Hosting Requirements

The Student Grouping Tool application consists of two main .NET projects: inBloom Client Library and **TeamMnMGroupingWebApp**. In order to compile and build the **TeamMnMGroupingWebApp** project, the inBloom Client library should be compiled and built first. The web application needs to be hosted in a Windows environment with IIS 7+ and .NET Framework 4.5+.

Continuous Integration

Consider using a continuous integration model to streamline the development and build process. This will make the production deployment a lot easier and faster. Windows Azure provides support for continuous integration with Git repositories. Every time code is checked in, the build process and unit tests should be run automatically, and if these are successful, the code will be published to Azure. For more information please visit http://www.windowsazure.com/en-us/develop/net/common-tasks/publishing-with-tfs/ and http://www.windowsazure.com/en-us/develop/nodejs/common-tasks/publishing-with-git/.

Nuget Package Manager

The Student Grouping Tool's .NET external dependencies (third-party libraries and tools) are handled by the **Nuget Package Manager**. When the source code is downloaded and opened in Visual Studio for the first time, these required dependencies might not be installed in the system. To install the necessary dependencies right-click on the project solution and click on **Manage Nuget Packages for Solution** (as shown in Figure 1). This will bring up the **Nuget Package Manager**. Select the required dependencies and click on **Install** to install them (shown in Figure 2).

The following is a list of all the project's .NET dependencies:

- ELMAH – used for error-logging
- Json.NET – used for serializing JSON objects into C# objects
- HTTP Client Libraries – used to make HTTP requests to REST endpoints
- ASP NET Web Pages 2 – core runtime required for ASP websites
- ASP NET Web Infrastructure – runtime libraries required for ASP MVC websites
- ASP NET MVC 4 – main runtime assemblies for ASP MVC websites
- ASP NET Razor 2 – runtime assemblies for the Razor HTML rendering engine
- ASP NET Web API – package for building HTTP services

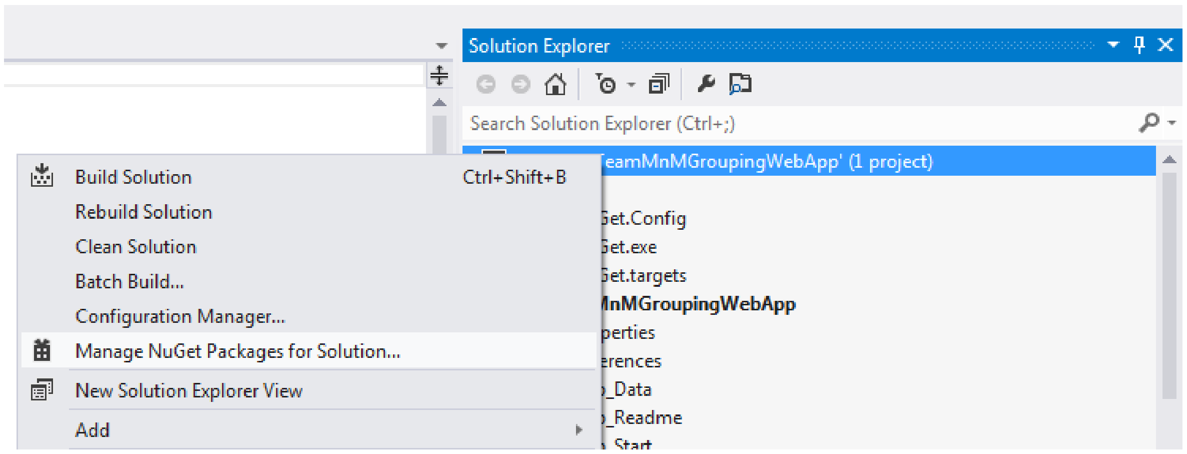
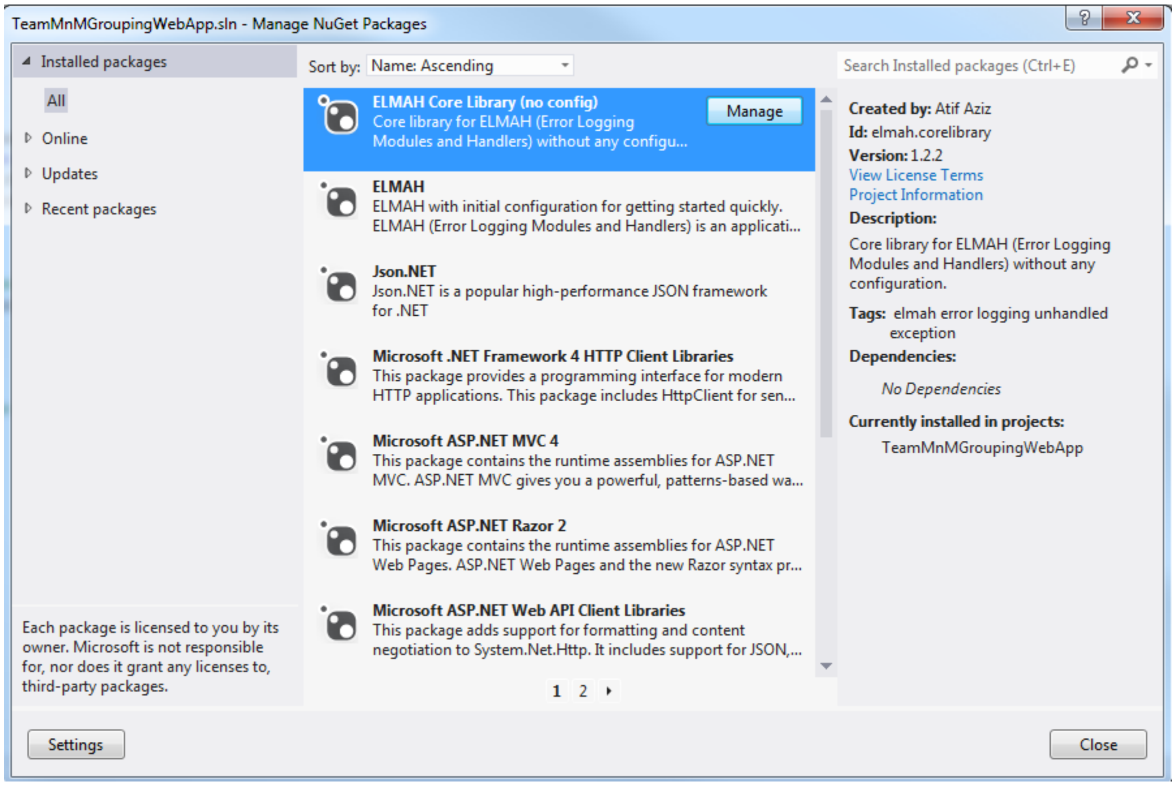


Figure – Selecting the Nuget Package Manager


TeamMnMGroupingWebApp.sln - Manage NuGet Packages

Installed packages — All | Online | Updates | Recent packages

Sort by: Name: Ascending

- **ELMAH Core Library (no config)** — Core library for ELMAH (Error Logging Modules and Handlers) without any configu... [Manage]
- **ELMAH** — ELMAH with initial configuration for getting started quickly. ELMAH (Error Logging Modules and Handlers) is an applicati...
- **Json.NET** — Json.NET is a popular high-performance JSON framework for .NET
- **Microsoft .NET Framework 4 HTTP Client Libraries** — This package provides a programming interface for modern HTTP applications. This package includes HttpClient for sen...
- **Microsoft ASP.NET MVC 4** — This package contains the runtime assemblies for ASP.NET MVC. ASP.NET MVC gives you a powerful, patterns-based wa...
- **Microsoft ASP.NET Razor 2** — This package contains the runtime assemblies for ASP.NET Web Pages. ASP.NET Web Pages and the new Razor syntax pr...
- **Microsoft ASP.NET Web API Client Libraries** — This package adds support for formatting and content negotiation to System.Net.Http. It includes support for JSON,...

Created by: Atif Aziz
Id: elmah.corelibrary
Version: 1.2.2
View License Terms
Project Information
Description:
Core library for ELMAH (Error Logging Modules and Handlers) without any configuration.
Tags: elmah error logging unhandled exception
Dependencies:
No Dependencies
Currently installed in projects:
TeamMnMGroupingWebApp

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Figure - Nuget Package Manager

Caching

Caching is used throughout the Student Grouping Tool to reduce the number of API calls and provide the user with a more responsive user interface. Caching is done using the HttpContext.Current.Cache, which caches objects in memory across all sessions. The objects that are currently being cached are all the DisplayObjects:

- CohortDisplayObject
- SectionDisplayObject

- StudentDisplayObject
- GroupingDisplayObject

Please refer to the Developer Documentation for more details regarding these objects. These objects are passed to the UI through the AJAX calls. Every time there is an update to a cohort, that cohort will be removed from the cache so that a new updated copy is cached.

Error Logging

Uncaught runtime exceptions thrown by the application are logged using the elmah library. The error logs can be accessed through the path root/elmah.axd. For security purposes remote access has been disabled, meaning that the logs can only be accessed locally (i.e. localhost) or by manually downloading the log files.

The elmah configuration can be found in the web.config file. You can specify which directory to save the log files to, as well as the path to access the elmah error logs screen. For more information on elmah please visit:
http://code.google.com/p/elmah/wiki/DotNetSlackersArticle.

If the web application is hosted on the cloud, such as Windows Azure, then instead of accessing the logs locally, the logs can be downloaded and consolidated into reports using the Elmah Log Download tool at:
http://code.google.com/p/elmah/wiki/ErrorLogDownloadApplications.

Security Considerations

Every time there is a request to the **HomeController**, the **HomeController** checks to see if there is an **access token** from the user. This **access token** is used to call the inBloom REST APIs to retrieve student and cohort data. If the **access token** is not present, the request is redirected to the inBloom authentication page.

Additionally the application automatically logs out after the user has been idle for 20 minutes and redirects the user to the login page in order to protect the student data. The idle timeout period is specified in the **groupSelectionMain.js** and **multipleGroupsEdit.js** files.

Disaster Recovery Guidelines

It is recommended that the application be hosted on various geographical locations for both load balancing and redundancy. In case one server fails, another one can take on the additional user requests. If the application is hosted on a Cloud environment, such as Windows Azure, then the cloud provider will take care of application availability,

In terms of data recovery, this application does not store any data. All data is retrieved from the inBloom data stores, except for the lesson plans stored on the local file system. The data can be modified and changes will be saved back to their respective servers.