

Control de versiones con TFS

Parte 2

Joyas de TFVC

Spanish translation
by Juan María Laó Ramos

Visual Studio ALM Rangers

La información contenida en este documento representa la visión de Microsoft Corporation sobre los asuntos analizados a la fecha de publicación. Dado que Microsoft debe responder a las condiciones cambiantes del mercado, no debe interpretarse como un compromiso por parte de Microsoft, y Microsoft no puede garantizar la exactitud de la información presentada después de la fecha de publicación.

Este documento es sólo para fines informativos. MICROSOFT NO OFRECE NINGUNA GARANTÍA, EXPRESA, IMPLÍCITA O LEGAL, EN CUANTO A LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO.

Microsoft publica este documento bajo los términos de la licencia Creative Commons Attribution 3.0 License. Todos los demás derechos están reservados.

© 2014 Microsoft Corporation.

Microsoft, Active Directory, Excel, Internet Explorer, SQL Server, Visual Studio, and Windows son marcas comerciales del grupo de compañías de Microsoft.

Todas las demás marcas son propiedad de sus respectivos dueños

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, you should not interpret this to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Microsoft grants you a license to this document under the terms of the Creative Commons Attribution 3.0 License. All other rights are reserved.

© 2014 Microsoft Corporation.

Microsoft, Active Directory, Excel, Internet Explorer, SQL Server, Visual Studio, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Índice

Prólogo.....	4
Introducción	5
Control de Versiones de Team Foundation.....	6
Estructura de directorios del control de versiones.....	6
Workspaces	9
Merging	13
Seguimiento y gestión	13
Merges huérfanos.....	14
Hands-on Lab (HOL) – Workspaces & Merging	17
Prerrequisitos	17
Ejercicio 1: Crea un workspace local.....	18
Ejercicio 2: Usa el workspace local en modo online	19
Ejercicio 3: Usa el workspace local en modo offline	23
Ejercicio 4: Investiga las herramientas de comparación y merge.....	23
Apéndice: Configuración del entorno.....	29
FAQ.....	30
Conclusión	31

Prólogo

Desde la primera versión de la guía de Branching de TFS, han cambiado muchas cosas en el mundo del control de versiones. Las soluciones de control de versiones están por todas partes, y muchas de ellas incluyen integración con builds, seguimiento de proyectos, y otros servicios. Los sistemas de Control de Versiones Distribuidos ya no son un nicho, y han cambiado lo que para muchos desarrolladores significa “tener versionado el código”. Hay muchos más desarrolladores que usan control de versiones que antes – y eso es algo fantástico para miles de millones de usuarios finales de su software.

Más desarrolladores usando control de versiones también significa, ahora más que nunca, que la industria necesita guías sólidas, prácticas y fáciles de entender. Esta guía, y todas las que la precedieron, se esfuerzan en eso – ofrecer una guía sobre el control de versiones que todo equipo de desarrollo necesita para ser efectivo y a la vez accesible y flexible. En esta última edición, hemos racionalizado la orientación y simplificado conceptos con el objetivo de ayudar a equipos de todas las formas y tamaños para que consigan una estrategia que les permitan conseguir la flexibilidad y agilidad que los equipos de desarrollo modernos necesitan.

También quiero decir que esta guía no podría ir por esta versión sin los lectores. Gracias a todos los que habéis contribuido con el feedback y las ideas que nos han ayudado a darle forma a esta guía durante los últimos años. Como en versiones anteriores, si ves algo en esta guía que te gustaría cambiar o mejorar, ¡háznoslo saber!

¡Feliz versionado!

Matthew Mittrik – Program Manager, Cloud Dev Services

Introducción

El objetivo de esta guía es ofrecer una visión y una orientación práctica sobre cómo usar las funcionalidades de Team Foundation Version Control (TFVC), como los workspaces, merging, y la nueva funcionalidad Code Lens. Es una de las guías adicionales como se indica en el capítulo “¿Qué hay de nuevo?” en la guía **Control de Versiones con TFS Parte 1 – Estrategias de Branching**.

A quién va dirigida

En esta guía, nos centramos en **Garry**, el jefe de equipo, **Doris**, la desarrolladora, y **Dave**, el administrador del TFS. Leed [ALM Rangers Personas and Customer Profiles](#)¹ para más información de estas y otros personajes.

Visual Studio ALM Rangers

Los Visual Studio ALM Rangers ofrecen una guía profesional, experiencia práctica y proporcionan soluciones a la comunidad ALM. Son un grupo especial compuesto por miembros del grupo de producto de Visual Studio, de Microsoft Services, de Microsoft Most Valuable Professionals (MVP) y de Visual Studio Community Leads. La información sobre sus miembros está disponible aquí [online](#)².

Colaboradores

Anna Galaeva, Gordon Beeming, James Waletzky, Malcolm Hyson y Michael Fourie.

Un agradecimiento especial a los equipos e ALM Ranger que sentaron las bases con las versiones v1 y v2: Anil Chandr Lingam, Bijan Javidi, Bill Heys, Bob Jacobs, Brian Minisi, Clementino de Mendonca, Daniel Manson, Jahangeer Mohammed, James Pickell, Jansson Lennart, Jelle Druyts, Jens Suessmeyer, Krithika Sambamoorthy, Lennart Jansson, Mathias Olausson, Matt Velloso, Matthew Mitrik, Michael Fourie, Micheal Learned, Neno Loje, Oliver Hilgers, Sin Min Lee, Stefan Mieth, Taavi Koosaar, Tony Whitter, Willy-Peter Schaub, y la comunidad ALM.

Uso del código de ejemplo, erratas y soporte

Todo el código Fuente y la revisión de esta guía están disponibles para su descarga a través del sitio [Version Control Guide](#)³. Puedes contactar con el equipo usando el foro de CodePlex.

Más ALM Rangers y otros Recursos

[Understanding the ALM Rangers](#)⁴

[Visual Studio ALM Ranger Solutions](#)⁵

¹ <http://vsarguidance.codeplex.com/releases/view/88001>

² <http://aka.ms/vsarindex>

³ <http://aka.ms/treasure18>

⁴ <http://aka.ms/vsarunderstand>

⁵ <http://aka.ms/vsarsolutions>

Control de Versiones de Team Foundation

Estructura de directorios del control de versiones

Los directorios son una herramienta en TFS para organizar y ofrecer un contenedor de branches, otros directorios, o archivos. Es importante tener una estructura de directorios consistente en todo el árbol de fuentes. Un conjunto de convenciones claro ayuda a los miembros del equipo a encontrarse cómodo en áreas del proyecto que no conozcan. También nos permite usar las mismas prácticas de build y scripts en todas las funcionalidades y componentes.

La mejor estructura de directorios depende del modelo de branching que el equipo haya elegido. (Hemos hablado sobre los diferentes modelos de branching en la Parte 1 de esta guía.) Por simplicidad, esta sección asume que estamos usando la estrategia **Service & Release Isolation**, una de las estructuras de branching más común, aunque hablaremos de otras estrategias también.

Hablaremos de dos niveles de estructura de directorios: una global para todo el proyecto, y otra local para las funcionalidades y componentes individuales.

Estructura Global de Directorios

NOTA

Con “Estructura Global de Directorios” nos estamos refiriendo a la organización general de nuestro repositorio de código que podemos ver en el Source Control Explorer. Este tema es específico al control de versiones de TFS y no es aplicable a Git.

Mantenlo simple. Hay bastantes estructuras de directorios posibles, y no hay una única respuesta correcta. Reflexiona un poco sobre cómo organizar el repositorio, y reevalúa la estructura regularmente para asegurarte de que sigue ajustándose a las necesidades de tus usuarios. Dos puntos de los que partir para establecer una estructura de directorios suelen ser las releases y los branches.

Ordenando por Release

Vamos a asumir que tenemos un producto llamado “Product A” con dos releases previas y un equipo trabajando en la release actual, si ordenamos por release podríamos obtener una estructura similar a la siguiente:

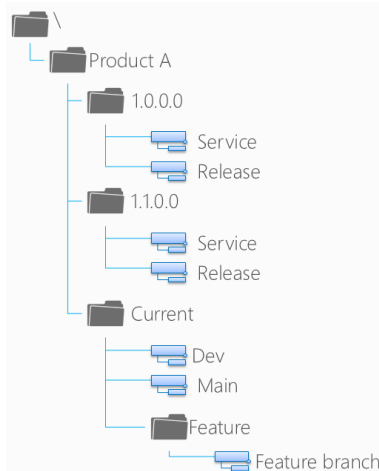


Figura 1 – Directorios del sistema de control de versiones ordenados por release

Ventajas

- Es fácil encontrar los branches de release (refleja fielmente la estructura de branches)
- Si un desarrollador no está trabajando en una release en concreto, es fácil ocultar el directorio, reduciendo así el ancho de banda y el espacio de disco a la hora de hacer un “get”
- Hace que la estructura de branches sea más intuitiva ya que la gente suele pensar primero en la release

Desventajas

- El directorio de Release no se obtiene muy a menudo, por lo que tendremos que ocultar varios branch de release
- Si usamos branches de funcionalidad, estarán muy abajo en la jerarquía. Podemos alcanzar las limitaciones de TFS con respecto a la longitud de los path.

NOTA

El directorio "Current" no es estrictamente necesario, pero mantiene la consistencia con otros temas relacionados con los directorios.

El directorio "Feature" sólo sería necesario para aislar una funcionalidad (por ejemplo, una funcionalidad con alto riesgo de no ser publicada). Deberíamos considerar poner el directorio "Feature" al mismo nivel que "Current" si no dependen de una release.

Ordenando por propósito

Vamos a asumir que tenemos un producto llamado "Product A" con dos releases previas y un equipo trabajando en la release actual, si ordenamos por propósito:

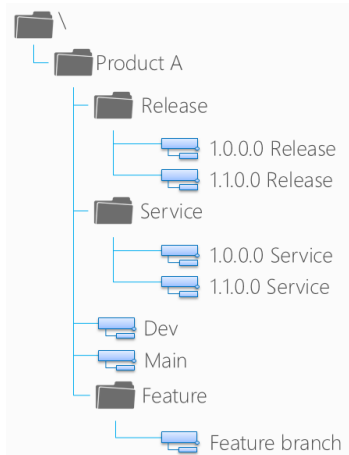


Figura 2 – Directorios del sistema de control de versiones ordenados por propósito

Ventajas

- En la mayoría de las ocasiones, un desarrollador nunca tocará el directorio "Release", así que podemos ocultar el directorio entero.
- El segundo nivel del árbol es coherente con el propósito de cada branch

Desventajas

- Es probable que queramos ocultar branches individuales de servicio, dependiendo de la release en la que estemos trabajando
- No refleja la estructura de branches de la manera más precisa

Ignora el nivel "Product A" del árbol si sólo tienes un producto en el control de fuentes.

En general, organizar los proyectos por release o por propósito funciona bien a un nivel global. Comprende las ventajas y desventajas del modelo, y estudia cuál es el mejor para tu equipo.

Estructura Local de Directorios

NOTA

Con "Estructura Local de Directorios" nos referimos a la organización de los directorios de un proyecto concreto o una funcionalidad. Este tema aplica tanto al control de versiones de TFS como a Git, a menos que se indique lo contrario.

También debemos pensar la estructura local de los directorios por cada proyecto o funcionalidad en el árbol de código. Por ejemplo, los branches Dev y Main/Master tendrán varias funcionalidades o componentes bajo ellos y deberían tener una estructura de directorios normalizada. De esta manera conseguimos sentirnos cómodos a medida que vamos trabajando en los diferentes nodos del árbol. Una estructura de ejemplo podría ser la siguiente:

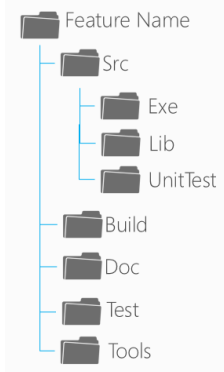


Figura 3 – Estructura de directorios de una funcionalidad

Directorio	Propósito
<Feature Name>	En TFVC, es el directorio raíz de algún componente de trabajo, que puede ser una funcionalidad o un componente del producto, una capa de arquitectura, o cualquier otra cosa que sea relevante para tu estructura. En Git, normalmente los branches representan funcionalidades, así que un directorio como este puede no existir.
Src	Todo el código fuente relacionado con la funcionalidad que se incluye en el producto o servicio.
Exe	[Opcional] Código que construye el ejecutable de una aplicación o componente. Fijaos que una arquitectura buena suele tener un ejecutable con librerías/DLLs de ayuda para construir el producto. Separar de concepto mejora la testabilidad, por lo que es aconsejable tener un directorio para estos casos.
Lib	[Opcional] Componentes individuales de la funcionalidad. Este directorio suele incluir, por ejemplo, assemblies o librerías en código nativo.
UnitTest	Todos los tests unitarios asociados a la funcionalidad. Los test unitarios deben acompañar siempre a todo el código y no deberíamos ocultarlo. No recomendamos poner los test unitarios en un árbol fuera del código fuente del producto, ya que el dueño es el equipo de desarrollo. Los test unitarios deberían ser de alta calidad y compilarse junto al código del producto. Los archivos de solución deberían incluir los proyectos de tests unitarios así como incluyen el código de la funcionalidad.
Build	[Opcional] Todos los script de construcción o despliegue específicos a la funcionalidad asociada.
Test	[Opcional] Cualquier test de integración del equipo de QA, test de carga, test de rendimiento, etc. necesarios para validar la funcionalidad. Aunque es aceptable tener estos test de QA en un árbol separado, es mejor incluirlos en los fuentes de la funcionalidad para tener un todo más cohesivo. Sin embargo, las librerías compartidas entre equipos sí deberían estar en un sitio distinto del árbol de código del contexto en el que estemos y no incluirlos en ninguna funcionalidad.
Tools	Cualquier herramienta relacionada con la funcionalidad, como un generador de código.

Tabla 1 – Subdirectorios sugeridos en una funcionalidad o componente.

Workspaces

Visual Studio Team Foundation Server 2012 y posteriores ofrecen dos opciones como workspace – un **server workspace** y un **local workspace**. Es una elección administrativa que hay que hacer para toda colección de Team Project. En esta sección revisaremos qué es un workspace y veremos algunas consideraciones a tener en cuenta para elegir un workspace se servidor o local.

NOTA

Leed también [Team Foundation Server – Trying to understand Server versus Local Workspaces](#)⁶ y la guía rápida de la terminología de la guía Branching and Merging Guide - Terminology Cheat Sheet para más información de referencia.

¿Qué es un Workspace?

De acuerdo con la [documentación](#)⁷, “Tu workspace es una copia local del código del equipo. Tu workspace te permite desarrollar y testear tu código de manera aislada en tu máquina de desarrollo hasta que estás listo para hacer un checkin de tu trabajo.”

Un Workspace mapea los directorios del servidor a directorios locales. Esto incluye:

- Una lista de todos los archivos de tu workspace
- La versión de cada archivo
- Una lista de los cambios pendientes
- Elementos activos y ocultos

Workspaces de servidor

Team Foundation Server 2010 y posteriores guardaban el mapeo de los directorios de servidor y locales en el servidor de Team Foundation. A esto lo llamamos **Server workspaces**. Hay un caché local para esta información, pero la administramos de manera central para permitir la coordinación de varias personas mientras trabajan tanto en modo conectado como desconectado.

Los workspaces de servidor nos permiten realizar operaciones de branching directamente en el servidor, con la mínima interacción del cliente. Como el servidor guarda esta información, otros usuarios con los permisos necesarios en el servidor pueden y que pueden ver los mismos directorios en el control de código pueden duplicar tu entorno de trabajo en sus máquinas. Sin embargo, no serán capaces de ver los detalles de cualquier cambio pendiente.

Los workspaces de servidor pueden ser privados (sólo los puede usar su propietario), públicos (cualquier usuario válido que tenga permisos de lectura/escritura) o limitado (cualquier usuario válido tiene permisos de lectura, pero sólo el propietario tiene permisos de escritura). Team Foundation Server también guarda esta información en el servidor.

NOTA

Con el modelo de workspace de servidor, no puedes iniciar ningún cambio hasta que no tengas una conexión con el servidor. Añadir, editar, renombrar o borrar archivos – todas estas operaciones necesitan una conexión con el servidor antes de hacerlas. Cuando tengas cambios pendientes, debes hacer un checkin al servidor para que puedas compartirlo con los demás.

Workspaces locales

Con workspaces locales, puedes hacer cambios en tu sistema de archivos local (no es necesaria una conexión con el servidor) con las herramientas que quieras, y luego puedes sincronizarlas con el servidor después. No importa

⁶ http://blogs.msdn.com/b/willy-peter_schaub/archive/2011/11/30/team-foundation-server-trying-to-understand-server-versus-local-workspaces.aspx

⁷ <http://msdn.microsoft.com/en-us/library/cc138514.aspx>

si estas offline mientras modificas el código – puedes trabajar en el autobús y sincronizarlo después. Team Foundation Server comprueba los cambios que has hecho y los gestiona adecuadamente.

En el lenguaje de gestión de configuración, llamamos a este estilo de control de versiones como **Modify-Merge-Commit** (también conocido como “**estilo Subversion**”). Funcionalidades:

- Los archivos no están protegidos contra escritura. Podemos editarlos sin tener que hacer un checkout explícito. Cuando modificamos archivos, TFS detecta los cambios automáticamente y cambia el estado del archivo automáticamente a “checked out”.
- Si creamos archivos nuevos, Team Foundation Server los detectará y nos permitirá añadirlos a nuestro proyecto.
- Si borramos archivos en local, Team Foundation Server nos dará la opción de borrarlos también del servidor o restaurarlo desde el servidor.
- Ya no necesitamos una herramienta para hacer la integración de versiones (ya que el master es el sistema local de archivos) así que podemos incluso usar el Bloc de notas o cualquier otra herramienta y Team Foundation Server seguirá funcionando como esperamos.
- Comparar archivos y otras operaciones siguen siendo posible ya que tenemos cacheado una copia de nuestro workspace

Workspaces de Servidor vs. Workspaces Locales

La siguiente tabla nos muestra las diferencias entre los workspaces de Servidor y Locales en Team Foundation Server.

Criterio	Workspace de Servidor	Workspace Local
Conexión de red	Requerido*	No requerido
Visibilidad de cambios pendientes a otros miembros del equipo	Visible	No visible
Cambios dependientes de la conexión	No	Sí
Alto rendimiento con proyectos grandes	Sí	Puede degradarse con grandes cantidades de archivos
Compatibilidad con Visual Studio 2010 y anteriores	Sí	No
Bloqueos de check-outs	Sí	No

Tabla 2 – Workspaces de Servidor vs. Workspace Locales

NOTA

*Es posible trabajar offline con un workspace de servidor, pero el Source Control Explorer no estará disponible. Si sueles trabajar sin conexión normalmente será mejor usar workspaces locales.

Decidir entre Workspaces Locales vs. Servidor

Recomendamos workspaces Locales en la mayoría de los escenarios. Elige un workspace de Servidor si:

- Necesitas escalar el control de código de Team Foundation Server a proyectos muy grandes, ya que los workspaces locales suelen tener problemas de rendimiento cuando guardan una gran cantidad de datos en local.
- Necesitas mantener las prácticas heredadas de VSS (single check out, get latest on checkout)

Configurando Workspaces

Podemos indicar si queremos un workspace Local o de Servidor. Sin embargo, esto sólo se puede hacer a nivel de Team Project Collection. Requiere coordinación entre los equipos que pertenezcan a esa colección para poder hacer la transición al tipo elegido.

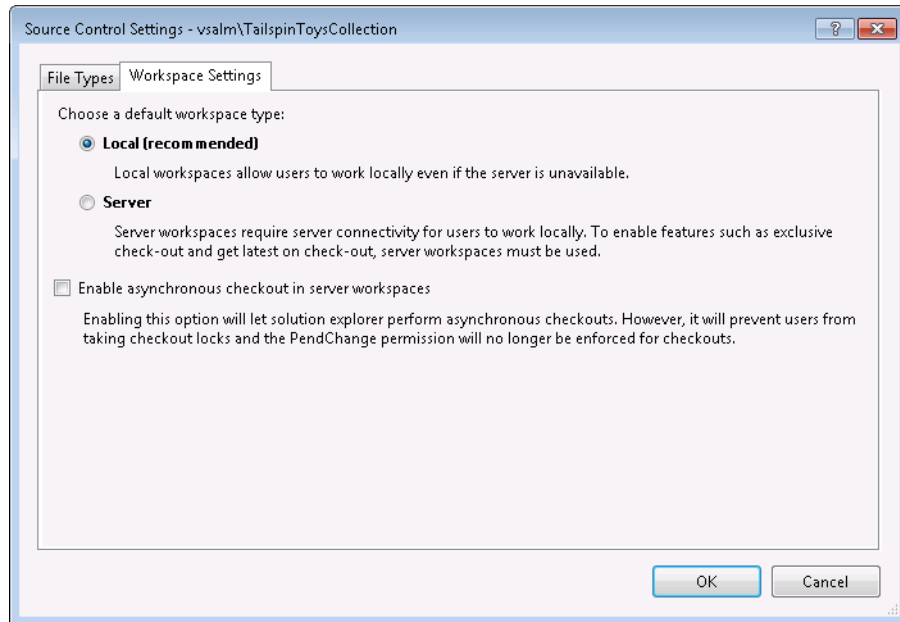


Figura 4 – Opciones de Workspace

NOTA

Los workspaces de Servidor siguen estando disponibles pero en las nuevas instalaciones de TFS el workspace por defecto es Local.

Cuando os actualicéis desde TFS 2010, la opción por defecto es mantener los workspaces de servidor por compatibilidad hacia atrás. Podéis cambiar ese comportamiento a través de las opciones de Team Project Collection Source Control Settings en el Team Explorer. Una vez que hagáis este cambio, los usuarios de una colección recibirán un aviso para que conviertan sus workspaces de servidor a workspaces locales cuando vayan a la sección de Pending Changes de un proyecto que tengan cargado y conectado al workspace de servidor. Esta operación sólo se hace una vez y puede que tarde algún tiempo en hacerlo en bases de código amplias.

Escenarios de uso

Escenario offline

Para workspaces de servidor, Team Foundation Server 2010, 2012 y 2013 tienen la misma funcionalidad.

Para workspaces locales:

- La velocidad de respuesta es muy rápida ya que el sistema de archivos es el master y no hay que contactar con Team Foundation Server para hacer checkout de nada. Muchas de las operaciones son asíncronas, como el empezar a modificar un archivo y comparar archivos.
- TFS cachea los archivos en los que estás trabajando en local, y soporta varias operaciones de archivo offline. Por ejemplo, podemos hacer una operación de comparación offline de manera que podemos ver qué hemos cambiado y deshacerlo – así podemos empezar de nuevo si queremos.
- Team Foundation Server 2012 introdujo el concepto de cambios locales de manera que podemos hacer un *tf status* (y ver el estado en Visual Studio) mientras estamos offline y ver todos los archivos que hemos modificado (aunque no le hayamos dicho nunca a Team Foundation Server que hemos modificado nada).

- También podemos hacer un *tf delete*, *tf add* o *tf rename* mientras no estamos conectados, y cuando nos reconectemos al servidor, todo se actualizará automáticamente y parecerá que siempre hemos estado conectados.

NOTA

Para hacer algunas operaciones es necesario que estemos conectados:

- No podemos hacer checkin mientras estamos offline. (No es como un sistema de control de versiones distribuido como Git)
- No podemos ver el historial, crear branches o hacer merges.

Si lo intentamos, veremos los mensajes de error que nos dirán que necesitamos estar conectados.

Merging

Seguimiento y gestión

El seguimiento y gestión de changesets y bugs es lo que se lleva la mayor parte del coste asociado a crear branches. Es fundamental que el equipo esté pendiente de cuándo crearlos para mantener esos costes bajos, e implementar algún proceso para medir y corregir esos costes.

Si estás siguiendo la estrategia de development and release isolation (conocida formalmente como el plan más básico) como el que vemos en la siguiente figura, mantener el control de los costes de merge suele ser fácil. Sólo tienes que asegurarte de que llevas los cambios a Main y a dev para integrarlos con las siguientes funcionalidades. De esta forma apenas tendrás que cambiar los procesos ni crear ningún tipo de automatización para ver que el coste se ha incrementado. Podemos usar las funcionalidades que ya trae el IDE Visual Studio para gestionar este escenario.

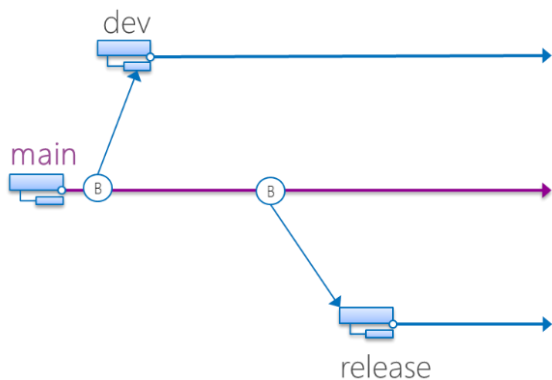


Figura 5 – Seguimiento del coste de un merge básico

En un escenario más avanzado de branching como el que vemos en la siguiente figura, REL A está en producción; mientras trabajamos en un service pack en SER A. REL B1 está desplegado en cliente para que lo pruebe. SER B está obteniendo feedback de REL B1 y también se están implementando nuevas funcionalidades que tendremos que incluir tanto en REL A y en el service pack antes de hacer un branch y publicar REL B2. Mientras todo esto está ocurriendo, tenemos que integrar (forward integrate) cualquier cambio a través de Main en el branch de funcionalidad FEA 1. FEA 1 se enviará al cliente como una versión sin soporte antes de integrar los cambios en el branch principal para que pase al proceso de producción. No se puede gestionar este nivel de complejidad sin algún nivel de automatización o procesos de cambios.

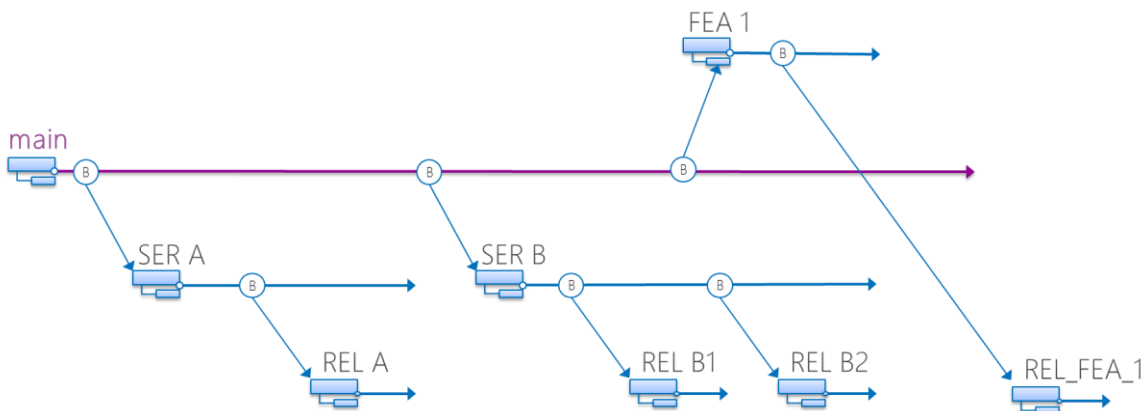


Figura 6 – Seguimiento del coste de un merge avanzado

Si tienes varios release vehicles activos, una solución para la gestión de defectos es crear un bug duplicado uno para cada branch donde hace falta aplicar la corrección. Sin embargo, es más fácil mantener un solo bug, con campos adicionales para indicar cuales son las releases que tienen que ser corregidas. Actualizar la plantilla de un Bug de TFS con campos adicionales. Por ejemplo, podríamos tener una entrada por cada branch activo: DEV, MAIN, RELEASE(S); y valores como Merged, Merge Required, y Merge Not Required. Así seremos capaces de informar más fácilmente el estado de los bugs en todos los release vehicles activos.

En escenarios de branching complejos, recomendamos automatizar el diagnóstico y la notificación de los costes de los merge. Podemos hacer esto de varias formas. Lo más fácil es ejecutar tf.exe en todos los path activos para notificar el estado de las cosas. El ejemplo siguiente ofrece una solución basada en MSBuild para conseguir esto. Ejecuta un preview de un merge sobre una colección de paths:

```
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <!-- Create a collection of active merge paths -->
  <ItemGroup>
    <MergePath Include="DevToReleaseA">
      <SourceBranch>$/ProjectName/Development</SourceBranch>
      <TargetBranch>$/ProjectName/Releases/RelA</TargetBranch>
    </MergePath>
    <MergePath Include="ReleaseBtoDev">
      <SourceBranch>$/ProjectName/Releases/RelB</SourceBranch>
      <TargetBranch>$/ProjectName/Development</TargetBranch>
    </MergePath>
  </ItemGroup>
  <!-- Set which workspace to operate in and whether to ignore conflicts or not -->
  <PropertyGroup>
    <WorkspacePath>C:\YourWorkspacePath</WorkspacePath>
    <IgnoreConflicts>true</IgnoreConflicts>
  </PropertyGroup>
  <Target Name="Report" Inputs="@ (MergePath)" Outputs="% (Identity) ">
    <Message Text="% (MergePath.SourceBranch)" to "% (MergePath.TargetBranch)"/>
    <Exec Command='tf merge "% (MergePath.SourceBranch)" "% (MergePath.TargetBranch)" /preview
/recursive' WorkingDirectory='$(WorkspacePath)' IgnoreExitCode="$(IgnoreConflicts)"/>
  </Target>
</Project>
```

Otra forma es que Team Foundation Server ofrece una API rica para notificar (VersionControlServer.GetMergeCandidates) y ejecutar (Workspace.Merge) el coste de un merge. Podrías escribir una UI simple para que te ayude a monitorizar y gestionar los costes. Permitir a un miembro del equipo determinar cuántos cambios están pendientes y cuántos conflictos hay; hacer un merge automático, crear una tarea de merge y asignarla a un miembro del equipo si se encuentran conflictos; y quizás permitir desechar changesets que no se quieren incluir en el merge. Si mantienes los path de merges ‘limpios’ estarás más seguro de que los bugs y las nuevas funcionalidades se integran en sus branches.

Merges huérfanos

Normalmente, se suelen realizar merges de manera bidireccional entre branches que comparten una relación padre-hijo. En ocasiones, puede ser necesario hacer un merge entre branches que no tienen esa relación. Llamamos a estos merges “merges huérfanos”. Un merge huérfano crea una relación entre los branch origen y destino que no forman parte de una relación padre-hijo.

Hacer un merge huérfano debería ser una excepción y no algo normal. Si llega un momento en el que os encontráis haciendo este tipo de merges asiduamente, será mejor que reviséis la estrategia de branching que hayáis elegido. Sin embargo, hay razones legítimas para realizar un merge huérfano, así que no los descartes del todo. Por ejemplo, podrías necesitar hacer una corrección urgente en producción justo mientras estás en el ciclo de testing del branch principal: así que tienes que mover la corrección sin pasar por el branch principal.

Desventajas de los merges huérfanos

Como no hay una versión base, hacer un merge de 3 direcciones no es posible con un merge huérfano. Un merge de 3 direcciones compara los archivos de origen y destino con los de la versión base permitiendo un

merge más parecido a un auto-merge. Un auto-merge no es posible con merges huérfanos, por lo que tendremos que resolver los conflictos manualmente.

Antes de Visual Studio 2010 SP1, TFS no permitía hacer merge con archivos que estaban en estado de borrado desde el branch origen al branch de destino.

Escenario de merge

En el siguiente escenario, hay tres branches de desarrollo a partir de la principal. Con esta estructura, si necesitamos incluir un cambio desde DEV FT2 a DEV FT1, el camino normal sería hacer un merge (RI) desde DEV FT2 a MAIN, seguido de un merge (FI) desde MAIN a DEV FT1. Sin embargo, si no es posible incluir el código en MAIN, la única forma de moverlo desde DEV FT2 a DEV FT1 es a través de un merge huérfano.

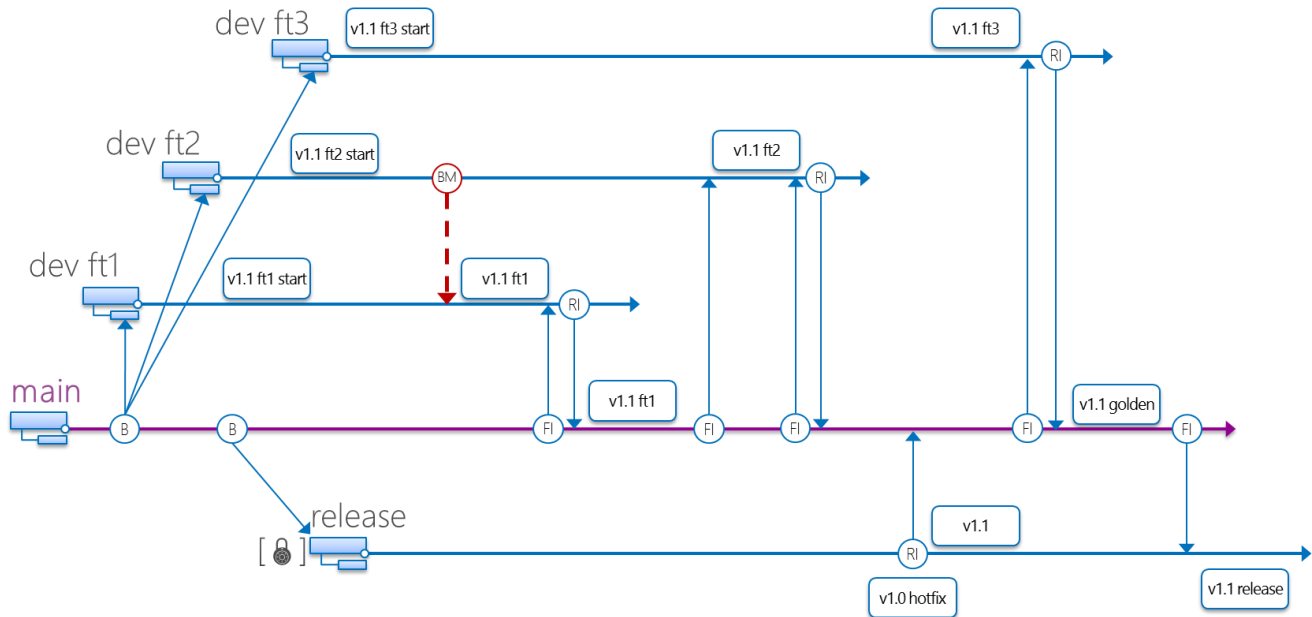


Figura 7 – Merge huérfano entre DEV FT1 y DEV FT2

Como no hay una relación directa entre los branches DEV FT2 y DEV FT1, hay que hacer primero un merge huérfano y resolver manualmente cualquier conflicto que pueda haber. Una vez que completemos el merge huérfano, Team Foundation server guardará un histórico de merges y establecerá una relación entre branches. En futuros merges, tendremos un merge de 3 direcciones, consiguiendo así reducir el número de conflictos.

Merges huérfanos en Visual Studio

En Visual Studio 2013, podemos hacer merges huérfanos en el Source Control Explorer. En la ventana de merge, usad el botón Browse para seleccionar el branch de destino. Si el origen y el destino no tienen una relación entre ellos, aparecerá un icono de aviso y un mensaje indicando que se va a realizar un merge huérfano.

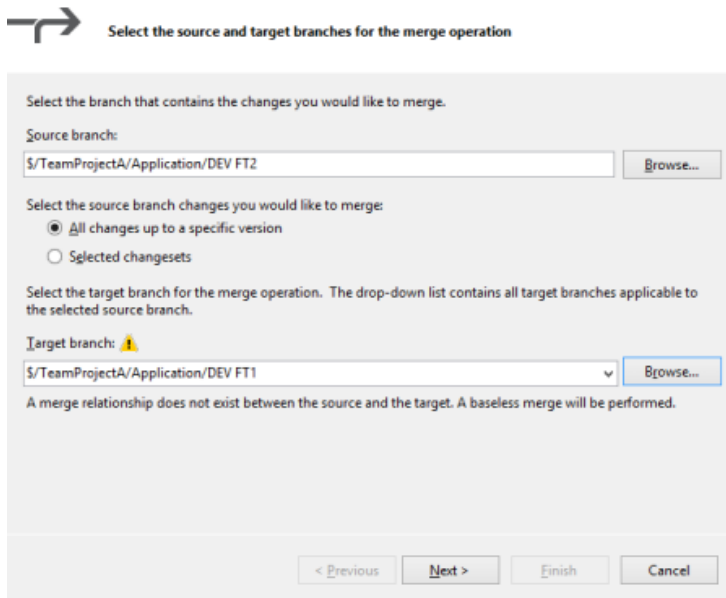


Figura 8 – Aviso de un merge huérfano

Desde Team Foundation Server 2010, no se pueden hacer merges huérfanos usando el Source Control Explorer. En la línea de comandos de Visual Studio escribimos:

```
tf merge /baseless <<source path>> <<target path>> /recursive
```

Podemos indicar qué changesets aplicar. Por ejemplo, para hacer un merge huérfano del changeset 150 desde DEV FT2 a DEV FT1:

```
tf merge /baseless /v:C150~C150 "$/TeamProjectA/DEV FT2" "$/TeamProjectA/DEV FT1" /recursive
```

Para hacer un merge huérfano de todos los changesets hasta el 150 desde DEV FT2 a DEV FT1:

```
tf merge /baseless /v:C150 "$/TeamProjectA/DEV FT2" "$/TeamProjectA/DEV FT1" /recursive
```

Podemos realizar nuevos merges entre estos branches usando el Source Control ya que existe una relación entre ellos.

Si realizamos un merge huérfano con "All changes up to a specific version," cuando hagamos checkin, todos los archivos tendrán el estado siguiente. Ya que un merge huérfano crea una relación de cada archivo del directorio padre (DEV FT1). Si seleccionamos "Selected changesets", la ventana de Pending Changes sólo mostrará los archivos incluidos en el changeset. TFS no establecerá ninguna relación entre DEV FT2 y DEV FT1, ya que TFS sólo establece relaciones entre archivos que están incluidos en el changeset.

La siguiente figura muestra la relación entre DEV FT2 y DEV FT1 después de realizar un merge huérfano con la opción "All changes up to a specific version".

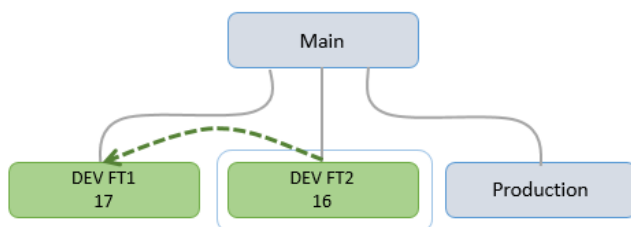


Figura 9 – Seguimiento de un Merge Huérfano

Hands-on Lab (HOL) – Workspaces & Merging

Con **Visual Studio 2013**, crearemos un workspace local y estudiaremos esta operación tanto en los modos online como en offline. Después investigaremos la funcionalidad de obtener las diferencias de archivos, veremos las herramientas de merge y luego veremos la nueva funcionalidad Code Lens.

IMPORTANTE

Hemos basado este Hands-on Lab (HOL) en la nueva plantilla de HOL, por lo que asumimos que tienes eres un usuario experimentado en Visual Studio ALM.

- Las instrucciones de cómo empezar con Visual Studio y crear un Team Project no están incluidas. Te recomendamos visitar <http://msdn.microsoft.com> y otra documentación del producto para obtener esa información.
- Hemos reducido y centrado las capturas de pantalla para reducir el coste de mantenimiento a medida que el producto evoluciona.

Prerrequisitos

Para completar este Hands-On Lab necesitaremos este entorno:

- [Visual Studio 2013 Application Lifecycle Management Virtual Machine from Brian Keller](#) ⁸

O

- Un servidor (físico o virtual) con el siguiente software instalado y configurado:

Software	Versión
Sistema Operativo	Windows 8, Windows Server 2012 (o posterior)
Visual Studio	2013
Visual Studio Team Foundation Server	2013

Tabla 3 – Prerrequisitos del Hands-On Lab

Por favor, ve al Apéndice de la página 29 para más información sobre cómo configurar el entorno para este Hands-On Lab.

Índice de tiempos del lab

Si quieres seguir este HOL paso a paso, en esta tabla tienes los tiempos mínimos estimados. Sin embargo, si quieres investigar cada paso en detalle el tiempo puede ser del doble como mínimo...

Tema	Duración en minutos	Página
Ejercicio 1– Crea un nuevo workspace local	5	18
Ejercicio 2 – Usa el workspace local en modo online	15	19
Ejercicio 3 – Usa el workspace local en modo offline	15	23
Ejercicio 4 – Explora las herramientas de comparación y merge	15	23
	TOTAL 50 min	

⁸ <http://aka.ms/almvms>

Tabla 4 – Índice de tiempos del Lab

Ejercicio 1: Crea un workspace local

META

En este ejercicio, nos conectaremos a Team Foundation server y crearemos un workspace local

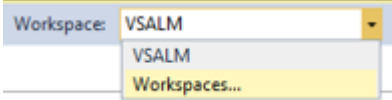
Tarea 1: Lógate en tu entorno

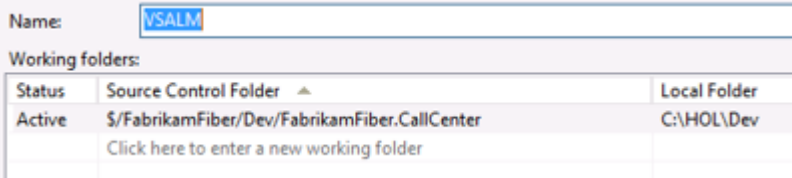
Paso	Instrucciones
Logon ☐ - Hecho	<ul style="list-style-type: none"> Si te estás logando en una instancia de la VM de BK, por ejemplo: en TechReady, usa las credenciales Administrator P2ssw0rd. Si no, usa las credenciales de tu entorno.

Tarea 2: Conexión a Team Foundation Server

Paso	Instrucciones
1 Abre Select Team Project ☐ - Hecho	<ul style="list-style-type: none"> En el Team Explorer haz clic en Select Team Projects
2 Selecciona Team Foundation Server ☐ - Hecho	<ul style="list-style-type: none"> Selecciona el localhost como servidor, si el servidor no existe añade un nuevo servidor en la url http://localhost:8080/tfs
3 Conéctate a un Team Project ☐ - Hecho	<ul style="list-style-type: none"> Elige FabrikamFiberCollection Marca FabrikamFiber Haz clic en Connect

Tarea 3: Crea el workspace local

Paso	Instrucciones
1 Abre Workspaces ☐ - Hecho	<ul style="list-style-type: none"> En el Team Explorer, haz clic en el enlace del Source Control Explorer. Encima del Source Explorer, selecciona el desplegable Workspace y selecciona Workspaces. 
2 Selecciona Working Folder ☐ - Hecho	<ul style="list-style-type: none"> Haz clic en Edit Bajo Working folders haz clic en <i>Click here to enter a new working folder</i> Haz clic en (...) Expande localhost\FabrikamFiberCollection Expande FabrikamFiber Expande Dev Selecciona FabrikamFiber.CallCenter

Paso	Instrucciones
	<ul style="list-style-type: none"> Haz clic en OK
3 Selecciona Local Folder ☐ - Hecho	<ul style="list-style-type: none"> Haz clic en el espacio que hay justo debajo de Local Folder Haz clic en (...) Navega y selecciona el directorio c:\hol\Dev (crea este directorio si no existe) Haz clic en OK 
4 Mapea y Get latest ☐ - Hecho	<ul style="list-style-type: none"> Haz clic en OK Haz clic en Yes cuando te indique que el workspace ha sido modificado. Por último, haz clic en Close.

REVISIÓN

Nos hemos conectado a Team Foundation Server, abierto el team Project FabrikamFiber, y creado un workspace local.

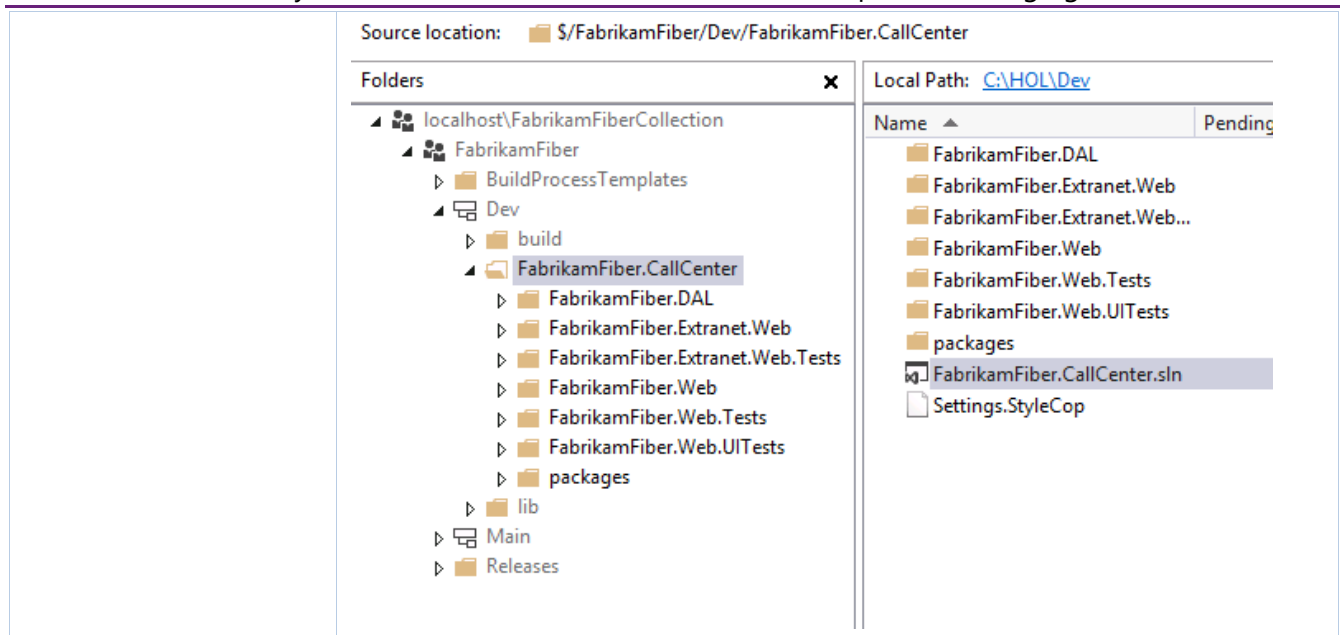
Ejercicio 2: Usa el workspace local en modo online

META

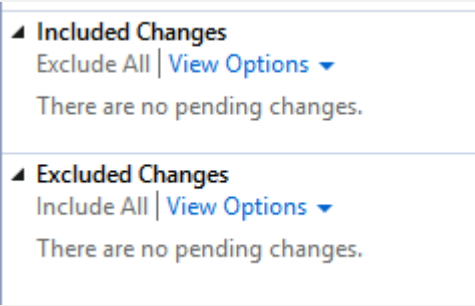
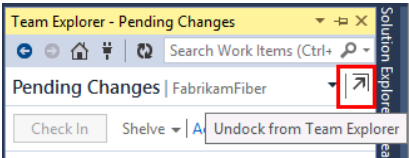
En este ejercicio, veremos la funcionalidad de los workspaces locales cuando trabajamos en modo online.

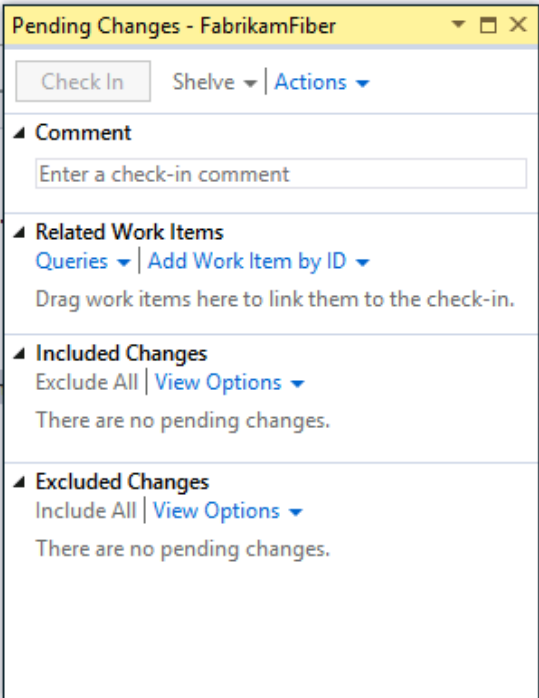
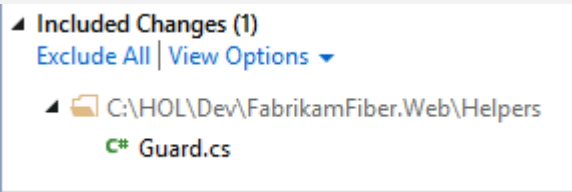
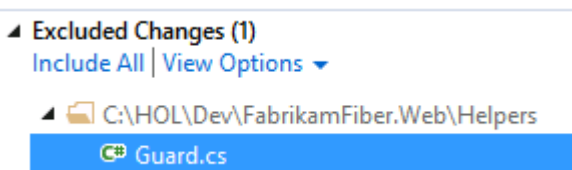
Tarea 1: Abre la solución FabrikamFiber Call Center

Paso	Instrucciones
Abre FabrikamFiber.CallCenter.sln ☐ - Hecho	<ul style="list-style-type: none"> Abre el Source Control Explorer y ve al directorio FabrikamFiber\Dev\FabrikamFiber.CallCenter Haz doble clic en FabrikamFiber.CallCenter.sln.



Tarea 2: Haz algún cambio usando Visual Studio

Paso	Instrucciones
1 Abre un archivo de código C# ☐ - Hecho	<ul style="list-style-type: none"> Abre el Solution Explorer Expande el proyecto FabrikamFiber.Web Expande el directorio Helpers Haz doble clic en Guard.cs para ver el código C#
2 Abre el Team Explorer ☐ - Hecho	<ul style="list-style-type: none"> Abre el Team Explorer Haz clic en el enlace Pending Changes. Fíjate en las secciones de Included Changes y Excluded Changes 
3 Puedes desanclar Pending Changes ☐ - Hecho	<p>Fíjate que en Visual Studio 2013 podemos desanclar la ventana de Pending Changes del Team Explorer, podemos hacer esto haciendo clic en la flecha de la esquina superior derecha de la sección de pending changes.</p>  <p>Después de esto la ventana luce así</p>

Paso	Instrucciones
	
<p>4</p> <p>Modifica el código</p> <p>☐ - Hecho</p>	<ul style="list-style-type: none"> • Añade un comentario (o modifícalo de otra forma) a la clase Guard. • Guarda el archivo • Fíjate que TFS lista el archivo en la sección Included Changes. 
<p>5</p> <p>Excluye Guard.cs de la lista de pending changes</p> <p>☐ - Hecho</p>	<p>Arrastra la entrada de Guard.cs desde la sección Included Changes y suéltala en <i>Drag changes here to exclude from the check-in</i> debajo de Excluded Changes. Fíjate que el archivo está ahora en la sección de Excluded Changes</p> 
<p>6</p> <p>Deshaz todos los cambios</p> <p>☐ - Hecho</p>	<ul style="list-style-type: none"> • En Pending Changes haz clic en Actions, Undo all • En el diálogo Undo Pending Changes haz clic en Undo Changes • En el message box Confirm Undo Checkout, haz clic en Yes to all. Fíjate que el cambio se deshace y el archivo ha sido eliminado de la sección Excluded Changes

Tarea 3: Haz cambios desde fuera de Visual Studio

Paso	Instrucciones
1 Abre el directorio FabrikamFiber.Web en el Explorador de Windows ☐ - Hecho	<ul style="list-style-type: none"> Abre el Source Control Explorer Navega hasta el directorio FabrikamFiber.Web Haz clic en el enlace junto a Local Path para abrir el directorio en el que está mapeado FabrikamFiber.Web en el explorador de Windows, que debería ser c:\hol\Dev\FabrikamFiber.Web. <div>Local Path: C:\HOL\Dev\FabrikamFiber.Web</div>
2 Haz cambios a los archivos del proyecto sin usar Visual Studio ☐ - Hecho	<ul style="list-style-type: none"> Haz doble clic en el archivo readme.txt para abrirlo con el Bloc de notas haz un cambio y guárdalo. Desde el explorador de Windows borra el archivo Web.config. Desde el explorador de Windows añade un nuevo archivo llamado Notes.txt. Haz doble clic en el directorio Helpers, haz clic derecho en Guard.cs, elige la opción Abrir con, Bloc de notas, haz un cambio y guárdalo.
3 Recarga los cambios en Visual Studio ☐ - Hecho	<ul style="list-style-type: none"> Vuelve a Visual Studio Haz clic en yes cuando te pregunte si quieres recargar la versión modificada de Guard.cs
4 Revisa los cambios detectados. ☐ - Hecho	<ul style="list-style-type: none"> Fíjate en los archivos del proyecto que han sido modificados ya están en la lista de Included Changes, aunque se hayan hecho desde fuera de Visual Studio Fíjate también que en la sección Excluded Changes incluye el enlace Deteced: 1 add(s), 1 delete(s).
5 Promociona los Excluded Changes ☐ - Hecho	<ul style="list-style-type: none"> Haz clic en el enlace Detected: 1 add(s), 1 delete(s). Fíjate que la lista Promote Candidate Changes contiene otros archivos que fueron modificados Haz clic en Promote. Todos los cambios que hemos hecho están ahora en la sección Include Changes
6 Deshaz el borrado de archivo ☐ - Hecho	<ul style="list-style-type: none"> En Included Changes haz clic derecho en la entrada de la eliminación del Web.config (y Web.Debug.config y Web.Release.config si están en la lista) Haz clic en Undo En el diálogo Undo Pending Changes, haz clic en Undo Changes. El archivo Web.config ha sido restaurado y se ha eliminado de la lista Included Changes
7 Límpialo todo ☐ - Hecho	<ul style="list-style-type: none"> En Pending Changes haz clic en Actions, Undo All En el diálogo Undo Pending Changes haz clic en Undo Changes En Confirm Undo Checkout, haz clic en Yes to All. Todos los cambios se desharán y se eliminarán de la lista de Included Changes Termina de limpiarlo todo desde el explorador de Windows para borrar el archivo Notes.txt que habíamos creado

REVISIÓN

Hemos abierto la solución FabrikamFiber CallCenter y hemos hecho cambio tanto desde dentro como desde fuera de Visual Studio.

Ejercicio 3: Usa el workspace local en modo offline

META

En este ejercicio, veremos la funcionalidad de los workspaces locales cuando trabajamos en modo offline.

Con los workspaces en modo offline, podemos editar un archivo aun cuando no tengamos conexión a la red

Tarea 1: Desconéctate de Team Foundation Server

Paso	Instrucciones
Para el IIS para desconectarte del TFS ☐ - Hecho	<ul style="list-style-type: none"> Abre una consola de comandos como administrador Ejecuta el comando iisreset /stop desde esa consola. Se debería mostrar el mensaje <i>Internet services successfully stopped</i>

Tarea 2: Haz cambios tanto desde dentro como desde fuera de Visual Studio

Paso	Instrucciones
1 Haz cambios con Visual Studio ☐ - Hecho	<p>Repite la tarea 2 del ejercicio 2 para hacer cambios al archivo Guard.cs del proyecto FabrikamFiber.Web usando Visual Studio</p> <p>NOTA Si recibes el mensaje TF400324: Team Foundation services are not available from server localhost\DefaultCollection ciérralo; el mensaje sólo está notificando que no se puede contactar con Team Foundation Server y Visual Studio está trabajando en modo offline.</p>
2 Haz cambios desde fuera de Visual Studio ☐ - Hecho	Repite la tarea 3 del ejercicio 2 para hacer varios cambios desde fuera de Visual Studio

NOTA

La primera tarea de este paso es abrir el directorio de la solución con el explorador de Windows usando el enlace Local Path del Source Control Explorer. Este enlace no está disponible ya que nos hemos desconectado del Source Control Explorer de Team Foundation Server por lo que tenemos que navegar manualmente al directorio.

REVISIÓN

Nos hemos desconectado de Team Foundation Server y hemos hecho cambios tanto desde dentro como desde fuera de Visual Studio.

Ejercicio 4: Investiga las herramientas de comparación y merge

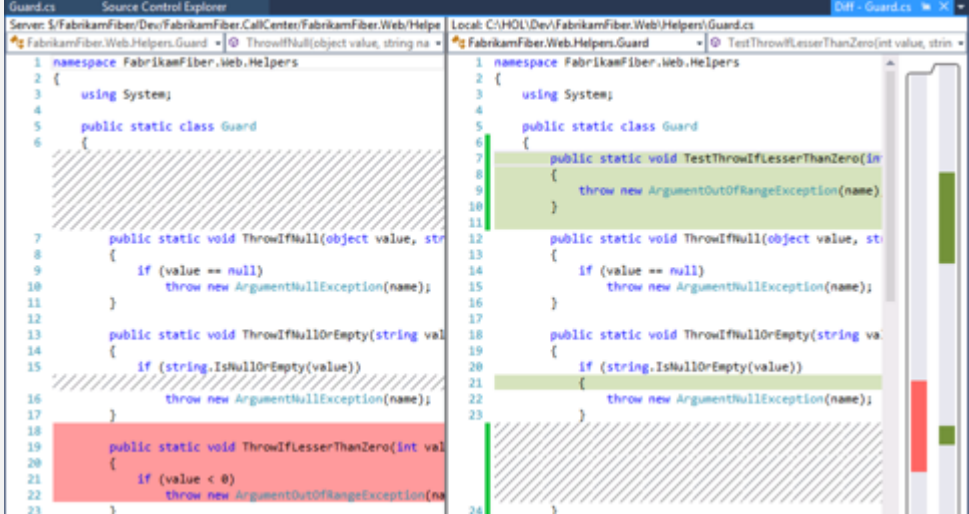
META

En este ejercicio, veremos las herramientas de comparación y merge que funcionan tanto en los modos online como offline.

NOTA

Este ejercicio asume que has completado los ejercicios anteriores y que estás trabajando en modo desconectado de Team Foundation Server. Si no es el caso, desconéctate completando la tarea 1 del ejercicio 3.

Tarea 1: haz una comparación en modo offline

Paso	Instrucciones
1 Modifica el código ☐ - Hecho	Vuelve a Visual Studio, haz varias modificaciones a la clase Guard y guarda los cambios
2 Compara con la versión del Workspace ☐ - Hecho	<ul style="list-style-type: none"> Abre Pending Changes Haz clic derecho en Guard.cs bajo Included Changes Haz clic en Compare with Workspace Version La ventana de comparación debería aparecer en una pestaña provisional.
3 Cambia el modo de comparación ☐ - Hecho	<ul style="list-style-type: none"> Localiza la barra de herramientas de Compare Files, haz clic en el primer icono de la lista Haz clic en el modo side-by-side
4 Explora la barra de herramientas Compare Files ☐ - Hecho	<p>Tómate un momento para explorar las opciones de la barra de herramientas de Compare Files y observa la funcionalidad en la ventana de diferencias. En concreto, fíjate en lo siguiente en la ventana de comparación:</p> <ul style="list-style-type: none"> Usa el editor de Visual Studio por lo que soporta las funcionalidades de IntelliSense, highlighting, deshacer, rehacer, y navegación. Visual Studio destaca los cambios en cada línea. Ay un mapa de cambios a la derecha. 

Tarea 2: Reconéctate a Team Foundation Server

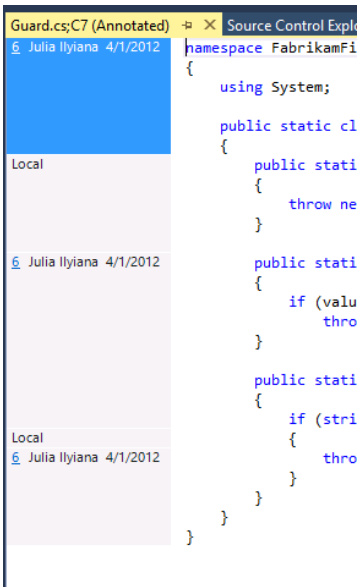
Paso	Instrucciones
1 Ejecuta IIS ☐ - Hecho	Vuelve a la consola de comandos que abrimos como administrador y ejecuta el comando iisreset /start . Debería mostrarse el mensaje <i>Internet services successfully started</i>

Joyas de TFVC – Hands-on Lab (HOL) – Workspaces & Merging

Paso	Instrucciones
2 Reconéctate a Team Foundation Server ☐ - Hecho	<ul style="list-style-type: none"> Abre el Source Control Explorer Haz clic en el icono Refresh para reconectarte al Team Foundation Server El Source Control Explorer debería reconectarse y actualizarse

Tarea 3: Anota los cambios del código

Puedes anotar en un archivo quién y qué cambios ha hecho en las versiones más tempranas del archivo

Paso	Instrucciones
1 Abre las anotaciones de Guard.cs ☐ - Hecho	<ul style="list-style-type: none"> Abre Pending Changes Haz clic derecho Guard.cs bajo Included Changes Haz clic en Annotate  <ul style="list-style-type: none"> Revisa las anotaciones a la izquierda del código
2 Log Off ☐ - Hecho	<ul style="list-style-type: none"> Guarda todos los cambios, cierra Visual Studio y deslogueate

Tarea 4: Haz cambios como otro usuario

Paso	Instrucciones
1 Log on ☐ - Hecho	Repite las tareas 1 y 2 del ejercicio 1 – Lógate como Adam Barr usando la misma contraseña
2 Mapea el directorio ☐ - Hecho	Repite la tarea 3 del ejercicio 1 – Usando el directorio c:\hol\Adam Barr como Local Folder

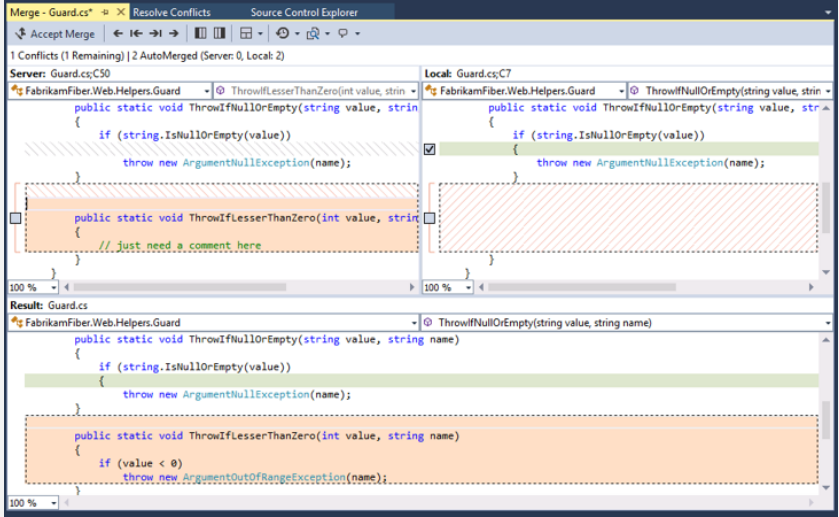
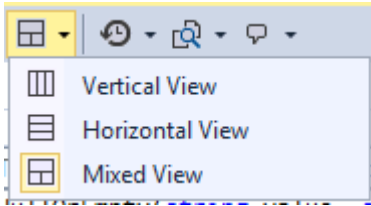
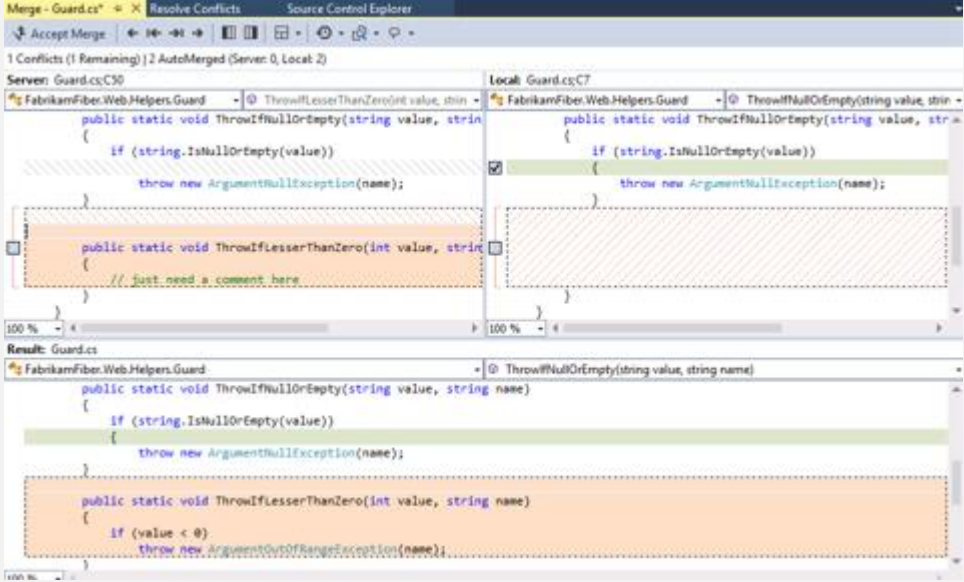
Joyas de TFVC – Hands-on Lab (HOL) – Workspaces & Merging

Paso	Instrucciones
3 Abre la solución FabrikamFiber CallCenter ☐ - Hecho	Repite la tarea 1 del ejercicio 2 para abrir la solución FabrikamFiber CallCenter
4 Abre un archivo de código ☐ - Hecho	Abre el Solution Explorer si no está abierto aún y expande el proyecto FabrikamFiber.Web, expande el directorio Helpers y haz doble clic en Guard.cs
5 Modifica el código ☐ - Hecho	Haz un pequeño cambio en el código. Debes modificar alguna de las líneas que modificaste cuando estabas logado como el primer usuario. Sin embargo, este cambio debería ser diferente. Ya que vas a hacer checkin de esta modificación, el cambio no debería afectar a la operación del código. Guarda el cambio.
6 Haz checkin ☐ - Hecho	<ul style="list-style-type: none"> Abre la ventana de Pending Changes si no está abierta y en la sección Comment haz clic en <i>Enter a check-in comment</i>. Escribe el comentario que quieras Haz clic en Check in En el mensaje de confirmación del checkin haz clic en Yes Verás un mensaje de confirmación en la parte de arriba de la ventana de Pending Changes indicando que se han confirmado los cambios.
7 Deslogate ☐ - Hecho	Deslogate y vuelve a logarte como Administrator

Tarea 5: Resuelve los conflictos

Paso	Instrucciones
1 Obtén la última versión ☐ - Hecho	<ul style="list-style-type: none"> Abre Visual Studio Abre Source Control Explorer Haz clic derecho en FabrikamFiber.CallCenter Haz clic en Get Latest Version

Joyas de TFVC – Hands-on Lab (HOL) – Workspaces & Merging

Paso	Instrucciones
2 Resuelve los conflictos <input type="checkbox"/> - Hecho	<ul style="list-style-type: none"> Debería aparecer la ventana Resolve Conflicts Haz clic en Merge Changes in Merge Tool Aparecerá la herramienta de Merge mostrando la versión del servidor, la versión local y el resultado 
3 Cambia el modo de Vista <input type="checkbox"/> - Hecho	<p>Puedes cambiar la vista haciendo clic en el icono View</p> 
4 Explora la barra de herramientas de Merge <input type="checkbox"/> - Hecho	<p>Tómate un momento para explorar las opciones de la barra de herramientas de Merge y explora sus funcionalidades. En concreto, fíjate en los checkbox de la izquierda en los paneles Server y Local para cada diferencia y conflicto. Estos checkbox ofrecen un mecanismo fácil para indicar qué cambios deben incluirse y cuales se deben excluir</p> 

Joyas de TFVC – Hands-on Lab (HOL) – Workspaces & Merging

Paso	Instrucciones
5 Deshaz todos los cambios □ - Hecho	<ul style="list-style-type: none">• Abre Pending Changes• Haz clic en Actions, Undo All• En el diálogo Undo Pending Changes haz clic en Undo Changes• En Confirm Undo Checkout, haz clic en Yes to All. <p>Todos los cambios se desharán y se eliminarán de la ventana de Included Changes</p>

REVISIÓN

Hemos realizado una comparación en modo offline, nos hemos vuelto a conectar al Team Foundation Server, anotado los cambios del código, hemos hecho modificaciones como un usuario diferente, y hemos resuelto los conflictos.

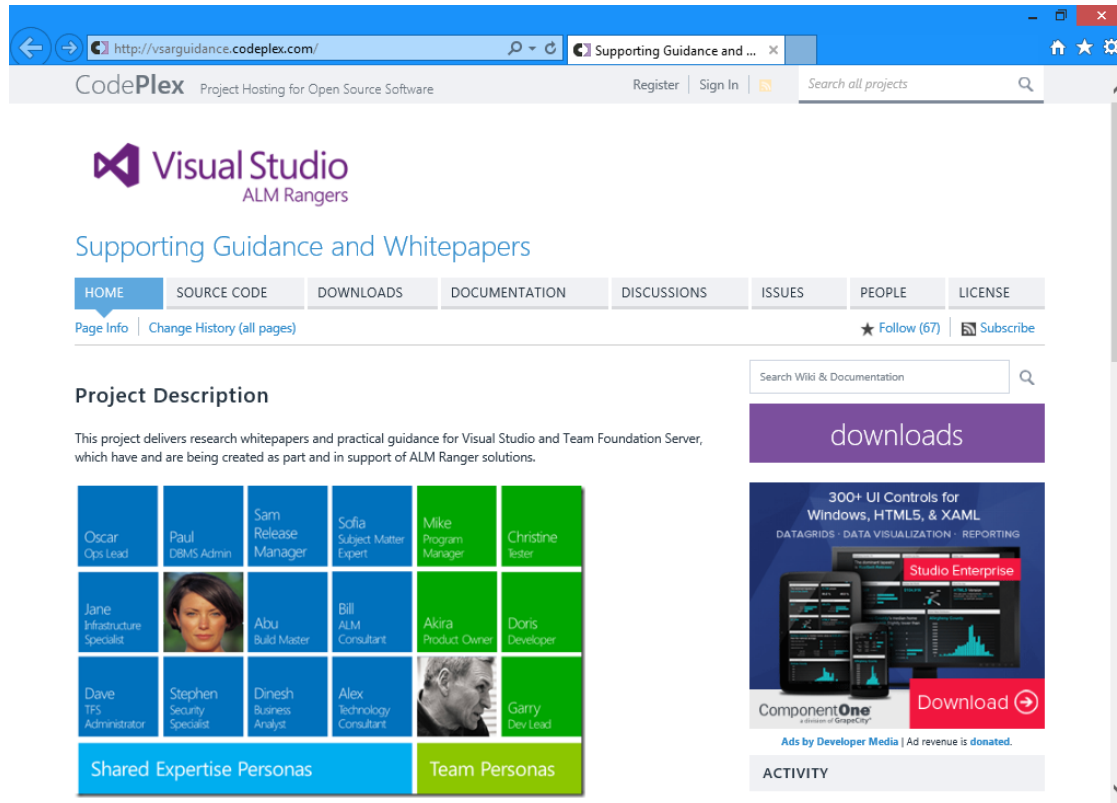
Apéndice: Configuración del entorno

NOTA

Este apéndice contiene información sobre cómo configuramos los entornos que hemos usado para este Hands-On Lab **si y sólo si** el entorno no incluye una instancia de la solución FabrikamFiber como se usa en las máquinas virtuales de Brian Keller (<http://aka.ms/almvms>).

1. Instala y configura la solución FabrikamFiber.

Este HOL usa un team Project llamado FabrikamFiber que contiene la solución FabrikamFiber.CallCenter con un directorio de desarrollo. Puedes descargar la solución de ejemplo del sitio de CodePlex de los Visual Studio ALM Rangers en <http://vsarguidance.codeplex.com>. Descarga el ejemplo y sigue los pasos de instalación.



Si estás usando la máquina virtual de Brian Keller, ya tienes todos los archivos que necesitas pero tendrás que [eliminar los mapeos de los workspaces](#) ya que ya están configurados la primera vez que te logas.

NOTA

Este Hands-On Lab no usa ninguna de las funcionalidades de la solución **FabrikamFiber**. Si no tienes disponible el ejemplo **FabrikamFiber**, puedes sustituirlo por cualquier otro código y hacer los cambios necesarios mientras sigues los pasos del HOL.

FAQ

¿Cuáles son las desventajas de no seleccionar todos los cambios?

Puede ocurrir que algún desarrollador haya hecho un cambio en otros archivos de los que tengas conocimiento. Como no hiciste un merge completo, has dejado esos archivos atrás. Ahora tu código objetivo está "roto", puede que no compile correctamente y puede que te pases horas intentando descubrir qué fue mal.

Un changeset hecho de esta forma puede caer en medio de un merge y en un futuro provocará conflictos.

En proyectos más complejos, este tipo de changeset suelen causar problemas como bugs que habías corregido que vuelven a aparecer en las próximas builds, o romper las builds debido a versiones incorrectas de los archivos.

¿Qué es un merge huérfano y en qué se diferencia de un merge normal?

Un merge huérfano permite hacer merge de dos directorios que no tienen ninguna relación. Una vez que se hace un merge huérfano, puedes hacer merges normales entre los branches (como si los dos branches tuviesen una relación normal entre ellos).

¿Cuál es la longitud máxima del path de un archivo que soporta TFS?

TFS actualmente soporta hasta 260 caracteres en cualquier cliente y 400 en el servidor. Aquí tenéis unos paths de ejemplo:

- Cliente: c:\code\src\...
- Servidor: \$/TP/SomeFolder/Code/Src...

Conclusión

Aquí termina nuestra aventura por las Joyas de TFVC. Hemos visto la teoría básica de directorios, workspaces y merging. Hemos concluido con un Hands-on Lab (HOL) viendo los workspaces y los merges.

Esperamos que esta guía te haya resultado útil.

Atentamente

The Microsoft Visual Studio ALM Rangers

