

ASP.Net Debugging and Logging



Glimse

Glimpse inspects web requests as they happen, providing insights and tooling that reduce debugging time and empower every developer to improve their web applications

Unique Perspective

While F12 tools like Firebug and proxy debuggers like Fiddler are extremely useful, only Glimpse provides diagnostics from the perspective of your server

Embedded Stock Components Category ComponentType Manage Components User Settings About User: au443291 Log off

Component type details

Edit

Back to List

| | |
|-----------------------|---|
| Component name | Arduino Due (Atmel ARM Cortex-M3) |
| Categories | • Arduino Development Boards/Shields |
| Location | |
| Component Type status | Available |
| Info | |
| Link to data sheet | http://au.dk |
| Manufacturer | |
| WikiLink | |

Manage components

Add one Component

Add many Components

| Comp. no | Serial no | Status | Admin comment | User comment |
|----------|-----------|----------------|---------------|--------------|
| 1 | 123asd | ReservedLoaner | Test | |
| 2 | 124asd | ReservedLoaner | | |
| 3 | qwe123 | ReservedLoaner | | |
| 4 | qwe123 | ReservedLoaner | | |
| 5 | 1234567 | Defect | Defect | |
| 6 | 1234567 | Loaned | | |
| 7 | 12qw | ReservedLoaner | | |

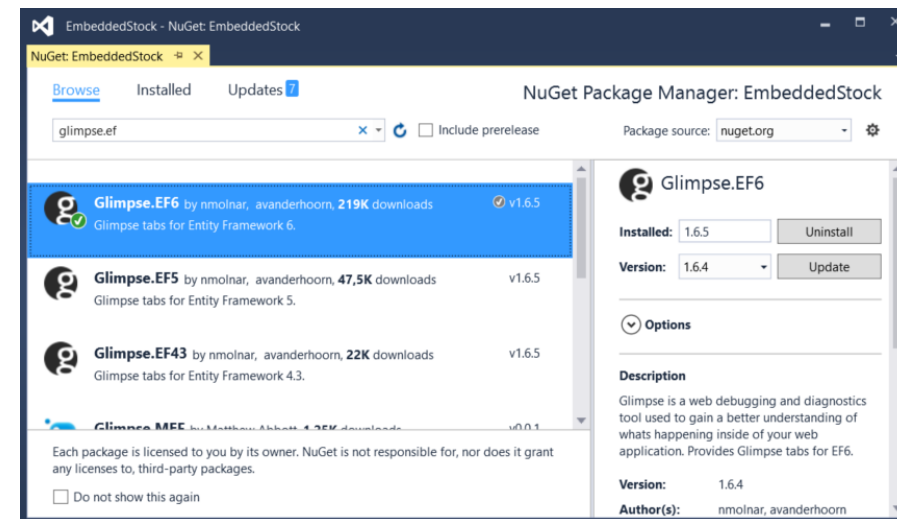
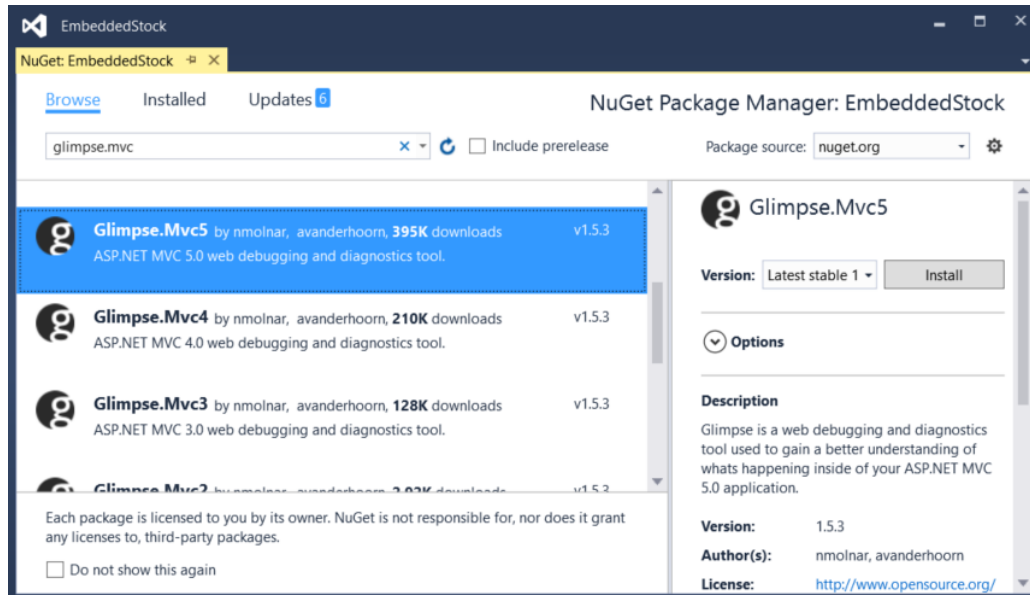
Glimse



Glimse How to

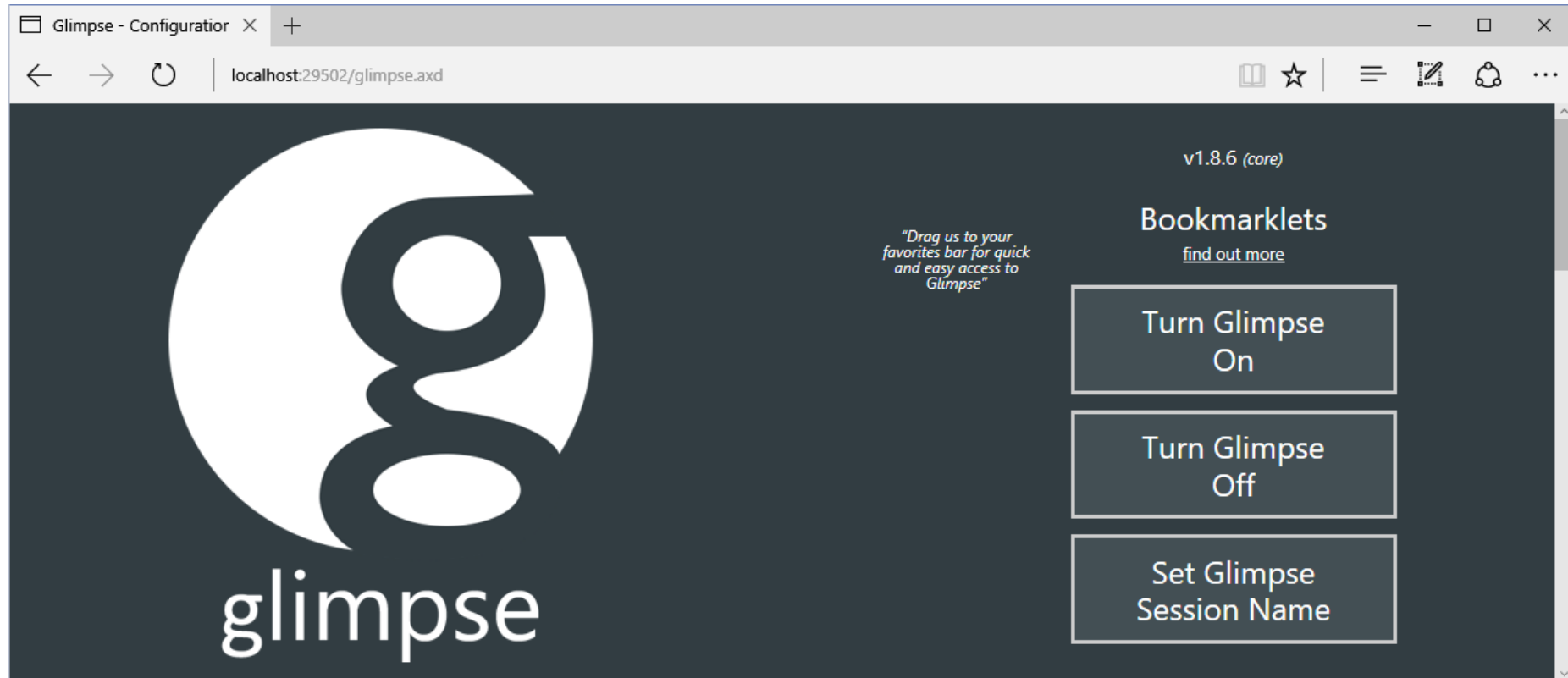
- Install Glimse.Mvc and optionally Glimse.EF with Nuget

```
PM> Install-Package Glimpse.Mvc5  
PM> Install-Package Glimpse.EF6
```



Enable Glimpse for localhost

- Navigate to `http://localhost:<port #>/glimpse.axd` and click the **Turn Glimpse On** button



Logging

Logging

- You write messages to the log as you would write to `Debug.WriteLine()`

NLog syntax

```
public class AccountController : Controller
{
    private ApplicationSignInManager _signInManager;
    private ApplicationUserManager _userManager;
    private StockDbContext context = new StockDbContext();
    private static Logger logger = LogManager.GetCurrentClassLogger();

    public async Task<ActionResult> Register(RegisterViewModel model)
    {
        ...
        logger.Error("AccountController.Register UserManager.CreateAsync failed: error={0}", result);
        ...
    }
    ...
}
```



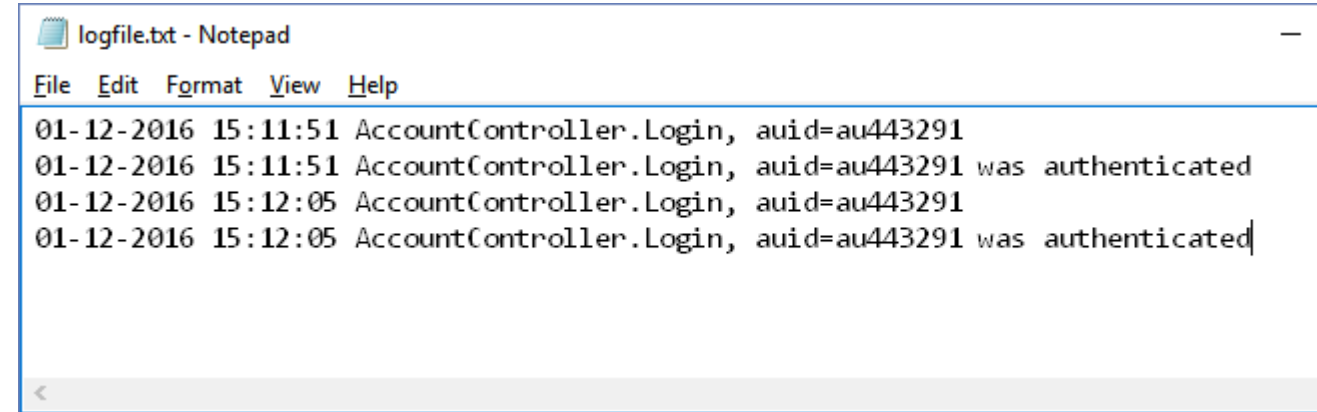
The diagram illustrates the NLog syntax for logging. A blue arrow points from the text "NLog syntax" to the `logger` variable in the code. Another blue arrow points from the text "log level" to the `Error` property of the `logger` object in the `Register` method.

Log levels

- Each log message has associated log level, which identifies how important/detailed the message is
- NLog can route log messages based primarily on their logger name and log level
- NLog supports the following log levels:
 - **Fatal** - very serious errors!
 - **Error** - error messages
most of the time these are Exceptions
 - **Warn** - warning messages
typically for non-critical issues
 - **Info** - information messages,
normally enabled in production environment
 - **Debug** - debugging information
less detailed than trace, typically not enabled in production environment
 - **Trace** - very detailed logs
which may include high-volume information
such as protocol payloads, typically only enabled during development

Log Configurations

- NLog (and many others) uses a configuration file to specify what log-levels to write (save) to which destinations
- NLog uses its own configuration file separate from Web.config



A screenshot of a Notepad window titled 'logfile.txt - Notepad'. The window contains four lines of log data. Each line starts with a timestamp '01-12-2016 15:11:51' or '01-12-2016 15:12:05', followed by the text 'AccountController.Login, auid=au443291'. The last two lines also include 'was authenticated'.

```
logfile.txt - Notepad
File Edit Format View Help
01-12-2016 15:11:51 AccountController.Login, auid=au443291
01-12-2016 15:11:51 AccountController.Login, auid=au443291 was authenticated
01-12-2016 15:12:05 AccountController.Login, auid=au443291
01-12-2016 15:12:05 AccountController.Login, auid=au443291 was authenticated
```

```
<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <targets>
    <target name="eventlog" xsi:type="EventLog" layout="${date}: ${message}"
            log="Application" source="Embedded Stock" />
    <target name="file" xsi:type="File" fileName="${basedir}/Logs/nLog.log"
            layout="${date}: ${message}" />
  </targets>
  <rules>
    <logger name="*" minlevel="Info" writeTo="eventlog" />
  </rules>
</nlog>
```


References & Links

- Glimse

- <https://www.asp.net/mvc/overview/performance/profile-and-debug-your-aspnet-mvc-app-with-glimpse>
- <http://getglimpse.com/>

- Logging

- The Art of Logging
<https://www.codeproject.com/articles/42354/the-art-of-logging>
- NLog
<https://github.com/NLog>
<http://www.strathweb.com/2012/06/using-nlog-to-provide-custom-tracing-for-your-asp-net-web-api/>
<http://justinp Davis.blogspot.dk/2010/04/logging-to-database-with-nlog.html>
- Log4Net
<https://logging.apache.org/log4net/>