits dit een methode is wordt vervolgens de eerste lijn in de methode aangeduid.
- Druk een aantal maal op F11 : voert aangeduide lijn uit,...
- Druk vervolgens Shift + F11 : de methode wordt volledig uitgevoerd, ...
- Druk F11
- Druk F5 : programma voert verder uit tot volgend breakpoint/einde

## 2. Debugging

- ▶ De debugging windows (Debug > Windows)
  - De breakmode wordt vaak gebruikt om waarden van variabelen of expressies af te drukken en te volgen. Dit kan op verschillende manieren:
    - Plaats muisaanwijzer boven de variabelenaam.
    - Quick watch : toont de waarde van een variabele, uitdrukking
    - Autos Window : toont de namen van de variabelen, waarden en gegevenstypes van huidige statement en de voorgaande coderegel
    - Local Window : tonen/aanpassen van waarden variabelen in huidige procedure/klasse (locale variabelen)
    - Watch window : om de waarde van een variabele, .. te volgen en aan te passen. Toont naam, waarde en gegevenstype
  - Of om code uit te testen
    - Immediate window

## 2. Debugging

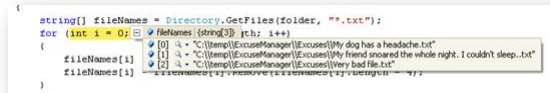
- Met muisaanwijzer over variabele
  - Toont de waarde
  - En/of +-teken. Klik op +-teken laat toe de properties te bekijken.
  - Voorbeeld
    - Restart programma (Ctrl+Shift+F5).
    - Programma wordt onderbroken bij eerste breakpoint
    - 2 maal F11
    - Ga met muisaanwijzer over fileNames en de IDE toont zijn waarde. fileNames bevat momenteel de null waarde
    - F11
    - Ga terug met muisaanwijzer over fileNames. Druk op +-teken voor verdere detail



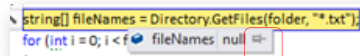
```
{
    string[] fileNames = Directory.GetFiles(folder, "*.txt");
    for (int i = 0; i < fileNames.Length; i++)
    {
        fileNames[i] = "C:\Temp\ExcuseManager\Excuses\My dog has a headache.txt"
        fileNames[i] = "C:\Temp\ExcuseManager\Excuses\My friend snored the whole night. I couldn't sleep..txt"
        fileNames[i] = "C:\Temp\ExcuseManager\Excuses\Very bad file.txt"
    }
}
```

## 2. Debugging

- Met muisaanwijzer over variabele
  - Ga terug met muisaanwijzer over fileNames. Druk op +-teken voor verdere detail



- Tip : hierdoor is de code niet meer volledig zichtbaar. Druk op **Ctrl** toets om code tijdelijk zichtbaar te maken
- Tip : je kan filenames ook vastpinnen zodat het steeds blijft staan

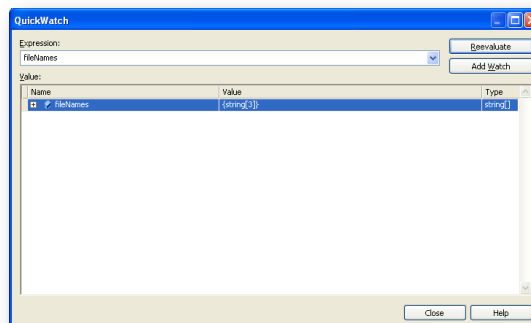


- Dit kan je dan verplaatsen op het scherm
- Of rechtsklik variabele > Pin to Source



## 2. Debugging

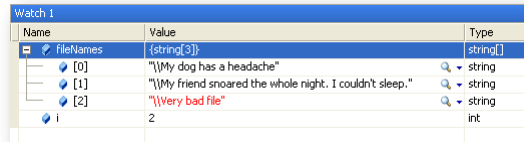
- De Quick Watch
  - Gebruik : om snel de waarde van een variabele, .. op te vragen en eventueel aan te passen
  - Selecteer variabele en druk CTRL+ALT+Q of Debug > Quick Watch
  - Je kan ook van hieruit een variabele toevoegen aan het Watch window, of een uitdrukking laten berekenen, of een waarde van een variabele aanpassen.





## 2. Debugging

- Het watch window (Debug>Windows>Watch)
  - Gebruik : observeren/aanpassen van de waarde van een variabele of uitdrukking tijdens de uitvoering van de code
  - Rechtermuisklik op variabele, selecteer in context menu Add Watch of via Debug > Windows > Watch (4 vensters)



Name	Value	Type
fileNames	{string[3]}	string[]
[0]	"{My dog has a headache"	string
[1]	"{My friend snoared the whole night. I couldn't sleep."	string
[2]	"{Wery bad file"	string
i	2	int

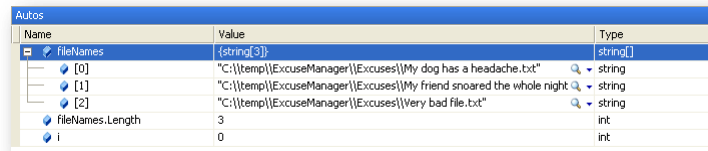
- Voorbeeld
  - Voeg fileNames en i toe aan het Watch window, eventueel ook fileNames[i]
  - Controleer of de waarde telkens met 1 verhoogt als je door de code stapt met F11. Controleer startwaarde en eindwaarde
  - Controleer de inhoud van fileNames[i]. Hoe wordt dit aangepast?
  - Voeg eventueel ook andere expressies toe. Zoek oorzaak “\” voor excuses in overzicht. Pas code aan en controleer opnieuw.

## 2. Debugging

- Het watch window
  - Elke instructie die geldig is binnen een programma, werkt binnen een watch venster
    - Voorbeeld : System.Threading.Thread.Sleep(2000) -> programma zal 2 seconden wachten (zie hourglass). Zal dan “Expression has been evaluated and has no value” teruggeven om aan te geven dat de instructie toch werd uitgevoerd maar geen return waarde heeft
  - Je kan ook de waarde van een variabele aanpassen
    - Voorbeeld
      - Vervang de waarde van i, nl. als de waarde gelijk is aan 2, plaats deze weer op 0. Wat gebeurt er?

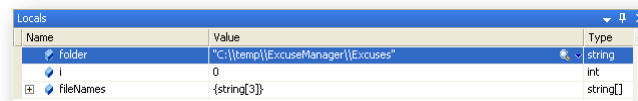
## 2. Debugging

- Autos Window : toont waarden variabelen in het actieve uitvoeringsstatement en de vorige code regel
- Debug>Windows>Autos of CTRL+ALT+V,A



Name	Value	Type
fileNames	{string(3)}	string[]
[0]	"C:\\temp\\ExcuseManager\\Excuses\\My dog has a headache.txt"	string
[1]	"C:\\temp\\ExcuseManager\\Excuses\\My friend snored the whole night"	string
[2]	"C:\\temp\\ExcuseManager\\Excuses\\Very bad file.txt"	string
fileNames.Length	3	int
i	0	int

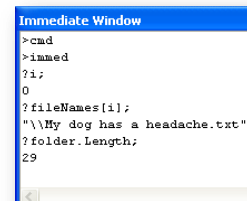
- Local Window : toont waarden variabelen in huidige procedure/form
- Debug>Windows>Locals of CTRL+ALT+V,L



Name	Value	Type
folder	"C:\\temp\\ExcuseManager\\Excuses"	string
i	0	int
fileNames	{string(3)}	string[]

## 2. Debugging

- ▶ Het venster Immediate
  - Kent 2 modi
    - Command : intypen van Visual Studio opdrachten
      - SaveAll, Close, Help, Exit,...
    - Immediate : Invoeren van instructies terwijl het programma onderbroken is
      - Waarde opvragen van variabele, property of uitdrukking: ?
      - Waarde aanpassen van variabele of property
      - Procedures, methodes uitvoeren
  - Switching modes
    - >cmd : switch naar Command mode
    - >immed : switch naar Immediate mode

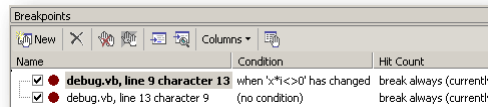


```
Immediate Window
>cmd
>immed
?i;
0
?fileNames[i];
"\\My dog has a headache.txt"
?folder.Length;
29
```

## 2. Debugging

### ► Geavanceerde breakpoints

- Debug > Windows > Breakpoints
  - Toont hit count (aantal maal reeds uitgevoerd)
  - Toevoegen, deleten, disablen van breakpoints
  - Instellen van condities : rechtermuisklik op een breakpoint , rechtermuis > Location..., Condition..., Hit count ....



## 2. Debugging

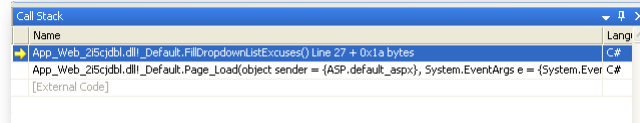
### ► Rechtsklik op lijn code

- Run to cursor : zal de uitvoering verderzetten tot deze lijn code bereikt is
  - Voorbeeld zet breakpoint in FindAll methode, eerste lijn code
  - Run
  - Selecteer dan de lijn return fileNames, rechtsklik > run to cursor
- Set Next Statement
  - Selecteer dan de lijn "int j=0" > rechtsklik > Set Next Statement
  - F11. Deze lijn code wordt uitgevoerd.

## 2. Debugging

### ► Call stack

- Volgen van de flow van programma (info over elke functieaanroep)
- Ctrl+Alt+C of Debug > Windows > Call Stack



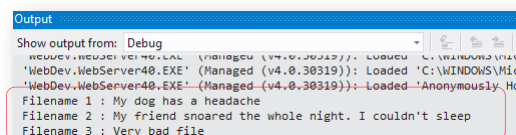
## 2. Debugging

### ► Debug.WriteLine(If) commando

- Debug class is onderdeel van System.Diagnostics.
- Print een boodschap naar de Visual Studio console.
- De uitvoering ervan gebeurt in debug mode (je hoeft de instructies dus niet te verwijderen bij release)

```
foreach (FileInfo f in files)
{
    fileNames[j++] = f.Name.Remove(f.Name.Length - f.Extension.Length);
    Debug.WriteLine("Filename " + j.ToString() + " : " + fileNames[j-1]);
}
```

- Tijdens runnen kan je Output venster bekijken (View > Output Window)



## 2. Debugging

- ▶ Edit and continue
  - Als de debugger stopt bij breakpoint, kan je code aanpassen en verdergaan met debuggen. VS hercompileert de applicatie en hercreëert de status van je applicatie op het debugger break moment
  - Moet je op 2 plaatsen enablen :
    - Tools > Options > Debugging > Edit and Continue > vink Enable Edit and Continue aan.
    - Project > Properties > Web > onderaan in Debuggers sectie "Enable Edit and Continue"
  - Run applicatie en ga ergens in breakmode. Voeg een lijn code toe (bvb var i=1;) en ga verder met de uitvoering

## 2. Debugging

- ▶ Script explorer : debuggen van scripts
- ▶ Threads
  - Toont treads igv multi threaded applicaties
- ▶ Modules
  - Toont de dll's en exe's gebruikt door het project
- ▶ Memory venster
  - Toont stukje geheugen gebruikt door applicatie
- ▶ Disassembly
  - Toont de assembly code die overeenkomt native code gecreëerd door de Just-in-Time (JIT) compiler, (niet de MSIL code gegenereerd door de Visual Studio compiler.)
- ▶ Registers
  - Toont de inhoud van de registers

## 2. Debugging

### ► Debug configuratie

- Gebeurt automatisch bij eerste maal klik op Debug. De web.config wordt aangemaakt met

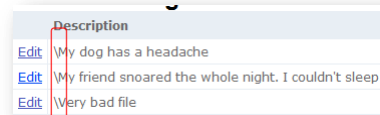
```
<compilation debug="true"/>
```

- Afzetten in productie!

## 2. Debugging

### ► Hoe pakken we oplossing van logische fout aan?

- Schrijf test



	Description
<a href="#">Edit</a>	My dog has a headache
<a href="#">Edit</a>	My friend snoared the whole night. I couldn't sleep
<a href="#">Edit</a>	Very bad file

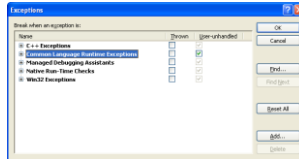
```
[TestMethod()]
public void AllExcusesContentsOk()
{
    excuseRepository = new ExcuseRepository(folder);
    string[] excuses = excuseRepository.FindAll();
    Assert.AreEqual("My dog has a headache", excuses[0]);
    Assert.AreEqual("My friend snoared the whole night. I couldn't sleep.", excuses[1]);
}
```

- Test faalt
- Pas code aan
- Test slaagt
- Ook testen kan je debuggen : Test > Debug

## 2. Debugging

### ► Debug mode en exceptions

- Normaliter : Het programma wordt onderbroken enkel bij unhandled exceptions. De debugger stopt in dit geval bij de lijn code waar exception zich voordoet. De debugger stopt niet bij afgehandelde exceptions.
- Kan je aanpassen : **Debug > Exceptions > Common Language runtime exceptions (eventueel verder openklappen): vink Thrown aan**



- De lijn code waar de exception zich voordoet (of enkele regels ervoor) is een goed startpunt voor debugging
- Voorbeeld : selecteer in overzicht "Very bad file"

## 2. Debugging

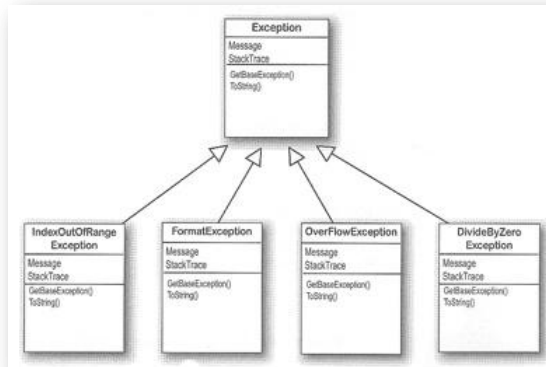
### ► Debug mode en exceptions

- Normaliter : Programma zal ook stoppen bij exceptions in gegenereerde code
- Kan je aanpassen : **"enable or disable Just My Code debugging"**
  - In Tools menu, kies Options.
  - In Options dialog box, open Debugging node en kies General.
  - Vink Enable Just My Code aan of uit.

### 3. Exceptionhandling

- ▶ Fouten genereren uitzonderingen (exceptions)

- Als er een fout optreedt, wordt een Exception object gecreëerd, met informatie over de uitzondering.



Exception is een basisklasse. Zo kan je je eigen Exception klassen schrijven.

ToString geeft een samenvatting van alle informatie in de exceptions attributen

### 3. Exceptionhandling

- ▶ Fouten genereren uitzonderingen (exceptions)

- Het Exception-object bevat informatie over de fout, en terwijl de gebeurtenis door de lagen heen naar boven borrelt, wordt de gebeurtenis ingepakt in steeds meer details. Het komt er ruwweg op neer dat de Application\_Error-uitzondering de Page\_Error-uitzondering bevat, die weer voortvloeit uit de basis-Exception, die het omhoog borrelen van de gebeurtenis in gang heeft gezet.

- ▶ Fouten dien je zoveel mogelijk op te vangen
- ▶ Igv niet opgevangen fouten, leid de gebruiker om naar een fout pagina
- ▶ Indien gewenst kan je niet opgevangen fouten loggen op applicatie niveau



## 3. Exceptionhandling

### ► Opvangen van fouten

- try ... catch...finally
  - Gebruik try..catch...finally waar mogelijk problemen kunnen opduiken
  - Voorbeeld : selectie van Excuse “Very bad file”.
    - Waar wordt de fout geworpen?
  - In de Help kan je voor elke methode terugvinden wat de mogelijke exceptions zijn

StreamReader.ReadLine Method  
[StreamReader Class](#) [See Also](#) [Send Feedback](#)

Exceptions	
Exception	Condition
<a href="#">OutOfMemoryException</a>	There is insufficient memory to allocate a buffer for the returned string.
<a href="#">IOException</a>	An I/O error occurs.

## 3. Exceptionhandling

- try ... catch...finally
  - Voorbeeld : selectie van Excuse “Very bad file”.
    - Is het wel ReadLine die een Exception throwt?
    - Wat is het dan wel? En welke Exception wordt geworpen?
- Enkele tips
  - Ontwerp code met exceptionhandling
  - Als je een exception moet opvangen, gebruik steeds de meest specifieke exception (niet gewoon Exception)
  - Maak NOOIT een leeg catch blok aan
  - Geef aan gebruikers bruikbare foutmeldingen, zodat gebruiker weet hoe hij fout kan aanpassen
  - Werp waar mogelijk .Net Exceptions. Werp enkel custom exceptions als je meer informatie wenst te geven. Geef dan ook de innerexception mee!
  - Vergeet het finally blok niet waar nodig. Zorg voor clean-up. (using is vaak een goed alternatief)
  - Meer op :  
<http://www.codeproject.com/KB/architecture/exceptionbestpractices.aspx>

### 3. Exceptionhandling

- Ook runtime fouten op TDD aanpakken :

```
DateTime lastUsed = DateTime.Parse(reader.ReadLine());
```

ArgumentNullException was unhandled by user code

- Schrijf testen

HoGent

```
[ExpectedException(typeof(FileNotFoundException))]  
[TestMethod()]  
public void ReadFileDoesNotExist() {  
    excuseRepository = new ExcuseRepository(folder);  
    Excuse actual = excuseRepository.FindBy("abc");  
}  
[ExpectedException(typeof(ArgumentException))]  
[TestMethod()]  
public void ReadFileNull() {  
    excuseRepository = new ExcuseRepository(folder);  
    Excuse actual = excuseRepository.FindBy(null);  
}  
[ExpectedException(typeof(ArgumentException))]  
[TestMethod()]  
public void ReadFileEmptyString() {  
    excuseRepository = new ExcuseRepository(folder);  
    Excuse actual = excuseRepository.FindBy(String.Empty);  
}  
[ExpectedException(typeof(ApplicationException))]  
[TestMethod()]  
public void ReadBadFile() {  
    excuseRepository = new ExcuseRepository(badFolder);
```

### 3. Exceptionhandling

- try ... catch...finally

Door using blok is finally blok overbodig. Using zorgt ervoor dat reader sowieso wordt afgesloten.

HoGent

```
public Excuse FindBy(string file)  
{  
    Excuse excuse;  
    if (String.IsNullOrEmpty(file))  
        throw new ArgumentException("file is required");  
    try  
    {  
        using (StreamReader reader = new StreamReader(folder + file))  
        {  
            string description = reader.ReadLine();  
            string results = reader.ReadLine();  
            DateTime lastUsed = DateTime.Parse(reader.ReadLine());  
            string pathAndFileName = file;  
            excuse = new Excuse(description, results, lastUsed);  
        }  
        return excuse;  
    }  
    catch (FileNotFoundException ex)  
    {  
        throw;  
    }  
    catch (Exception ex)  
    {  
        throw new ApplicationException("Contents file invalid", ex);  
    }  
}
```

### 3. Exceptionhandling

- ▶ Hoe dient de Controller dit op te vangen?
  - **TempData** : vergelijkbaar met Session maar de inhoud wordt enkel bijgehouden tot na de volgende HTTP request, en wordt dan weggegooid.
  - Ideaal bij redirections
  - Andere oplossing : **ModelState** (zie later validatie)

```
public ActionResult Edit(string id)
{
    try
    {
        Excuse excuse = excuseRepos.FindBy(id + ".txt");
        return View(excuse);
    }
    catch (Exception ex)
    {
        TempData["message"] = ex.Message;
        return RedirectToAction("Index");
    }
}
```

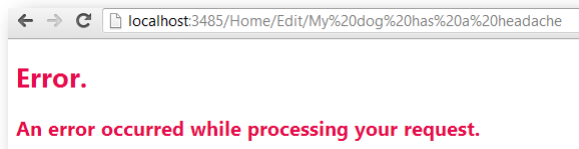
### 3. Exceptionhandling

- ▶ En de View?
  - Voorzie op elke pagina steeds een plaats voor foutboodschappen
  - Maak hiervoor gebruik van de Layout Page

```
<body>
<div class="field-validation-error">@TempData["message"]</div>
@RenderBody()
</body>
```

## 3. Exceptionhandling

- ▶ Wat als een fout niet werd opgevangen?
  - In debug mode wordt gestopt bij de lijn waar de exception geworpen wordt
  - In productie : gebruiker wordt omgeleid naar een foutpagina, zie Views > Shared > Error.cshtml
  - Voorbeeld :
    - Als bij editeren excuse tekst in de datum wordt ingegeven. (We hebben geen try catch toegevoegd)



## 3. Exceptionhandling

- ▶ Wat als een fout niet werd opgevangen?
  - In MVC wordt met een [HandleError] filter gewerkt.
  - Filters zijn stukjes code die uitgevoerd worden na of voor de uitvoering van een action methode of ActionResult. De [HandleError] filter wordt uitgevoerd na de uitvoering van de action methode of ActionResult.
  - Het uitvoeren van een filter kan worden ingesteld op niveau van actie methode, controller of op application niveau
  - In global.asax (op applicatieniveau)

```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();
    WebApiConfig.Register(GlobalConfiguration.Configuration);
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
}
```

### 3. Exceptionhandling

- ▶ Wat als een fout niet werd opgevangen?
  - In App\_Start > FilterConfig.cs

```
public class FilterConfig
{
    public static void RegisterGlobalFilters(GlobalFilterCollection filters)
    {
        filters.Add(new HandleErrorAttribute());
    }
}
```

### 3. Exceptionhandling

- ▶ Wat als een fout niet werd opgevangen?
  - Het omleiden definieer je in de web.config sectie <customErrors> in <system.web> gedeelte
    - Ipv het origineel foutbericht kan je aan externe gebruikers aangepaste fout pagina's tonen
      - Mode : status van de aangepaste fouten
        - On : aangepaste fouten zijn ingeschakeld
        - Off : geen aangepaste fouten
        - RemoteOnly (default) : aangepaste fouten worden enkel aan externe clients getoond (enkel als applicatie niet runt op localhost (developer ziet fout, user wordt omgeleid)
    - Om de werking in developmentmode te kunnen zien, voeg je onderstaande toe aan web.config. (Verwijder achteraf op plaats op RemoteOnly)




```
<customErrors mode="On"/>
```

### 3. Exceptionhandling

► Wat als een fout niet werd opgevangen?

- Bekijk Views > Shared > error.cshtml
- Geeft een Model object **HandleErrorInfo**

▲ Properties

	Name	Description
	ActionName	Gets or sets the name of the action that was executing when the exception was thrown.
	ControllerName	Gets or sets the name of the controller that contains the action method that threw the exception.
	Exception	Gets or sets the exception object.

Top

### 3. Exceptionhandling

► Wat als een fout niet werd opgevangen?

- Plaats onderstaande uit commentaar in Error.cshtml en run opnieuw

```
<p>Sorry, there was a problem processing your request. If you want to contact de webmaster  
to report the problem with more details,  
please <a href="mailto:webmaster@excuses.be">mail the webmaster</a>.</p>  
<p>What you normally don't show on this page (but can be usefull during development)<br />  
There was a <b>@Model.Exception.GetType().Name</b>  
while rendering <b>@Model.ControllerName</b>' s  
<b>@Model.ActionName</b> action.</p>  
<p>The exception message is <b>@Model.Exception.Message</b></p>  
<p>Stack trace :</p>  
<pre>@Model.Exception.StackTrace</pre>
```

- Idealiter dien je de fout ook te loggen. Hiervoor kan je zelf een custom Filter voor schrijven schrijven. Voor het loggen bestaat logging software zoals NLog. (zie later)

## 3. Exceptionhandling

### ► Eigen foutpagina

- Indien gewenst kan je dit overschrijven per actie methode/of alle acties in Controller. Plaats boven actie of boven Controller : [HandleError(View="Error2")]

- De properties

Property Name	Type	Meaning
Order	int	The execution order of this filter among other exception filters. Lower values go first. Inherited from <code>FilterAttribute</code> .
ExceptionType	Type	The exception type handled by this filter. It will also handle exception types that inherit from the specified value, but will ignore all others. The default value is <code>System.Exception</code> , which means that by default it will handle <i>all</i> standard exceptions.
View	string	The name of the view template that this filter renders. If you don't specify a value, it takes a default value of <code>Error</code> , so by default it would render <code>/Views/currentControllerName/Error.aspx</code> or <code>/Views/Shared/Error.aspx</code> .
Master	string	The name of the master page used when rendering this filter's view template. If you don't specify a value, the view uses its default master page.

## 3. Exceptionhandling

### ◦ Oefening

- Wat als de folder niet bestaat?
  - Maak de test
  - Pas de code aan
- Wat als geen files in de folder?
  - Test bestaat reeds
  - Hoe opvangen in controller?

### 3. Exceptionhandling

#### Brian finally gets his vacation...

Now that Brian's got a handle on his exceptions, his job's going smoothly and he can take that well-deserved (and boss-approved!) vacation day.

Good exception handling is invisible to your users. The program never crashes, and if there are problems, they are handled gracefully, without confusing error messages.



### 4. IntelliTrace

- ▶ Enkel in VS 2012 Ultimate
- ▶ Advanced debugging : Met IntelliTrace neem je de acties op en kan je dit later terug bekijken, afspelen
- ▶ Hiermee kan je ook data in productie verzamelen
- ▶ Meer op
  - <http://channel9.msdn.com/Events/TechEd/NorthAmerica/2012/DEV365>
  - <http://www.techbubbles.com/visual-studio-2010/using-intellitrace-in-visual-studio-2012/>



## Referenties

---

- ▶ Head First C#, O'Reilly, ISBN 0-596-51482-4