Source Code Control Recommendations

For

OpenSpan Studio v4.1 and Earlier

September, 2009

Introduction

OpenSpan Studio is a design and development environment that is used to integrate applications as well as create composite applications. It does this using a graphical interface to assist designers. The result of the development tool is a set of XML files which define the applications, automations and workflows.

Source code control is used to manage multiple revisions of projects including the code for the project as well as the metadata which describes how the project is built, deployed, and configured. Source code control systems keep the source, the data describing the source and the history of changes in a *source code repository* which is usually a database on a server.

OpenSpan Studio versions up to and including version 4.1 do not have built-in integration with source control repositories. Studio also has behaviors which require developers to strictly adhere to certain procedures when using source code control.

Difficulties using OpenSpan Studio with Source Control

The issues that require developers to adhere to procedures include:

- OpenSpan Studio versions prior to v4.1 will reset the read-only flag on a file and allow that file to be overwritten by changes. This is not an issue for v4.1 and later.
- Studio writes to the *project*.xml file when the solution is closed. If there are no changes to the project, then the file just has its timestamp changed. The problem is if Studio has a project opened and a newer project file is loaded from source code control. The newer *project*.xml file will be overwritten when Studio closes the project.
- Studio may make minor adjustments to the location of controls on the visual design screen by moving items a few pixels. This has no effect on the operation of the solution, but it will cause source control systems to flag a change.
- Deployment packages should not be kept in source control except as an archival backup since the package is compressed and source control systems are unable to perform merges.

- Files with the extension of *.debug* contain information relating to breakpoints, watches and general debugging requirements. These do not need to be kept in source control as their inclusion will cause developers to essentially "share" breakpoints.

Recommendations

OpenSpan recommends the following procedures to assist using source control. These recommendations are in additional to standard source control procedure such as verifying a solution works before checkin, and frequent checkin to and updates of local source from source control.

- **No solution should be opened in OpenSpan Studio while performing any source code repository operations**. When a solution is open in Studio, source control should not be accessed.

  Studio should either have its project closed or Studio should be closed before working with the source code control system. This is needed since Studio will overwrite files it believes are modified when it does a save and will **always** overwrite the *project*.xml file when it closes.

  The practical aspect of this is that files should first be checked in/out and then Studio can open the project for use. When ready to do further check in/out, the Studio solution should be closed before working with source code control.

- Developers should always check out and lock files they need to change so other developers will be unable to check copies in while the first developer is working with it. All files should be checked in when not being actively edited.

  This prevents the situation where one developer has the file checked out and someone else tries to check a file in. The source code control system must support a method to lock other users from changing the file in the repository.

  The file for which this is the greatest issue is the *project.*xml file as it contains a description of the overall project. Following this recommendation requires developers to coordinate changes that affect the project structure.

- The project should be organized where each application has its own folder which contains the application's adapter and automations. Items which are shared should likewise be kept in their own folder.

  A structure where each application is kept somewhat separate allows the source code system to more easily handle any changes and merges in the *project.*xml file that describes the project since developers will be modifying different sections of the file. However, it is still advised to have only one developer have the project file checked out at a time.