Elmah for Glimpse – Best of both worlds

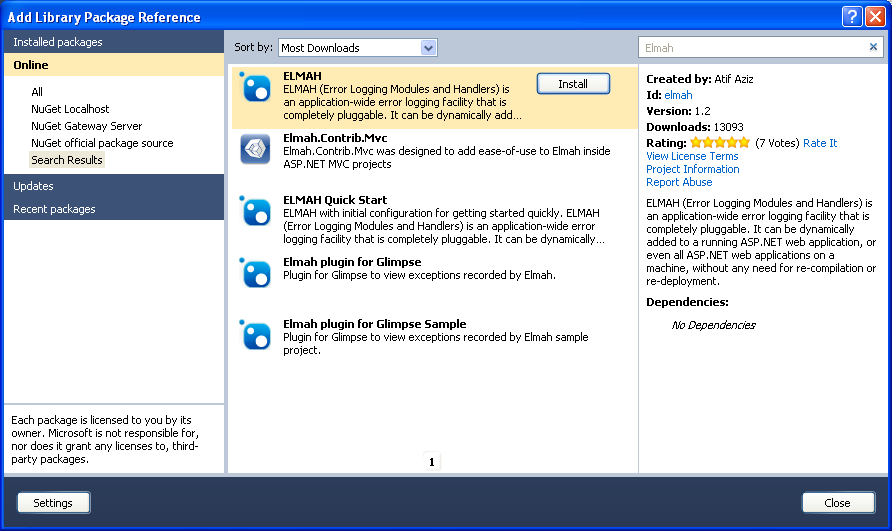
# Introducing Elmah

We all know [Elmah](http://code.google.com/p/elmah/) right? It’s one of the top packages in [NuGet](http://www.nuget.org/List/Packages/elmah). For those who don’t really know or use Elmah, I’ll give a brief explanation of the functionality it provides and how you can use it in your application.

Elmah stands for Error Logging Modules and Handlers. It’s an application-wide error logging library which logs nearly all unhandled exceptions. It provides you with a web page which enables you to (remotely) view the log of recorded exceptions. This log doesn’t only contain the error message; it also includes the entire stack trace and the values of all server variables at the time of the error.

# Using Elmah with ASP.NET MVC

Since we have NuGet, setting up Elmah with an ASP.NET MVC application has become really easy. Right click your web application and select ‘Add Library Package Reference…’ to pop up the NuGet Package Manager. Search for Elmah and install it. After installing, Elmah will have been added as a project reference.



The next step is to configure Elmah. I won’t go into too much detail here. We’ll just set up Elmah to log all errors to an in-memory repository and to expose those errors through a web page.

First let’s add the following code to the <configSections> node to make Elmah read it’s configuration from your web.config.

<sectionGroup name="elmah">

  <section name="errorLog"

requirePermission="false"

type="Elmah.ErrorLogSectionHandler, Elmah" />

</sectionGroup>

Elmah allows you to log errors to several storages. They currently support Microsoft SQL Server, Oracle, SQLite, Access, XML files, in-memory, SQL CE, MySQL and PostgreSQL. Insert the code below in your web.config to log error to an in-memory storage. When using an in-memory repository, all errors that got recorded will vanish when the application restarts.

<elmah>

  <errorLog type="Elmah.MemoryErrorLog, Elmah" />

</elmah>

We still have to do two things to get everything up and running:

1. Make Elmah catch and log all unhandled exceptions
2. Expose a web page to allow the user to view everything that got caught by Elmah

To achieve this we have to add an HTTP module and an HTTP handler. The module is responsible for logging the exceptions while the HTTP handler will render a page with a list of errors. You’ll have to add these sections to both the <system.web> node and the <system.webServer> node in order to support IIS running in Classic mode as well as Integrated mode. More info in this can be found on MSDN – [How to: Register HTPP Handlers](http://msdn.microsoft.com/en-us/library/46c5ddfy.aspx).

<system.web>

  <httpHandlers>

    <add type="Elmah.ErrorLogPageFactory, Elmah"

path="elmah.axd"

verb="POST,GET,HEAD" />

  </httpHandlers>

  <httpModules>

    <add name="ErrorLog" type="Elmah.ErrorLogModule, Elmah"/>

  </httpModules>

</system.web>

<system.webServer>

  <modules runAllManagedModulesForAllRequests="true">

    <add name="ErrorLog" type="Elmah.ErrorLogModule, Elmah"/>

  </modules>

  <handlers>

    <add name="Elmah"

type="Elmah.ErrorLogPageFactory, Elmah"

path="elmah.axd"

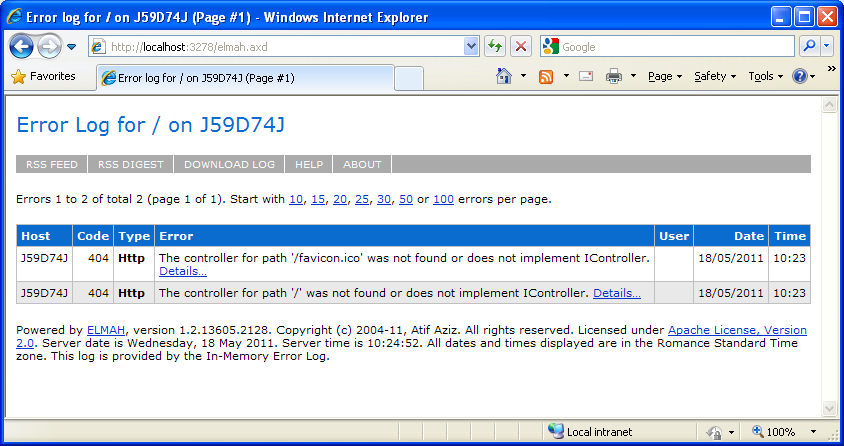
preCondition="integratedMode"

verb="POST,GET,HEAD" />

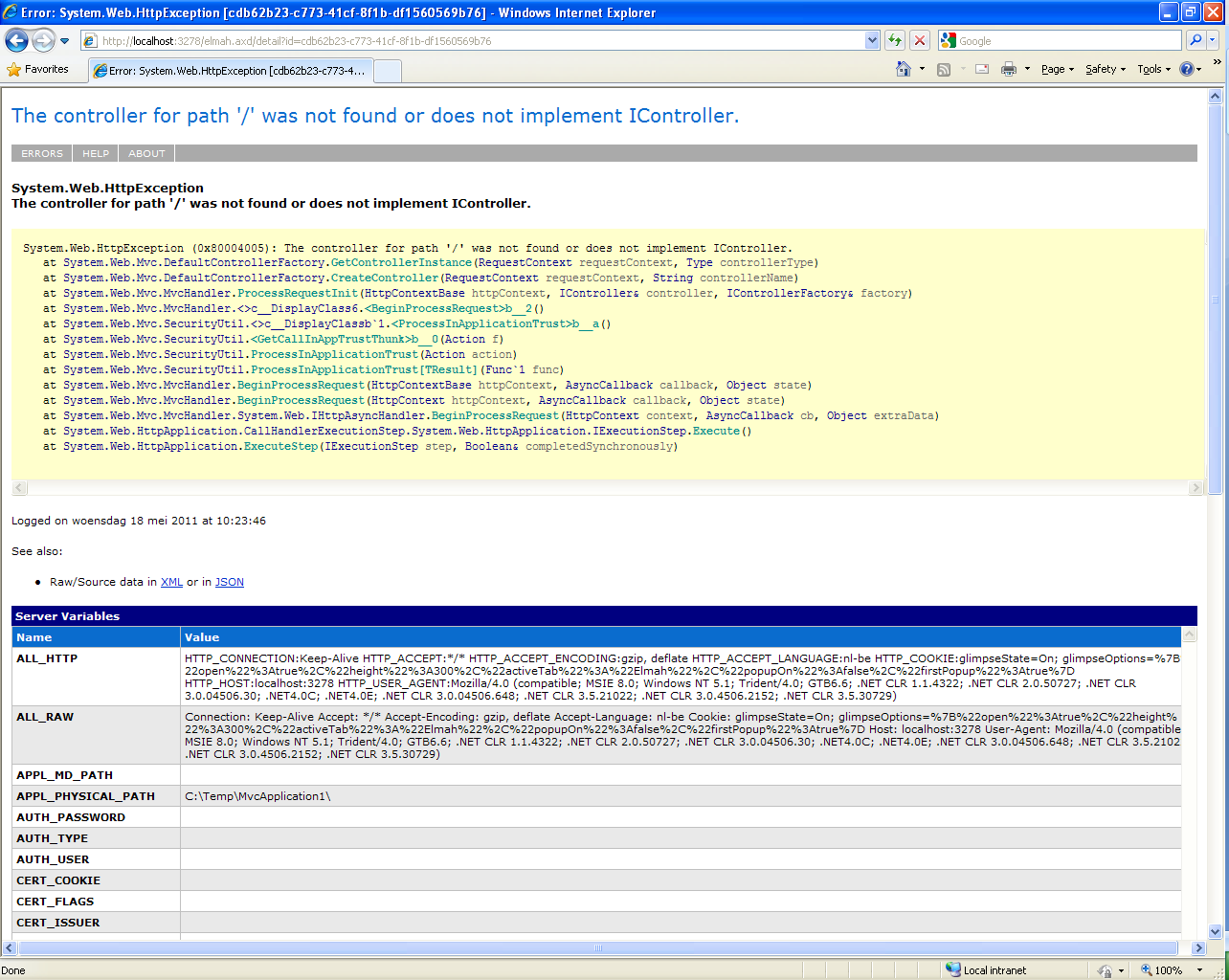
  </handlers>

</system.webServer>

Let’s take a closer look at the registration of the HTTP handler. We specified a path ‘elmah.axd’, this means that the handler will only kick in when we browse to that specific path. So, let’s launch our application and take a look. If I browse to <http://localhost/elmah.axd> I get this:



And oh, it even gets better. Lets click on the ‘details’ link next to the error message. This is where the magic happens:



This is really neat; we get a ton of information on what was going on when the error was thrown. If an application error occurred we can almost immediately pin-point the problem.

Obviously you don’t want to give access to this log to everyone that visits your website, as user might be able to spot vital problems in your code and exploit them. You can prevent access to this log in various ways (no remote access, through ASP.NET authorization …). More info on this topic can be found here: [Elmah: Securing Error Log Pages](http://code.google.com/p/elmah/wiki/SecuringErrorLogPages).

# Introducing Glimpse

Most of you probably know Firebug. And actually, [Glimpse](http://getglimpse.com/) is a lot like Firebug, except it’s implemented in JavaScript on the client side with hooks in to ASP.NET on the server side. What Firebug is for the client, Glimpse does for the server... in other words, a client side Glimpse into what’s going on in your server.

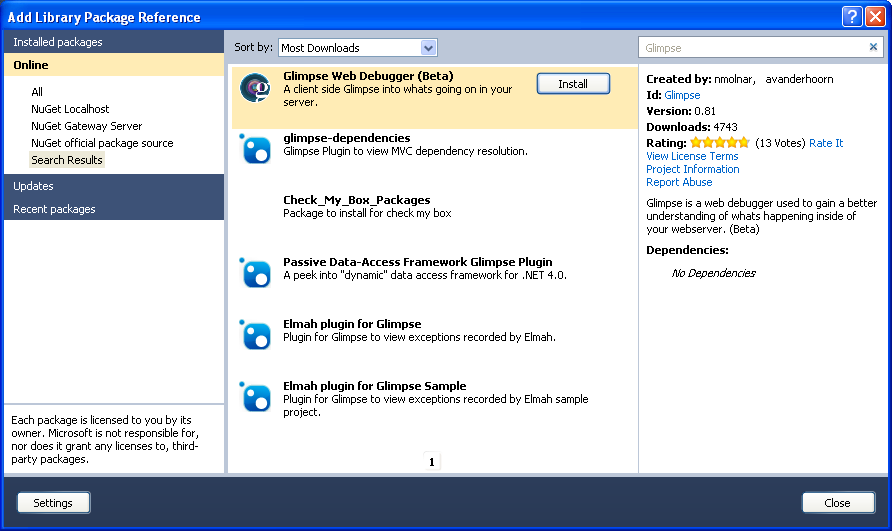
Currently Glimpse can only be used within web applications that target .NET 4.0; however the Glimpse team is currently working on a build that supports.NET 3.5 as well. As a matter of fact, Glimpse is not just a .NET only tool; currently it only works with ASP.NET MVC and Web Forms but eventually it will support Ruby on Rails, PHP and others.

For ASP.NET MVC 3 it currently already has a bunch of nice features, like:

* Ajax call tracing
* Route debugging
* Request inspection
* Server variables inspection
* Session inspection
* .NET Tracing (no more need for the trace.axd)
* …

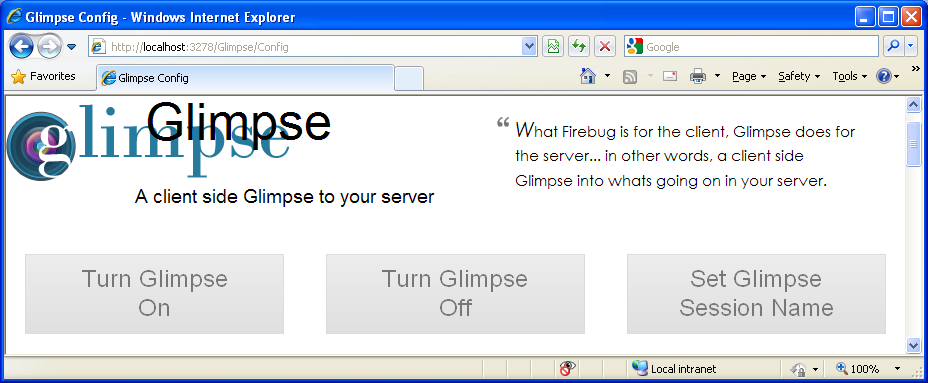
# Using Glimpse with ASP.NET MVC

Integrating Glimpse with ASP.NET MVC is very easy. It’s available as a [NuGet package](http://www.nuget.org/List/Packages/glimpse) so installing it literally takes about 30 seconds. Let’s have a look … Right click your web application and select ‘Add Library Package Reference…’ to pop up the NuGet Package Manager. Search for Glimpse and install it:

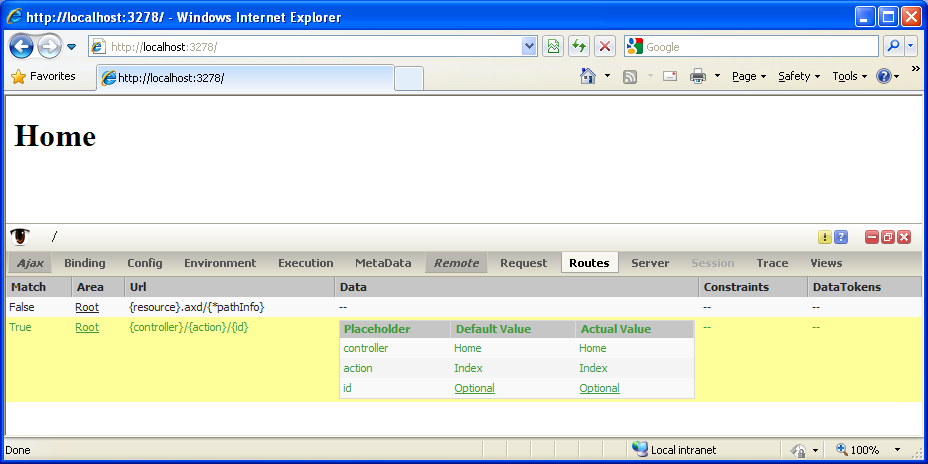


The Glimpse NuGet package will automatically reference the right assembly and it will add a configuration section to your web.config. The only thing you have to do to get it up and running is the following:

* Launch your application
* Browse to: http://localhost/Glimpse/Config
* Click ‘Turn Glimpse On’.



* Go back to you application, you should now see something like this:



What these Glimpse guys developed is really mind blowing. Currently Glimpse is still beta, but in my opinion it’s stable enough to integrate with all of your current ASP.NET projects.

# Elmah for Glimpse - Best of both worlds

Scott Hanselman, unofficially titled the funniest guy at Microsoft, recently featured Glimpse as [‘NuGet Package of the Week’](http://www.hanselman.com/blog/NuGetPackageOfTheWeek5DebuggingASPNETMVCApplicationsWithGlimpse.aspx). At the end of the article he says:

*‘Glimpse, along with ELMAH, is officially my favorite add-on to ASP.NET MVC. I'll be using it every day and I recommend you do as well.’*

So … wouldn’t it be cool to integrate Elmah with Glimpse?

## Plugin System

Glimpse has a really clean plugin system. Internally they use [MEF](http://mef.codeplex.com/) to discover extensions. The best of all is that writing a Glimpse plugin is really easy. All you have to do is:

* Implement the IGlimpsePlugin interface
* Decorate your class with the GlimpsePlugin attribute

Let’s take a look at my Elmah plugin for Glimpse:

[GlimpsePlugin]

public class ElmahPlugin : IGlimpsePlugin

{

public string Name

{

get { return "Elmah"; }

}

public object GetData(HttpApplication application)

{

if (application == null || application.Context == null)

return null;

var errorList = new List<ErrorLogEntry>();

var errorCount = ErrorLog

.GetDefault(application.Context)

.GetErrors(0, 15, errorList);

if (errorCount == 0)

return null;

var data = new List<object[]>

{

new object[]

{

"Host", "Code", "Type", "Error", "User", "Date", "Time"

}

};

data.AddRange(

errorList.Select(

errorEntry => new object[]

{

         errorEntry.Error.HostName,

errorEntry.Error.StatusCode,

errorEntry.Error.Type,

errorEntry.Error.Message,

errorEntry.Error.User,

errorEntry.Error.Time.Date.ToShortDateString(),

errorEntry.Error.Time.ToShortTimeString()

}));

return data;

}

public void SetupInit(HttpApplication application)

{

throw new **NotImplementedException**();

}

}

In the code sample above you see I implement the IGlimpsePlugin interface. The interface has the following methods:

* A getter for the Name of the plugin: this property will be used as the name of the tab in the Glimpse user interface.
* GetData method: accepts an HttpApplication parameter and returns an object. The object you return will be serialized into JSON, sent to the client and rendered by the Glimpse UI. How it will be rendered depends on the kind of object you return. More details about this can be found on the [Glimpse Protocol](http://getglimpse.com/protocol) page.
* SetupInit method: this is the place where you perform any initialization that needs to be done. This method will run once (when your plugin is loaded for the first time). If you want this method to be called by Glimpse, you will need to set the ShouldSetupInInit property on the GlimpsePlugin attribute to true: [GlimpsePlugin(ShouldSetupInInit=true)]

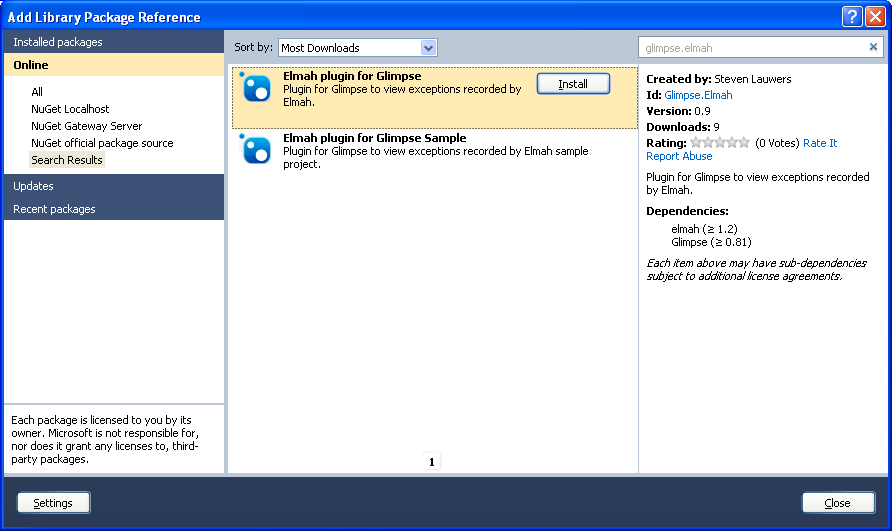
The GetData method is the place where you should make any decisions about what should be rendered and how you want to render it. If this method returns null, then your tab will be disabled in the Glimpse UI.

In the above implementation I return null if no HTTP context is available or when no errors have been logged yet. In case errors have been logged, I return them as a list of objects. The first entry in the list will be used as the title row; all subsequent rows are data rows. Currently only the last 15 errors are returned to the user.

## NuGet Package

The Elmah plugin for Glimpse is available as a [NuGet package](http://www.nuget.org/List/Packages/Glimpse.Elmah). To install the package, right click your web application and select ‘Add Library Package Reference …’ search for either ‘Glimpse’, ‘Elmah’ or ‘Glimpse.Elmah’ and the Package Manager will come up with the ‘Elmah plugin for Glimpse’.

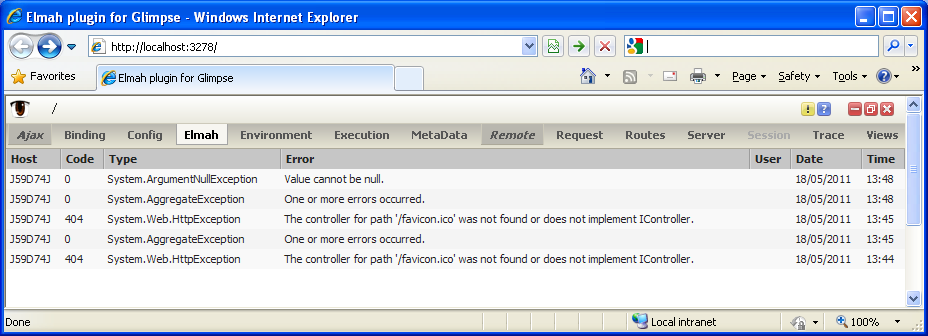
The package has dependencies to Glimpse and Elmah, so if you haven’t got these package installed yet then NuGet will get them automatically as well. If you didn’t have Elmah before, you might still have to configure it.



After installing the package you still have to include the Elmah for Glimpse client side script on your pages, preferable in your main master page.

<script src="<%: Url.Content("~/Glimpse/Resource/?resource=Pager") %>" type="text/javascript"></script>

Let’s run our application again and see what happens. Our Elmah plugin has been added as a reference to the project so it will be discovered by the Glimpse plugin system. Glimpse will load it into its UI and voila, there we have it:



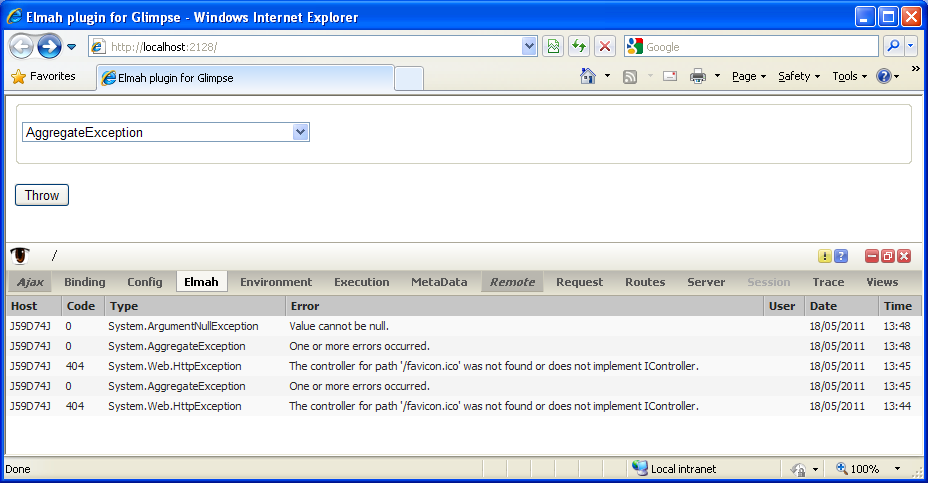
## NuGet Sample Package

The easiest to get started is by getting the [‘Elmah plugin for Glimpse Sample’ package](http://www.nuget.org/List/Packages/Glimpse.Elmah.Sample), which is also available through NuGet.

Just create a new empty ASP.NET MVC 3 application and add a library package reference to the Elmah plugin for Glimpse Sample package. The package will automatically:

* Install Glimpse
* Install Elmah
* Install the Elmah plugin for Glimpse
* Install jQuery
* Configure everything
* Create a new controller, model and view so you can throw some random exceptions.

After installing the package you’ve got yourself an application to throw exceptions! Cool huh!? OK, so let’s run the application, throw some exceptions and see what happens:



If you try the sample yourself, but you don’t see the Glimpse plugin you might still have to enable it. Please take a look at the ‘Using Glimpse with ASP.NET MVC’ section on how to do this.

## Future

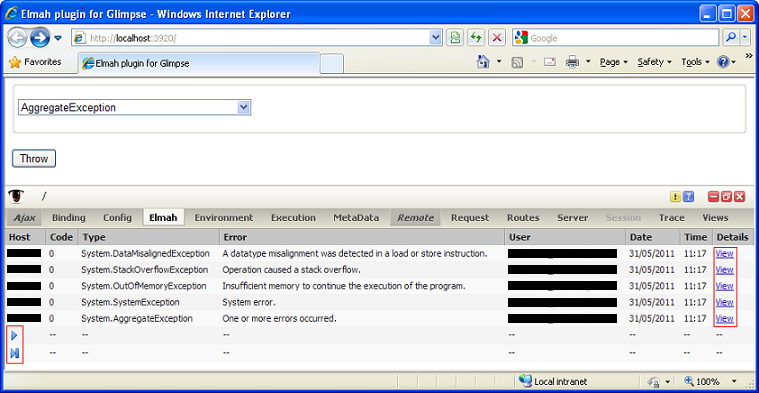
If there's any feature you would like to see implemented, you can always submit a feature request on the [project home page at CodePlex.](http://elmah4glimpse.codeplex.com/) . If you want to implement a feature yourself, you can fork the project and submit a pull request.  
  
Things I’m considering for the next version are:

* Adding some kind of configuration (page size, ...)
* Automatically include the client script

## Release Notes

06/03/2011 - 0.9.2.0  
- Updated to version 0.82 of Glimpse  
- Updated to version 1.2.0.1 of Elmah

05/31/2011 - 0.9.1.0  
- Added details link  
- Added pager



05/18/2011 - 0.9.0.0  
- Initial version