Progress

Upgrading Sitefinity CMS

Best Practices

WHITEPAPER

Upgrading Sitefinity CMS – Best Practices

Prepare for the Upgrade

Understanding the task at hand and spending sufficient time to prepare increases your chance of success and reduces the possibilities of unexpected problems.

1. Plan sufficient time for the upgrade. Your project size, level of customization, automation and so on might affect the needed time to upgrade. Planning sufficient time ensures there is enough time to resolve any unexpected issues that occur. If the upgrade completes smoothly sooner than expected, you can use the remaining time for user acceptance testing.
2. Review the release notes of all major versions between your current version and the target version. You'll get valuable information about the expected new product functionality and changes in existing features behavior.
   - Review the API breaking changes between your current Sitefinity version and the version you are targeting for the upgrade.
     We strive to minimize API breaking changes, however refactoring sometimes requires us to do so. Look for usage of the affected APIs in your project and build a plan for modifying any affected code after the upgrade. API changes are among the most common reasons for post-upgrade issues on customized projects. Anticipating them and having a clear plan for addressing the changes will make your upgrade much smoother.
   - Review the Database changes between the versions.
     While not required, it is good to be familiar with the database changes that will be in place after the upgrade. The Sitefinity upgrade log file *(UpgradeTrace.log*, located in *~/App_Data/Sitefinity/Logs)* reflects all changes applied to the database during upgrade and their status.
3. Check whether the target Sitefinity version is compatible with the .NET version running on your machine, and install any new version if required.
4. Install the target Sitefinity version and create a new project with that version locally.
5. Make sure it is working properly. This will help isolate whether any issues pertain to your project specifically, or your environment needs additional configuration to meet the new version requirements.
6. Make a test deployment of the empty project on the target staging/test environment. Ensure the empty project deploys smoothly. Try to resolve any

deployment issues. If you require assistance, open a support ticket for help. Ensuring an empty project deploys successfully will give you the confidence to later deploy the actual upgraded project more easily and only deal with any project-specific issues.

7. Make backups of the project files and corresponding database(s).
   Sitefinity upgrade is a non-reversible process, thus having a backup from a point in time right before the upgrade is a must. It enables you to restore your website to a backed up version if anything unexpected happens during the upgrade process, and guarantees data loss prevention.
8. Restore a copy of your website project (and corresponding database(s)), that will be upgraded, locally.
9. Change all connection strings and make them point to a local copy of the database(s).
   - It might be a good idea to prohibit access of the local machine to staging and production database servers.
     This will ensure an error is thrown in case Sitefinity tries to access any of those resources. Once the upgrade executes, the Sitefinity database is modified irreversibly (downgrade is not supported), thus making sure your test project is not connected to the live database is a must.
10. Ensure the pre-upgraded project builds and runs successfully on the local machine.
    This way you verify the healthy project state prior to the upgrade and can isolate any pre-upgrade issues.
11. Decide on an upgrade mechanism (Project Manager or NuGet packages).
    Note that if the project is already using NuGet packages the upgrade needs to be done with NuGet packages.
12. Stop the local site from IIS.
    - If upgrading with Sitefinity Project Manager, and you have the project open in Visual Studio as well, close Visual Studio as well.
      This prevents potential lock of the Sitefinity project file (*SitefinityWebApp.csproj*) and ensures that of Sitefinity Project Manager can update the file if needed.
13. Empty the Sitefinity logs folder (*~/App_Data/Sitefinity/Logs*). You might need to stop the site from IIS/ Visual Studio/Sitefinity Project Manager, if the running worker process has locked some of the log files and prevents you from moving/deleting the old logs.
    It's best to have all logs generated from the time of the upgrade. This way you can identify more easily any upgrade or post-upgrade logged errors. If using custom logging mechanism (for example ELMAH), either stop it for the upgrade procedure, or make sure the local application can write logs and that they are accessible.

Perform the Upgrade

1. Use Sitefinity Project Manager or NuGet to replace the Sitefinity compiled logic. Depending on the selected upgrade mechanism, replacing the **.dll* files can happen in the following manner:
   - Use the Sitefinity Project Manager Upgrade option to directly replace the existing *.dll* files in the /bin folder of the selected project with new ones.
   - If you are using NuGet, from your Visual Studio use the Manage NuGet Packages for Solution option of Nuget Package Manager or directly update the package using the Package Manager Console. Then build your project, so the Visual Studio build replaces the *.dll* files in the /bin folder of the Sitefinity project.

   Replacing your current Sitefinity compiled logic with the new version means that your site will start using the new version when it runs.

2. Modify the *web.config*
   - When upgrading your website through NuGet packages to Sitefinity versions 11.0 and later the Sitefinity NuGet packages adjust the *web.config* automatically. If upgrading to a version prior to Sitefinity 11.0, the upgrade with NuGet packages does not modify the web.config. Apply the changes manually following the Upgrade instructions in the Sitefinity documentation (https://www.progress.com/documentation/sitefinity-cms/upgrade).
   - When upgrading your website through Sitefinity Project Manager, it prompts you to apply the *web.config* changes automatically. Follow the instructions to get your website *web.config* modified accordingly.

   Sitefinity is an AP.NET WebApplication. When you modify the *web.config* you reflect any changes in ASP.NET functionality introduced in the new Sitefinity version and ensure any new HTTP modules registrations, authentication mechanism changes, and so on are handled with the upgrade.

3. Adjust the project references
   - When upgrading with NuGet packages to Sitefinity versions 11.0 and later, the Sitefinity NuGet packages automatically adjust the project references.
   - When upgrading your website through Sitefinity Project Manager, it prompts you to adjust the project references automatically. Follow the instructions to get your project references modified accordingly. If upgrading to version prior to Sitefinity 11.0, these changes need to be done manually before upgrading the NuGet packages. Follow the

Upgrade instructions in the Sitefinity documentation (https://www.progress.com/documentation/sitefinity-cms/upgrade).

By adjusting your project references you apply any relevant changes to the *SitefinityWebApp.csproj* file introduced in the new Sitefinity version, for example introducing a new assembly.

4. Address any API breaking changes in your code.
   If you have extended your Sitefinity website using the Sitefinity API, you must check the API changes in Sitefinity CMS documentation and identify whether any of the APIs you are using have been changed. Address any of the changes that apply to your code, custom project references, and so on. This helps ensure your custom logic is compatible with the new Sitefinity version, any necessary binding redirects are in place and so on, prior to running your upgraded project for the first time.
5. Build the project, verify it builds successfully.
6. Run your site to execute the upgrade scripts.
7. Once your Sitefinity website runs for the first time with the upgraded assemblies, each Sitefinity module that needs to be upgraded runs its upgrade scripts, modifies the database schema, and finally updates its entry in the database and configuration files to indicate that it's now running on the new version.
8. Apply the new license.

Test the Upgraded Site

1. Ensure the frontend of the site looks and feels the same with no obvious differences.
   This way you ensure the upgrade has not affected the website pages, no exceptions are thrown by the widgets on your pages, and so on.
2. Make sure you can successfully log into the backend.
   Some Sitefinity versions introduce changes in the authentication mechanism for better security and integration. Successful login to the backend tests whether these changes have applied successfully.
3. Check the *Administration -> Modules & Services* page for any failed modules.
   If a module has failed initializing after the upgrade this indicates potential failed upgrade scripts or problems with custom logic that prevents it from starting.
4. Check the Sitefinity logs. Pay special attention to the *UpgradeTrace.log* and *Error.log* and check if there are any records indicating failed operation. Contact the Sitefinity Support team if you require further assistance interpreting or addressing any errors.

Deploy the upgraded project

1. Plan for a deployment window
   Ideally, deployment should be done during times of no or low(er) traffic to the site. This way you ensure ample time for the deployment and resolving any issues after that.
2. Either deploy the upgraded project files and restore the upgraded database on the target environment, or deploy just the project files and have the upgrade scripts execute and modify the live database automatically on the target environment.
   - If you decide to deploy both the upgraded project files and database:
     - Stop the production website.
     - Deploy the upgraded project files to the web server.
     - Restore a copy of the upgraded database on the production database server.
     - Modify the connection string to point to the correct database.
   - If you decide to deploy only the upgraded project files and have the upgrade scripts execute on the production database:
     - Stop the production website.
     - Deploy the upgraded project files to the web server.
     - Modify the connection string to point to the correct database.
     - Run the website to have the upgrade scripts execute against the database.

About Progress

Progress (NASDAQ: PRGS) offers the leading platform for developing and deploying strategic business applications. We enable customers and partners to deliver modern, high-impact digital experiences with a fraction of the effort, time and cost. Progress offers powerful tools for easily building adaptive user experiences across any type of device or touchpoint, award-winning machine learning that enables cognitive capabilities to be a part of any application, the flexibility of a serverless cloud to deploy modern apps, business rules, web content management, plus leading data connectivity technology. Over 1,700 independent software vendors, 100,000 enterprise customers, and two million developers rely on Progress to power their applications. Learn about Progress at www.progress.com or +1-800-477-6473.

Rev 2019/01 | RITM0034642

Progress