Creating Modules

Overview

You can produce a module in different ways:

- Create an entire module from scratch.
- Start with module development templates, such as:
  - DNN 8 Templates
  - Chris Hammond's DotNetNuke Module and Theme Development Templates (Installation instructions)
- Customize a pre-existing module. Hundreds of third-party modules and themes are available from these sources:
  - For sale: DNN Store - Some developers provide versions that include the source code.
  - Open source and free:
    - DNN Forge
    - DNN at Codeplex
    - DNN at GitHub

You can also use different programming frameworks (Web Forms, MVC, SPA) and languages (C#, VB) to create your module.

How to Create a Module

Prerequisites

- A local DNN installation with **Host** permissions.
- Visual Studio 2015 is the recommended IDE for developing DNN modules.

Steps

1. **Create a new module or modify an existing one.**
   - Create a very simple module inside a DNN installation.
   - Create a Web Forms module using templates.
   - Create an MVC module using templates.
   - Create a SPA module using templates.
2. **Package your module.**
3. **Test your module.**
   a. Install the package.
   b. Apply the module to a pane in a new webpage in your DNN website.
   c. Check how the module behaves in different browsers.

   Tip: As part of your tests, run your module through the DNN Extension Verification Service (EVS) to perform the compatibility tests in three areas:

   - **Module packaging**. EVS verifies:
     - That a valid **.dnn** manifest file exists. EVS throws an error, if a required section is

        missing, or a warning, if an optional section is missing.
        - That all files listed in the manifest exist in the package.
        - That all files included in the package are listed in the manifest.
    - **Data layer**. EVS verifies:
        - That core database objects were not modified.
        - That any SQL scripts in the module are compatible with SQL Azure and can execute without errors. If Azure-incompatible SQL scripts are found, EVS generates Azure-compatible versions of those scripts and makes them available in a zip file.
        - That SQL's {databaseOwner} and {objectQualifier} are correctly used.
        - That the uninstall script completely removes all objects added by the installation script.
    - **Assemblies**. EVS verifies:
        - That no assembly errors exist.
        - That every assembly reference points to an assembly that exists either in DNN or in the .NET global assembly cache (GAC). If the assembly is not found, EVS returns an error.

See Also

- Contents of a Module Package
- DNN 8 API Reference

Sources

- DNN Wiki: Module Development
- DNN Community Blog: Module Development series by Clinton Patterson
- DNN Community Blog: Extension Verification Service (EVS) and Extension Verification Service (EVS) Update by Nathan Rover