Hands-On Lab

# Making Developers More Productive with Visual Studio Team Foundation Server 2015

Lab version: 14.0.25123.0

Last updated: 5/2/2016



1. **TABLE OF CONTENT**

[Making Developers More Productive with Visual Studio Team Foundation Server 2015 1](#_Toc430861098)

[Overview 3](#_Toc430861099)

[Prerequisites 3](#_Toc430861100)

[About the Fabrikam Fiber Scenario 3](#_Toc430861101)

[Exercises 3](#_Toc430861102)

[Exercise 1: Team Explorer Overview 4](#_Toc430861103)

[Exercise 2: My Work 10](#_Toc430861104)

[Exercise 3: Local Workspaces 13](#_Toc430861105)

[Exercise 4: Pending Changes 21](#_Toc430861106)

[Exercise 5: Suspend and Resume 25](#_Toc430861107)

[Exercise 6: Code Reviews 32](#_Toc430861108)

[Exercise 7: Merging 43](#_Toc430861109)

## Overview

* 1. This lab is all about those “fit-and-finish” scenarios in Visual Studio Team Foundation Server that will make developers more productive as they write code and collaborate with other team members. You’ll learn about integrated code reviews; the “My Work” experience for managing your active tasks; and once you’re “in the zone”, Visual Studio will now help you stay focused on the task at hand, no matter how much you’re randomized, with a suspend/resume experience. You’ll also learn how to work offline seamlessly, how the new merging experience works, and how you can more easily find work items.

### Prerequisites

* 1. In order to complete this lab you will need the Visual Studio 2015 virtual machine provided by Microsoft. For more information on acquiring and using this virtual machine, please see [this blog post](http://aka.ms/almvm).

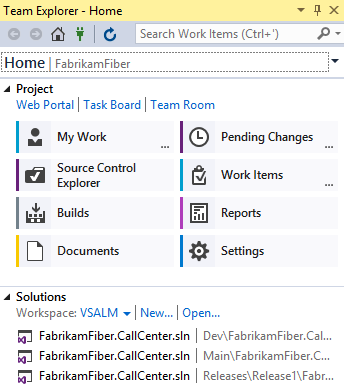
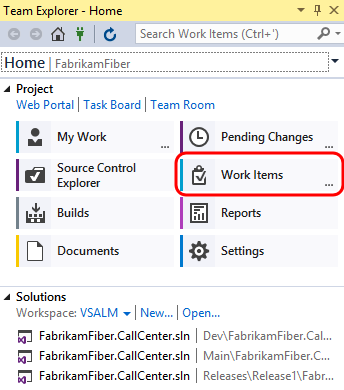
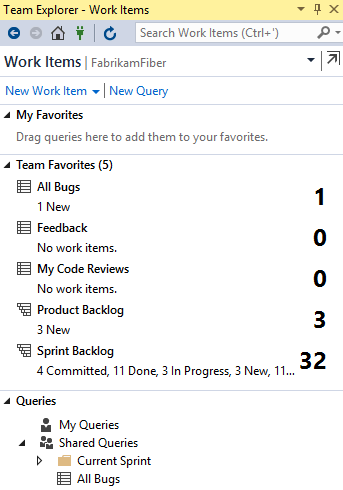
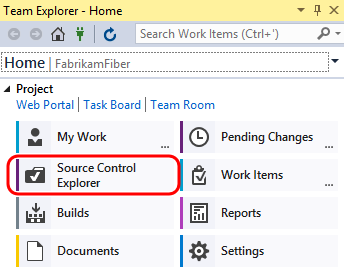
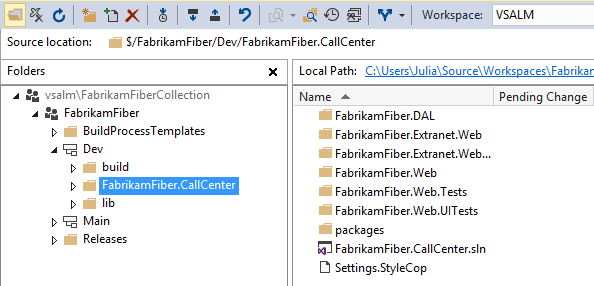
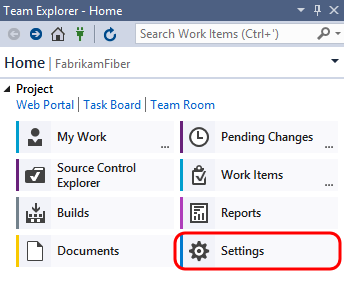
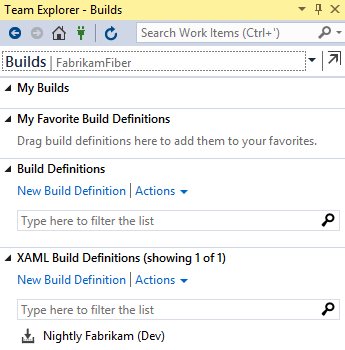
### About the Fabrikam Fiber Scenario

* 1. This set of hands-on-labs uses a fictional company, Fabrikam Fiber, as a backdrop to the scenarios you are learning about. Fabrikam Fiber provides cable television and related services to the United States. They are growing rapidly and have embraced Windows Azure to scale their customer-facing web site directly to end-users to allow them to self-service tickets and track technicians. They also use an on-premises ASP.NET MVC application for their customer service representatives to administer customer orders.
  2. In this set of hands-on labs, you will take part in a number of scenarios that involve the development and testing team at Fabrikam Fiber. The team, which consists of 8-10 people, has decided to use Visual Studio application lifecycle management tools to manage their source code, run their builds, test their web sites, and plan and track the project.

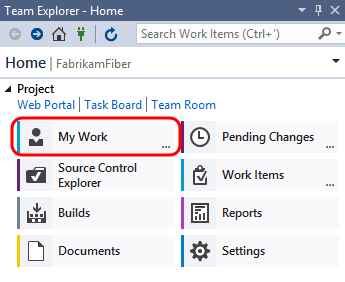
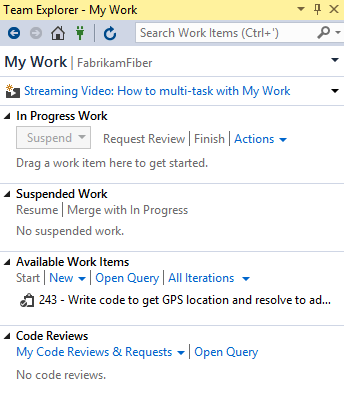
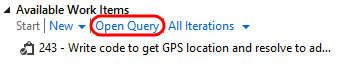
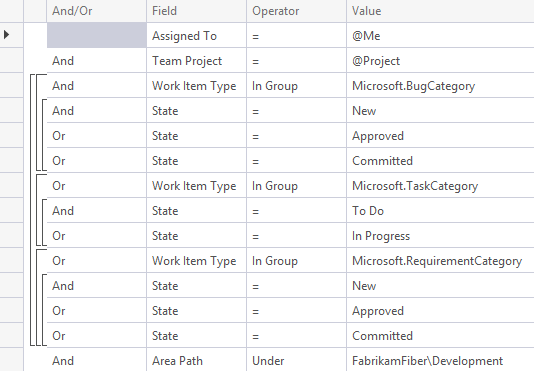
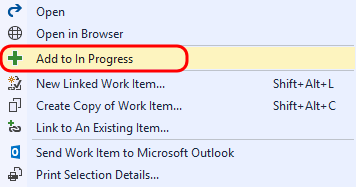
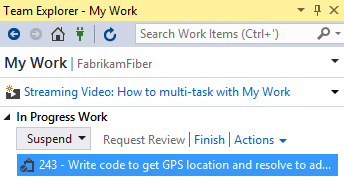
### Exercises

* 1. This hands-on lab includes the following exercises:
  2. Team Explorer Overview
  3. My Work
  4. Local Workspaces
  5. Pending Changes
  6. Suspend and Resume
  7. Code Reviews
  8. Merging
  9. Estimated time to complete this lab: **60 minutes**.

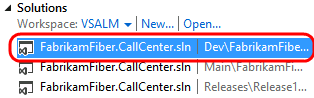
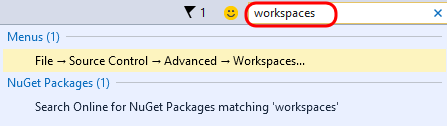
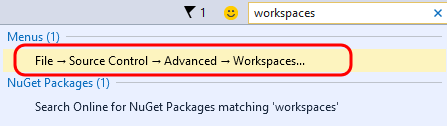
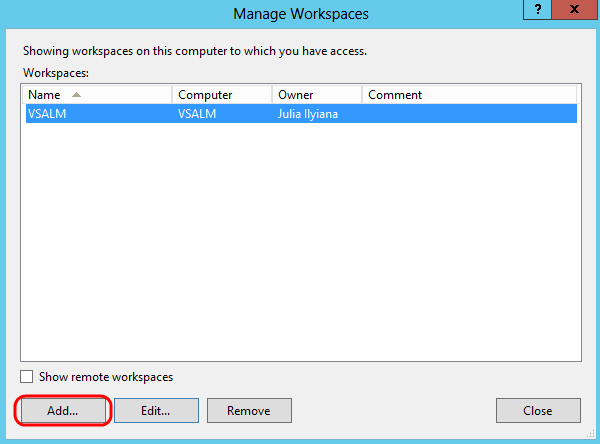
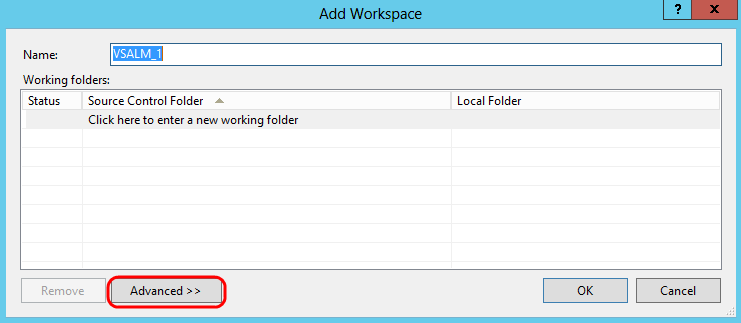
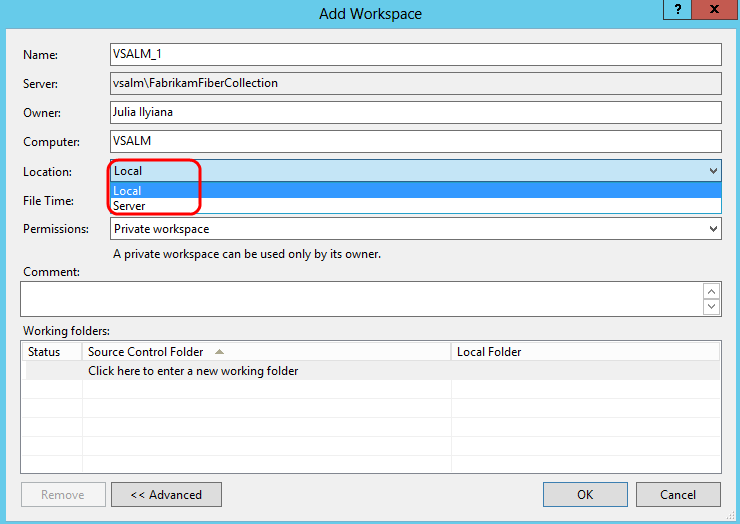
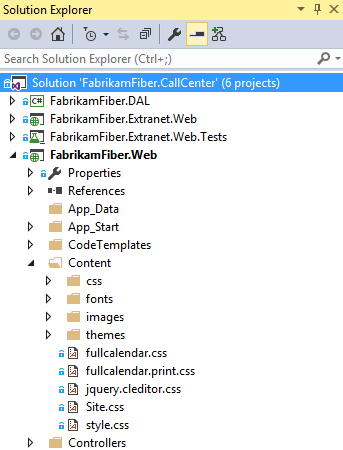
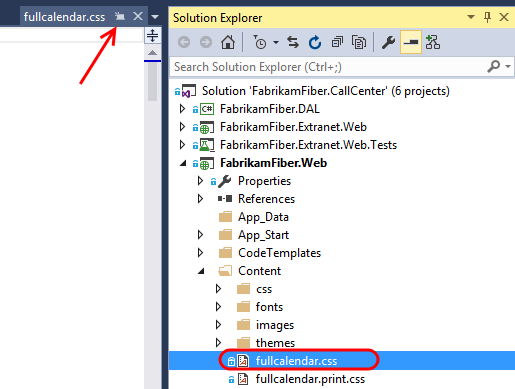
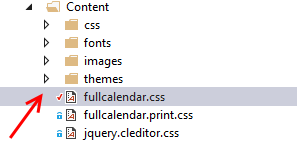
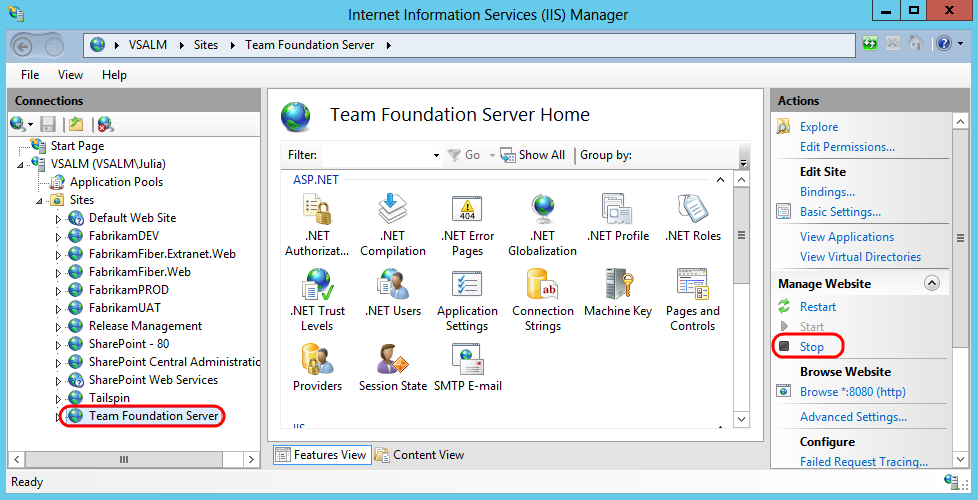
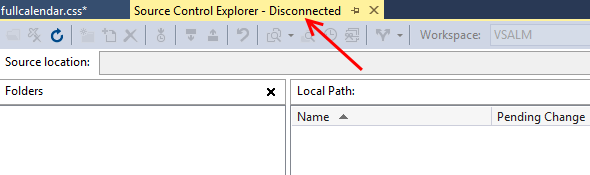
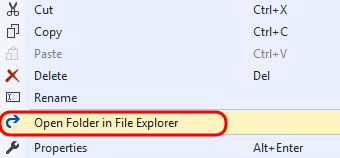
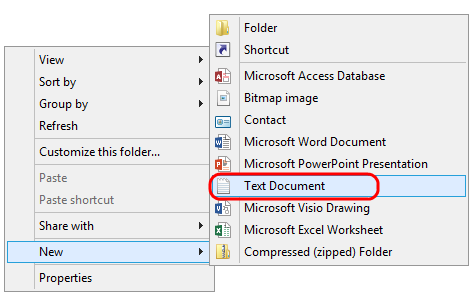
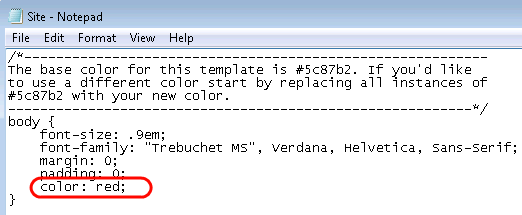
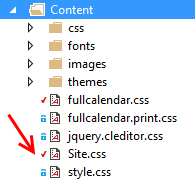
## Exercise 1: Team Explorer Overview

* 1. In this exercise, you will see a high-level view of the changes made to Team Explorer in this release and learn how to navigate the functionality that is made available there. If you are new to Team Foundation Server, you can think of Team Explorer as the primary way in which a Visual Studio user accesses Team Foundation Server functionality.
  2. Log in as **Julia** (VSALM\Julia). All user passwords are **P2ssw0rd**.
  3. Launch **Visual Studio 2015** from the taskbar.
  4. The **Team Explorer** window provides a central hub for common tasks that developers undertake when working on a project, including source control, work item tracking, collaboration features such as code review and team rooms, and even quick links to other resources found in the web portal such as task board.
     1. 
     2. Figure
     3. Team Explorer – Home window
  5. Navigation within **Team Explorer** operates similarly to a typical web application in that you follow links from the home screen, and can navigate backwards and forwards or return home at any time. We will put this navigation to use in the next exercise, as well as drill into the various hubs and pages exposed within Team Explorer.
     1. 
     2. Figure
     3. Navigation controls for Team Explorer
  6. Click **Work Items**.
     1. 
     2. Figure
     3. Work Items tile
  7. The **Work Items** view shows favorite queries for the logged-in user and the team, as well as provides the capability to add new queries and work items.
     1. 
     2. Figure
     3. Work Items view showing all queries
  8. Click the **back** arrow to return to the home view.
     1. 
     2. Figure
     3. Location of back arrow
  9. Click **Source Control Explorer**.
     1. 
     2. Figure
     3. Source Control Explorer tile
     4. 
     5. Figure
     6. Source Control Explorer window
  10. Click **Settings** to view the team project and team project collection options.
      1. 
      2. Figure
      3. Settings tile
      4. 
      5. Figure
      6. Team Explorer settings
  11. Click the **back** arrow once again to return to the home view.
  12. Click **Builds** to manage build definitions and view build history.
      1. 
      2. Figure
      3. Builds tile
      4. 
      5. Figure
      6. Builds window showing build definitions and past builds
  13. We will look at the new **My Work** functionality in the remaining exercises.

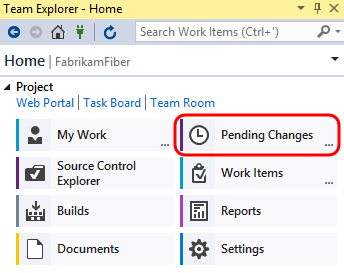
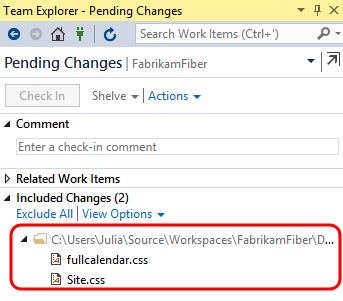
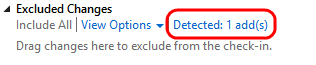
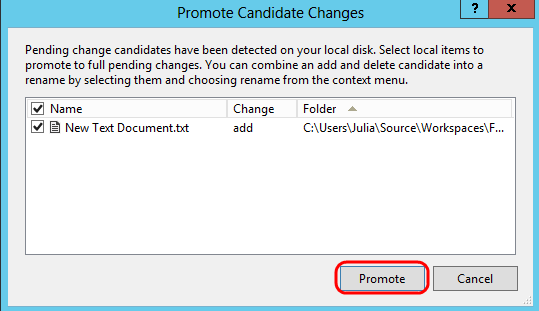
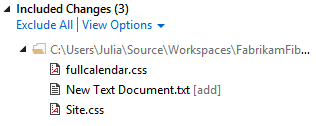
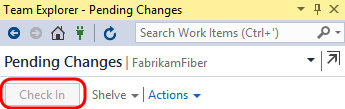
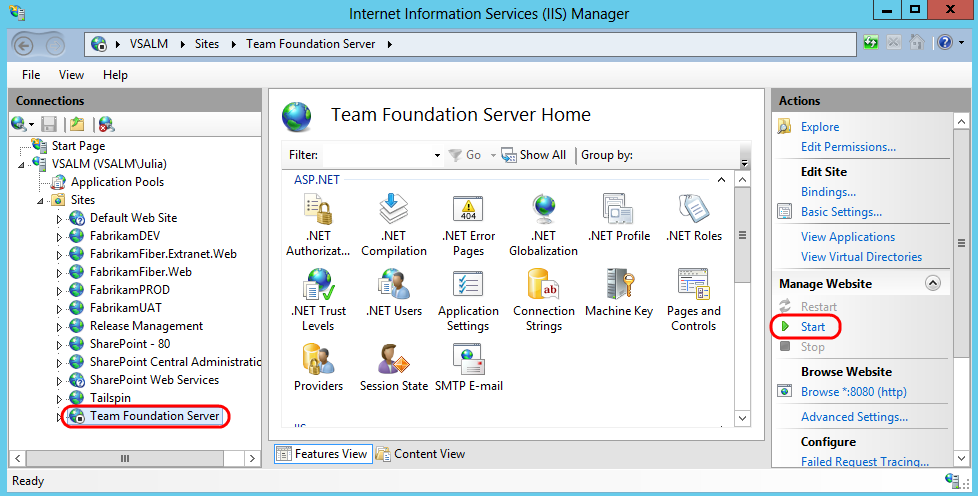
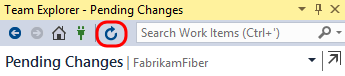
## Exercise 2: My Work

* 1. In this exercise, you will learn about the new My Work experience for managing your active tasks.
  2. In **Team Explorer - Home**, click **My Work**.
     1. 
     2. Figure
     3. My Work link
  3. The **My Work** node shows work items in progress, available work items, and suspended and shelved work for the currently logged in user. This workflow uses a task-based development model to help keep team members in-sync with each other.
     1. 
     2. Figure
     3. My Work window showing available and in progress work
  4. Click the **Open Query** hyperlink that is located in the **Available Work Items** section.
     1. 
     2. Figure
     3. Open Query link
  5. Click **Edit Query**.
     1. 
     2. Figure
     3. Viewing the default Available Work Items query definition
     4. 
     5. Figure
     6. Query clauses for the Available Work Items query
     7. **Note:** The types of work items offered and whether or not they fit into the “My Work” concept is dependent upon the process template. In the Scrum process template, this includes tasks.
  6. **Close** the query window without saving changes.
  7. Let’s say that we want to work on one of our available work items. On the **My Work** page in **Team Explorer**, **right-click** on the item that starts with “**Write code to get GPS location…**” and choose the **Add to In Progress** option from the context menu. This communicates the current state of the work item to our team.
     1. 
     2. Figure
     3. Right-click on work items to move them between states
     4. 
     5. Figure
     6. Resulting view of My Work after moving work from Available to In Progress

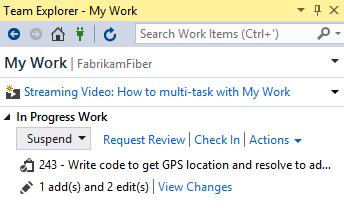
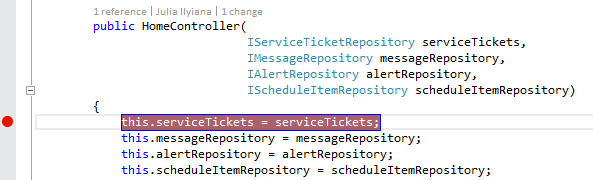
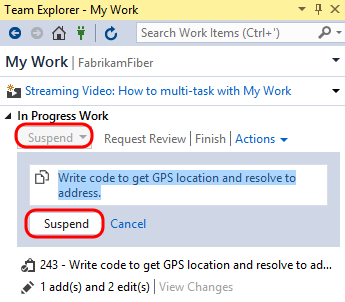
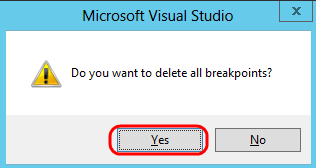
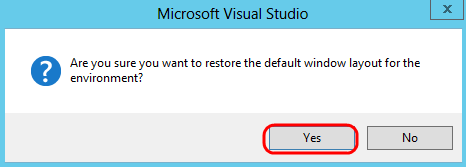
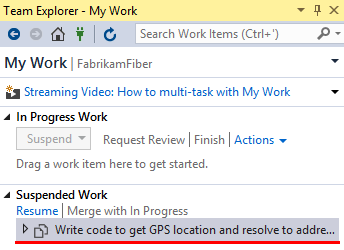
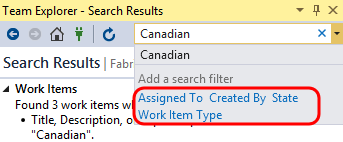
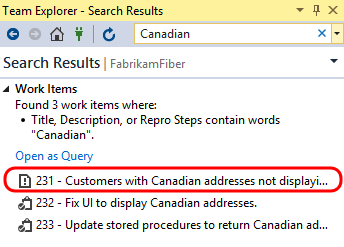
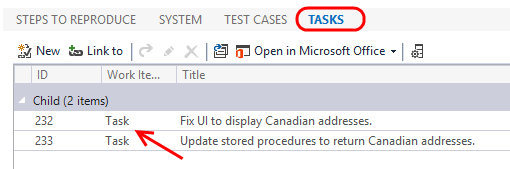
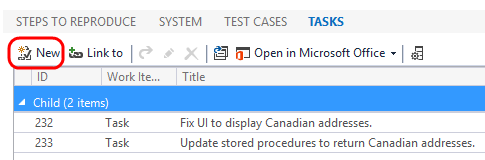
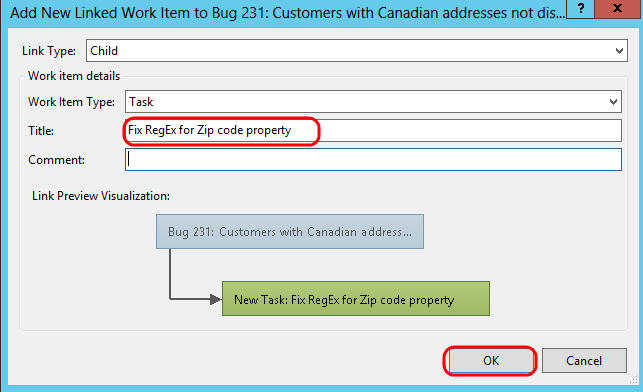
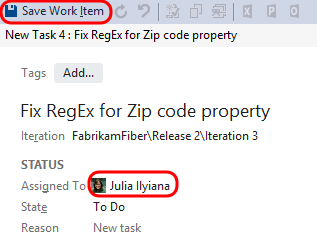
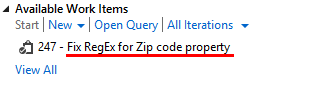
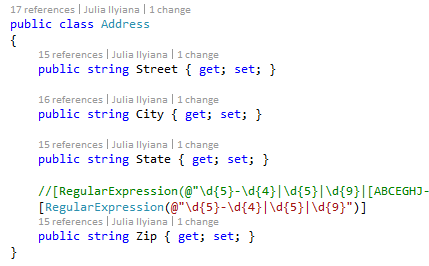
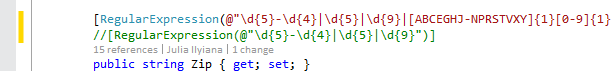
## Exercise 3: Local Workspaces

* 1. In this exercise, you will learn about local workspaces and how they improve the offline development experience.
  2. In **Team Explorer – Home**, **double-click** on the first **FabrikamFiber.CallCenter.sln** solution.
     1. 
     2. Figure
     3. Loading Fabrikam Fiber solution
  3. Server-based workspaces used to be the only option available in Team Foundation Server (prior to 2012). With server workspaces, source control operations are performed exclusively on the server, so working in a disconnected, offline scenario can be tedious. **Local workspaces**, which are now default, improve the development experience for smaller teams and projects. They provide a more client-centric workflow, a style of version control often described as **Modify-Merge-Commit**.
  4. As an aside, server-based workspaces, which are better suited for larger projects, can certainly still be used if desired. Load the Manage Workspaces window by typing ‘**workspaces’** into the **Quick Launch** search box in the top-right corner of Visual Studio and wait a second or two for the results to load.
     1. 
     2. Figure
     3. Location of Quick Launch bar
     4. **Note:** The **Quick Launch** search box can be used to find most of the settings, tools, and windows available in Visual Studio, no matter where they are found, saving you precious time clicking through menus. Use the **Ctrl+Q** key combination to avoid the mouse altogether.
  5. Select the **Workspaces…** command link.
     1. 
     2. Figure
     3. Selecting the Workspaces link
  6. In the **Manage Workspaces** window, click **Add…** to start creating a new workspace.
     1. 
     2. Figure
     3. Location of Add button to create new workspace
  7. In the **Add Workspace** window, click **Advanced** to see all available options.
     1. 
     2. Figure
     3. Location of Advanced button to view advanced workspace properties
  8. Locate the **Location** workspace property. By default, new workspaces will use local workspaces, but you can choose to create a server-based workspace if desired.
     1. 
     2. Figure
     3. Choosing between Local and Server workspace
  9. Close the workspace windows by pressing the **Escape** key twice.
  10. In **Solution Explorer**, expand the **FabrikamFiber.Web\Content** folder.
      1. 
      2. Figure
      3. Navigating to the Content folder of FabrikamFiber.Web project
  11. **Single-click** the *fullcalendar.css* file to view its contents. Note that the file is loaded in the CSS Source Editor as expected, but the tab for it is shown on the right-hand side. This new tab type is called a **preview** tab – as you use the arrow keys to navigate between files or single-click on them, the preview tab will be replaced with the most recent selection. Preview tabs help address the problem where you can easily overwhelm your Visual Studio tab real estate with files that you only need temporarily.
      1. 
      2. Figure
      3. Preview tab
      4. **Note:** There is a button on the preview tab itself that allows you promote it to a regular tab (see the arrow in the screenshot above).
  12. Make an arbitrary change to the ***fullcalendar.css*** file by changing one of the style properties. Visual Studio will indicate that the file has been checked out, but keep in mind that this has only been done in the local workspace. Other team members will not see that you are working on this file.
      1. 
      2. Figure
      3. Check mark next to file shows that you have made changes to it locally
  13. Two of the major benefits of the local workspaces are that the disconnected, offline work scenario is greatly improved, and it is much easier to use external tools to work with files in source control. To demonstrate both of these benefits in action, let’s proceed by stopping the Team Foundation Server site in IIS. This will simulate a scenario where you are working offline, such as writing code from the beach. In a real-world scenario, you would never write code from the beach - you would be enjoying your vacation.
  14. Launch **Internet Information Services (IIS) Manager** from the taskbar.
  15. In IIS Manager, expand the **VSALM** server node and the **Sites** node to find and select the **Team Foundation Server** site.
  16. In the **Actions** pane, click **Stop**.
      1. 
      2. Figure
      3. Stopping Team Foundation Server
  17. Return to Visual Studio, load the **Source Control Explorer** window, and click the **refresh** button. Note that the window is now in a **disconnected** state.
      1. 
      2. Figure
      3. Source Control Explorer in disconnected state
  18. Even though we are now in a disconnected state with Team Foundation Server, we can continue to do work. **Right-click** the **Content** folder in Solution Explorer and select the **Open Folder in File Explorer** option.
      1. 
      2. Figure
      3. Location of option to open folder in Explorer
  19. Add a new file into the Content folder by right-clicking in a blank area of the view and selecting **New | Text Document** from the context menu. Accept the default name.
      1. **Note:** We will use this file in a demonstration a little later – the contents are not important and can be left blank.
      2. 
      3. Figure
      4. Creating a new text document
  20. In the Explorer window, **right-click** on the ***Site.css*** file and select **Open With | Notepad** from the context menu.
  21. In Notepad, change the ‘**body**’ style color to ‘**red**’.
      1. 
      2. Figure
      3. Editing a file in Notepad and saving changes
  22. In Notepad, **save** and **close** the Site.css file and return to **Solution Explorer**. Note that Visual Studio detected the change to the file and that the file was not read-only as it would have been in the case of a server-based workspace.
      1. 
      2. Figure
      3. Visual Studio shows that the file has changed

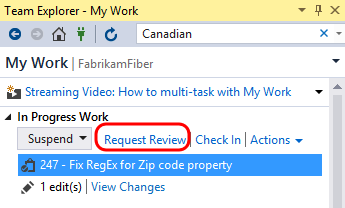
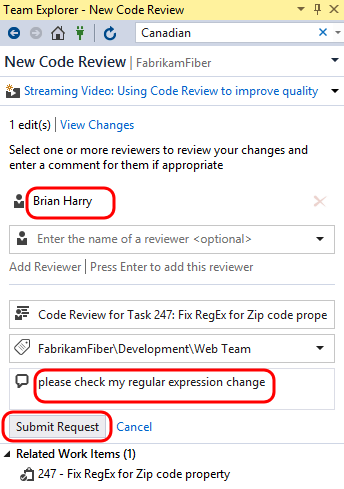
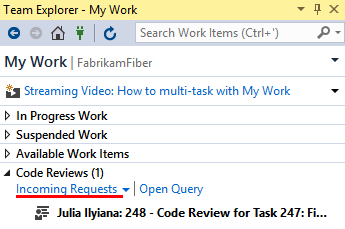
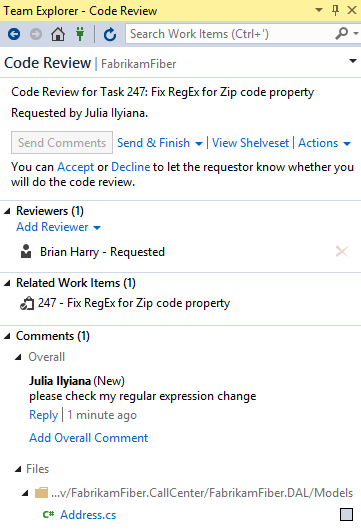
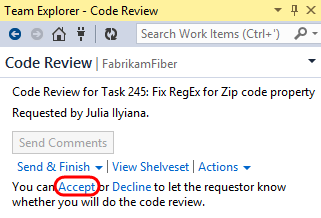
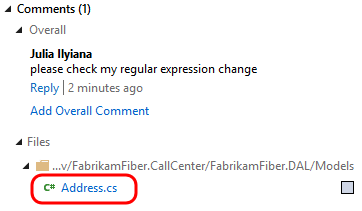
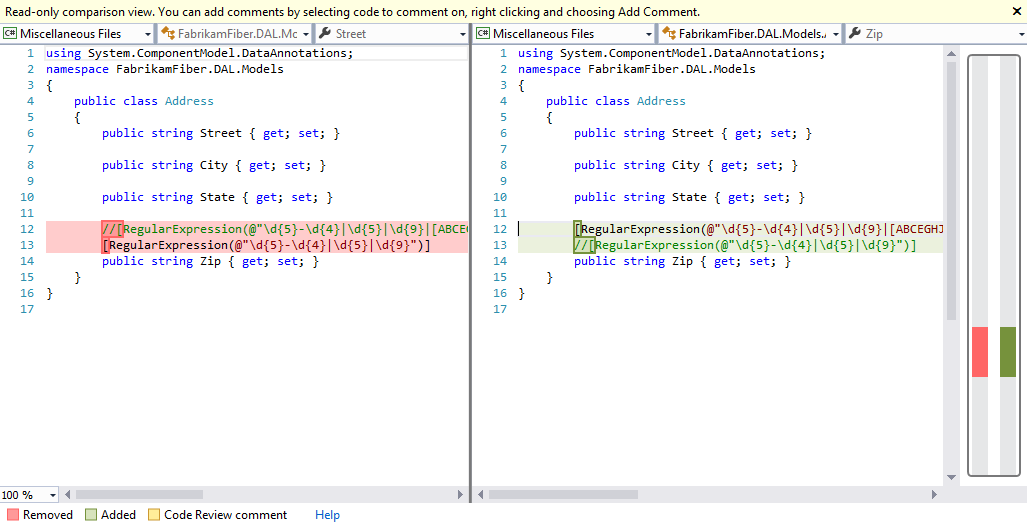
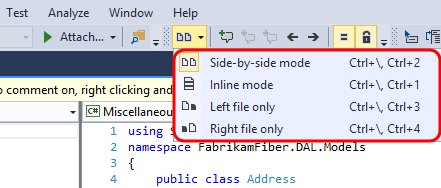
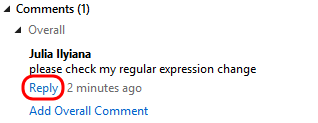
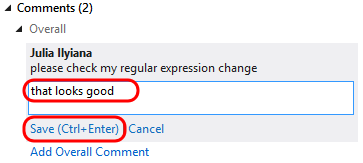
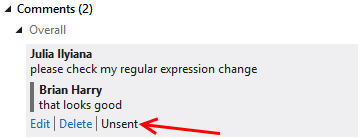
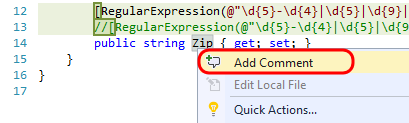
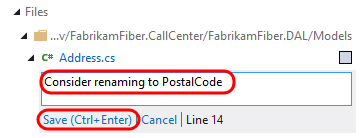
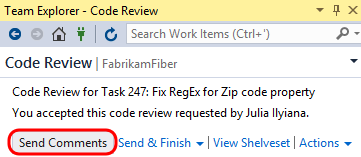
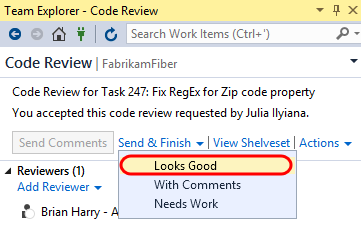
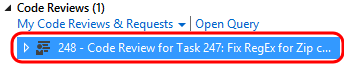
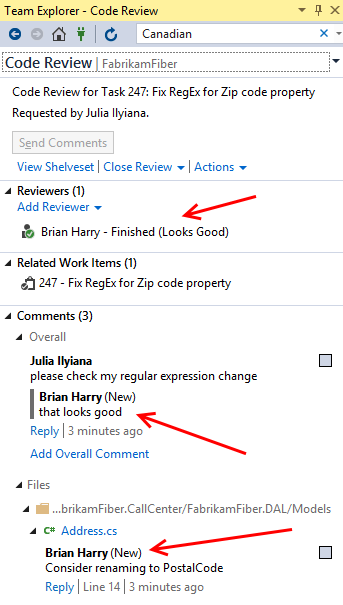
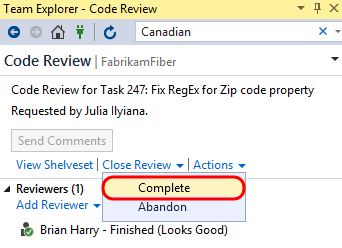
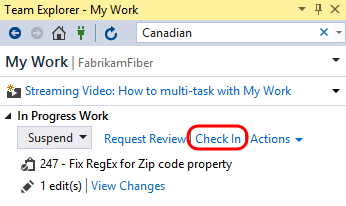
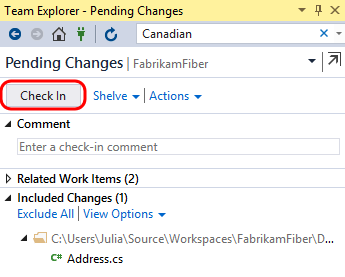
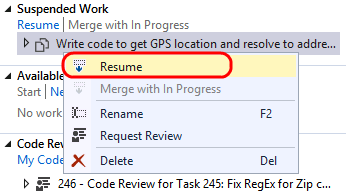
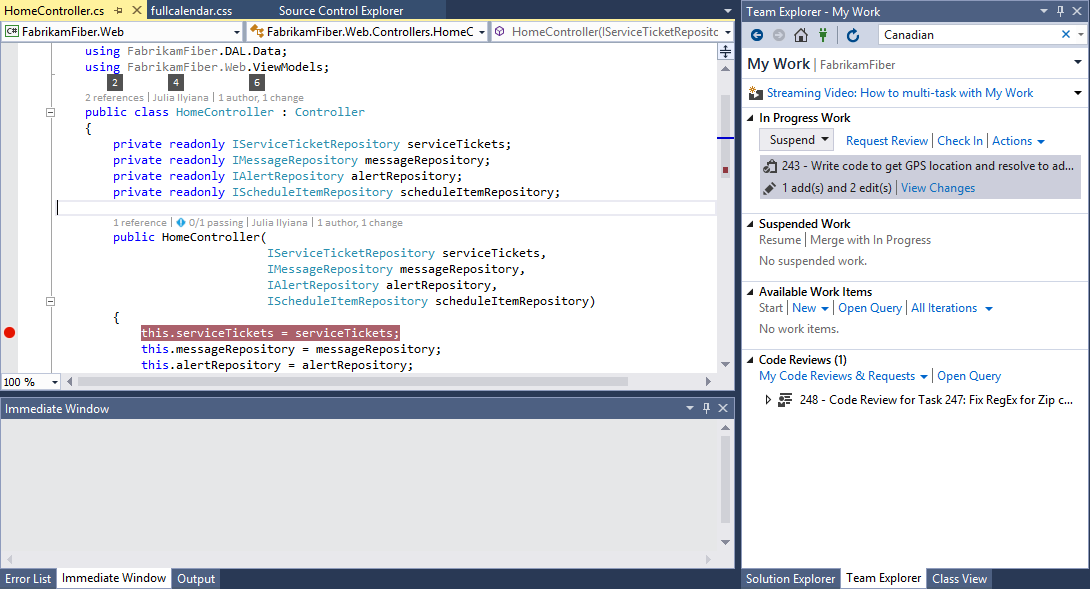
## Exercise 4: Pending Changes

* 1. In this exercise, you will learn about the Pending Changes view within Team Explorer.
  2. In **Team Explorer** **- Home**, click **Pending Changes**.
     1. 
     2. Figure
     3. Pending Changes link
     4. **Note:** You should see an error message in the Pending Changes window about being unable to connect to the remote server, but you can simply ignore it. We purposefully stopped Team Foundation Server during the previous exercise.
  3. In the **Pending Changes** window, we can see that the two changes made to existing files in the previous exercise automatically show up under the **Included Changes** section.
     1. 
     2. Figure
     3. Pending Changes window showing included changes
  4. In the **Excluded Changes** section, there is a detected change that was not automatically marked as an included change. Click the **Detected** link.
     1. 
     2. Figure
     3. Some changes are detected but not automatically promoted
  5. In the **Promote Candidate Changes** window, you can see the addition of the new text document. Click **Promote** to move it to the set of changes that we want to include in the next check-in.
     1. 
     2. Figure
     3. Promoting changes manually
  6. You should now see three included changes.
     1. 
     2. Figure
     3. Updated view showing all included pending changes
  7. Although we could continue to work on our project while offline, note that we can’t perform a check-in at this time as the **Check In** button is currently disabled.
     1. 
     2. Figure
     3. Can’t check in changes when offline
  8. In the Internet Information Services (IIS) Manager window, select the **Team Foundation Server** site and then click **Start** in the Actions pane.
     1. 
     2. Figure
     3. Starting Team Foundation Server
  9. Return to Visual Studio and then click the **refresh** button in the **Team Explorer – Pending Changes** window. We should now be working in a connected state again with the **Check In** button enabled once again. **Do not check in your changes yet**.
     1. 
     2. Figure
     3. Refreshing the Pending Changes window to connect to Team Foundation Server

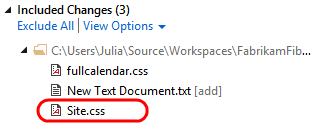
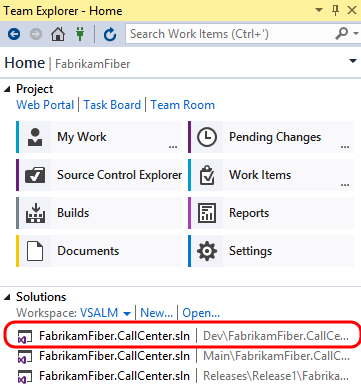
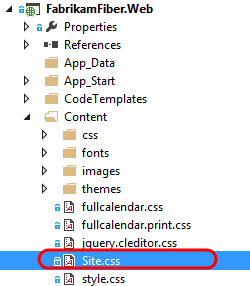
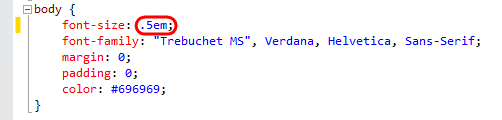
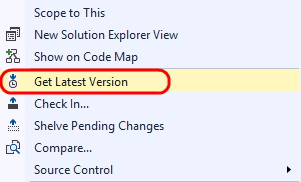
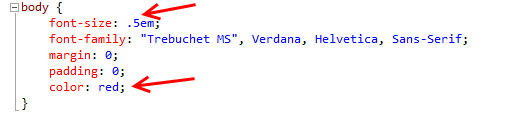
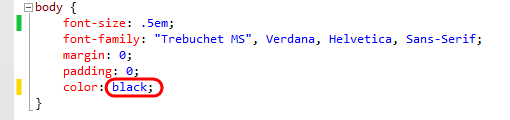
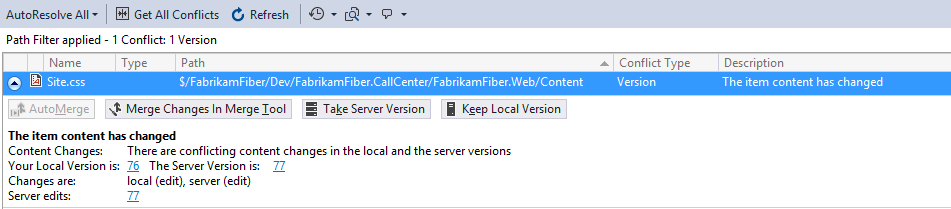
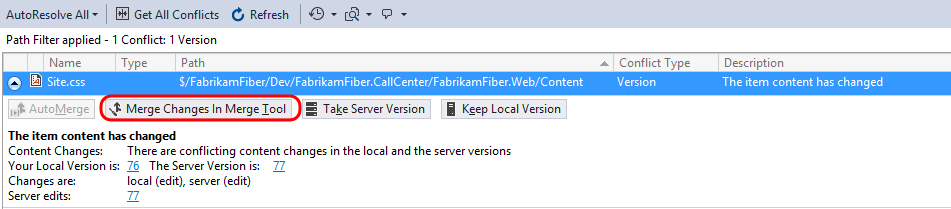
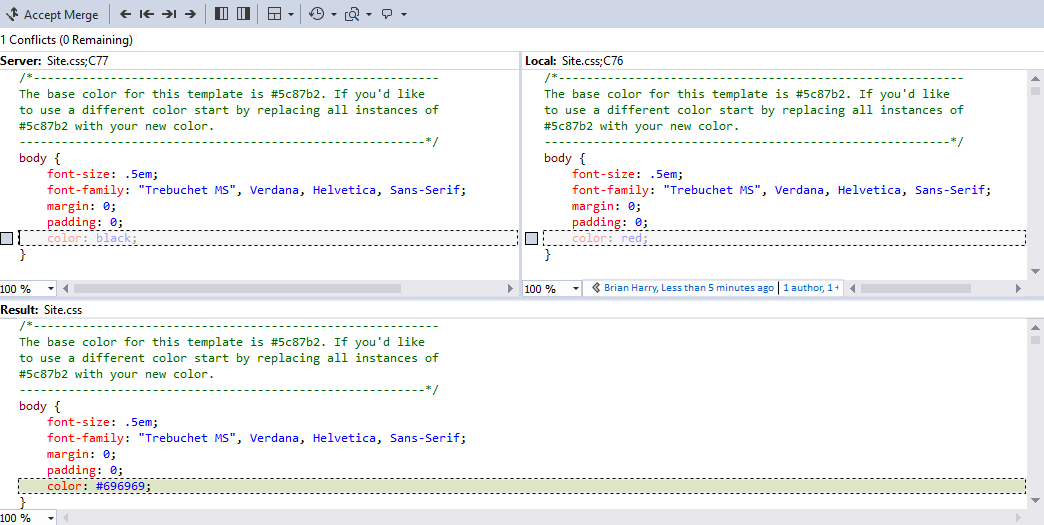
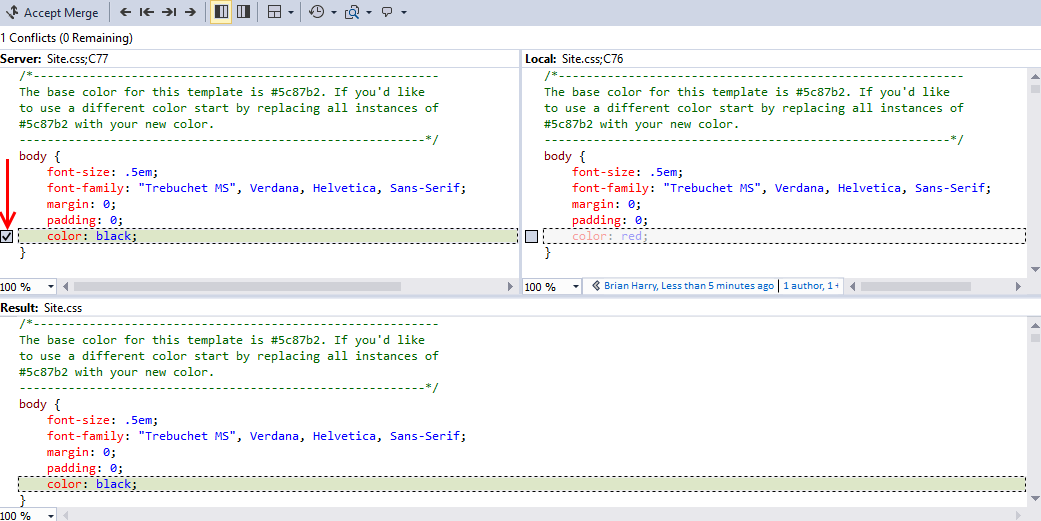
## Exercise 5: Suspend and Resume

* 1. In this exercise, you will learn about the suspend and resume functionality for work items and the integrated code reviews.
  2. Another common but unavoidable difficulty faced by developers is context switching between different tasks. In past versions of Team Foundation Server, this was addressed with **Shelvesets** that would preserve your changes on the server even if you weren’t ready to check those into source control. In more recent versions, this has been taken one step further with the **Suspend** and **Resume** feature – in addition to creating a shelveset of changes, some other data like breakpoints and tool window positions are also saved locally.
  3. We begin this exercise in the state that we ended the previous exercise in - we have a set of pending changes to be made as part of the work that we have in progress. Load the **My Work** window in Team Explorer to view the work that is in progress.
     1. 
     2. Figure
     3. My Work showing tasks in all states
  4. As part of this scenario, imagine that we also have some other debugging tool windows open and a break point set as we debug code that we are writing. Double-click on ***HomeController.cs*** from **FabrikamFiber.Web\Controllers** and add a breakpoint to the first line in the HomeController constructor.
     1. 
     2. Figure
     3. Setting a breakpoint
  5. Open the **Immediate** window from **Debug | Windows | Immediate**.
  6. Now imagine that your boss asks you to stop working on the current task, and work on a high-priority bug instead. We currently have a lot of time invested in getting Visual Studio debugging tools in place, and we would like to be able to return to this current context after the bug fix is complete.
  7. Press **Ctrl+Shift+S** to save all open files.
  8. In Team Explorer, navigate to the **My Work** screen, select the **Suspend** drop-down, and then click **Suspend**. We could modify the description, but since we are currently working on a work item, it is pre-populated for us.
     1. 
     2. Figure
     3. Suspending work
  9. Clear out all breakpoints by pressing **Ctrl+Shift+F9**, and then click **Yes** to confirm the action.
     1. 
     2. Figure
     3. Deleting all breakpoints
  10. Close all document windows by selecting **Window | Close All Documents** from the main menu.
  11. Reset to the default window layout by selecting **Window | Reset Window Layout** from the main menu. Click **Yes** when asked to confirm the action.
      1. 
      2. Figure
      3. Restoring default window layout
  12. At this point, you should have a very clean looking workspace and the previous task that you were working on should be listed in the **Suspended Work** section of the **My Work** window.
      1. 
      2. Figure
      3. Suspended section of My Work
  13. Note the search box for work items at the top of the Team Explorer window. This will perform a search of all work items in the currently selected team project. Type “**Canadian**” into the search box and press the **Enter** key to locate the high-priority bug that you have been asked to work on.
      1. 
      2. Figure
      3. Searching for work items
      4. **Note:** This quick search functionality searches the team project Title, Description, and Repro Steps fields by default for the keyword provided. You can also click the drop-down arrow to show your search history, and to add search parameters for metadata such as work item type, or whom the work item is assigned to.
      5. ****
  14. **Double-click** the bug titled “**Customers with Canadian addresses not displaying properly”**.
      1. 
      2. Figure
      3. Location of high-priority bug that needs fixed
  15. In the window displaying the form for the bug, select the **Tasks** link to view the linked work items. Note that there are two child tasks.
      1. 
      2. Figure
      3. Viewing existing child tasks
  16. Imagine that other team members are tackling these child tasks already, but that we know of another problem related to the parent bug that also needs to be fixed. Click **New** in the Tasks section of the form to create a new child task.
      1. 
      2. Figure
      3. Creating a new child task
  17. For the new linked work item, select **Link Type** of **Child**, **Work Item Type** of **Task**, enter a **Title** of “**Fix RegEx for Zip code property**”, and then click **OK**.
      1. 
      2. Figure
      3. Creating a new child task
  18. Assign the new task to **Julia Ilyiana** and then click **Save Work Item**.
      1. 
      2. Figure
      3. Assigning the new work item
  19. Return to the **Team Explorer – My Work** window. You should now see the new task appear in the **Available Work Items** section.
      1. 
      2. Figure
      3. Location of the new task in My Work
  20. **Right-click** on the new task entitled “**Fix RegEx for Zip code property**”, and then select the **Add to In Progress** option from the context menu.
  21. In **Solution Explorer**, open the ***Address.cs*** file from the **Models** folder of the **FabrikamFiber.DAL** project. The **Address.Zip** property is currently using an incorrect regular expression data annotation, with the result being that users of the web application can enter invalid postal codes for Canadian addresses.
      1. 
      2. Figure
      3. Source code with bug
  22. Let’s assume that the correct fix is to update the regular expression. Comment out the existing regular expression and then un-comment the other one.
      1. 
      2. Figure
      3. Source code with bug fixed
  23. Press **Ctrl+S** to save the changes to the file.

## Exercise 6: Code Reviews

* 1. Now that the fix has been made, let’s ask Brian to perform a code review on the changes to make sure things look correct. In **Team Explorer - My Work**, click the **Request Review** link.
     1. 
     2. Figure
     3. Location of Request Review link
  2. Type **Brian Harry** for the name of the reviewer and then press the **Enter** key, type a description of “**please check my regular expression change**”, and then click **Submit Request**.
     1. 
     2. Figure
     3. Entering details for code review request
  3. Log in as **Brian Harry** to perform the code review. The password for all users in this virtual machine is **P2ssw0rd**. The easiest way to do this is to use **Remote Desktop** and connect to **VSALM**. You can then close or minimize that session later on when returning to Julia’s account.
  4. Launch **Visual Studio**, open **Team Explorer**, and navigate to **My Work**. The new code review request is shown under the **Code Reviews** section at the bottom. Filter for **Incoming Requests.**
     1. 
     2. Figure
     3. Code Reviews section of My Work
  5. **Double-click** the code review request to open the **Code Review** window. This view allows Brian Harry to see a lot of information regarding the request so that he is able to determine whether he will accept or decline the task. He can view the entire shelveset of changes, see the other reviewers or even add someone to the reviewers list, see the related work items, comments related to the code review request, and finally the ability to show a diff view for all of the changed files with a single click.
     1. 
     2. Figure
     3. Code Review window
  6. In the **Code Review** window, click the **Accept** link to start working on the code review.
     1. 
     2. Figure
     3. Location of Accept button
  7. Click **Address.cs** from the comments section to open it with the inline diff tool.
     1. 
     2. Figure
     3. Location of file with bug fix
  8. We are now viewing the changes using the default side-by-side comparison mode. Red background colors represent changes in the old version (left) and green colors represent changes in the new version (right). Note the legend available at the bottom of the comparison view.
     1. 
     2. Figure
     3. Inline diff tool showing difference between old and new version of code
  9. In the Compare Files toolbar, select the **compare mode** drop down button to view the various display modes that the diff viewer can use. Try out a few of the different modes.
     1. **Note:** If the Compare Files toolbar is not open, open it from **View | Toolbars | Compare Files** from the main menu in Visual Studio.
     2. 
     3. Figure
     4. Location of button to choose compare mode
  10. Let’s assume that we agree that the changes are correct, so let’s ahead and add a review-level comment to that effect. Click the **Reply** link underneath Julia’s message.
      1. 
      2. Figure
      3. Location of Reply link to reply to comment
  11. For the comment, enter “**that looks good**” and then click **Save**.
      1. 
      2. Figure
      3. Entering a response to a comment
  12. Note that the new comment has the text “**Unsent**” next to it. The comment will not be sent until Brian is done with the review.
      1. 
      2. Figure
      3. Comment is not sent automatically
  13. You can also annotate individual files and blocks of code with comments. In the diff viewer, use the mouse to highlight the **Zip** property name, **right-click**, and select **Add Comment** from the context menu.
      1. 
      2. Figure
      3. Adding a comment to code
  14. In the comment box, enter “**Consider renaming to PostalCode**” and then click the **Save** link.
      1. 
      2. Figure
      3. Annotating code with a comment
  15. Click **Send Comments** near the top of the Code Review window to send the comments that we have batched up so far. Having the ability to send comments back right away can be beneficial for a code review that requires more work than this one. If the reviewer finds an issue, the developer can immediately start fixing the issue rather than waiting for the reviewer to finish.
      1. 
      2. Figure
      3. Send Comments link
  16. Select the **Send &** **Finish** drop down followed by the **Looks Good** option to end the code review.
      1. 
      2. Figure
      3. Ending a code review
  17. Switch users back to **Julia** once again.
  18. Return to **Team Explorer – My Work** and double-click on the code review item to open it.
      1. 
      2. Figure
      3. Code review has been completed
  19. In **Team Explorer – Code Review**, note that Brian finished his review and had comments.
      1. 
      2. Figure
      3. Code Review window showing updates after code review
  20. For this scenario, let’s say that we are done with the code review process. Select the **Close Review** link and select the **Complete** option.
      1. 
      2. Figure
      3. Closing the review as being completed
  21. Now that we are confident that we have fixed the high priority bug, we can check-in the changes. Click the **Check In** link to load the **Pending Changes** window and then click the **Check In** button to complete the task. Click **Yes** if asked to conform check-in.
      1. 
      2. Figure
      3. Check in changes
      4. 
      5. Figure
      6. Check in changes
  22. Now we can return to the code that we were originally working on before we were interrupted by the bug fix. In the **My Work** window, **right-click** on the task listed under the **Suspended Work** section, and then select the **Resume** option from the context menu.
      1. 
      2. Figure
      3. Resuming suspended work
  23. After the shelveset is opened and the task context is restored, you will notice that the document windows and tools windows, as well as their positions, are restored to the state that we left them in when we suspended this task. Although you won’t notice this in a virtual machine, had you been using multiple monitors you would have also noticed that all of the tool windows would be opened on their respective monitors. This capability allows you to quickly get in and out of tasks to help you deal with the frequent interruptions that many of us face on a daily basis.
      1. 
      2. Figure
      3. Restored development environment

## Exercise 7: Merging

* 1. In this exercise, you will learn about the merging experience in Visual Studio 2015.
  2. You should still be logged in as **Julia** from the previous exercise.
  3. In **Team Explorer – Pending Changes**, you should see three changes listed under the **Included Changes** section. Double-click on ***Site.css*** to open it.
     1. 
     2. Figure
     3. Opening Site.css
  4. You may recall that that we previously changed the body color property to red. Now we will take a look at what happens if another user makes a different change to the same file and then checks it in.
  5. Switch users back to **Brian Harry**. All user passwords are **P2ssw0rd**.
  6. In Visual Studio 2015, open the **FabrikamFiber.CallCenter** solution from the **Dev** branch.
     1. 
     2. Figure
     3. Loading Fabrikam Fiber solution
  7. From **Solution Explorer**, open ***Site.css*** from the **Content** folder of the **FabrikamFiber.Web** project.
     1. 
     2. Figure
     3. Location of Content folder in FabrikamFiber.Web project
  8. Let’s make a different change than Julia is going to make, so change the body **font-size** property from .9em to **.5em**.
     1. 
     2. Figure
     3. Making a change to a style file
  9. Press **Ctrl+Shift+S** to save all documents.
  10. In **Team Explorer – Pending Changes**, click **Check In**. If you are prompted to confirm the check-in, please do so.
  11. Switch users back to **Julia** once again.
  12. In **Solution Explorer**, **right-click** on ***Site.css*** and select **Get Latest Version** from the context menu.
      1. 
      2. Figure
      3. Get latest version of style file
  13. In the ***Site.css*** file, note that the body font-size property has been updated with Brian’s change, but that we also still have our current change where we updated the color property to red. This demonstrates that, wherever possible, Team Foundation Server and Visual Studio attempt to auto-merge changes, rather than needing the user to be involved with the merge process. This can save you several steps when compared to previous versions.
      1. 
      2. Figure
      3. Style file showing auto-merged update
  14. **Switch users** back to **Brian Harry**.
  15. Let’s create a change that will force a manual merge process. In ***Site.css***, change the body **color** property to **black**.
      1. 
      2. Figure
      3. Changing style with a change that will conflict with another user
  16. Press **Ctrl+Shift+S** to save all documents.
  17. **Check in** the pending change.
  18. **Switch users** back to **Julia**.
  19. Imagine that Julia is blissfully unaware that Brian has changed the body color property to black. In **Solution Explorer**, **right-click** on ***Site.css*** and select **Get Latest Version** from the context menu. This will load the **Resolve Conflicts** window.
      1. 
      2. Figure
      3. Resolve Conflicts window
  20. Click **Merge Changes In Merge Tool** to begin the manual merge process. If you are prompted to save and close Site.css, go ahead and click **Yes** to continue.
      1. 
      2. Figure
      3. Location of button to start Merge Tool
  21. The new merge process also uses the integrated diff viewer. By default, the merge view uses a side-by-side view to show the changes. Note that you can see that the server version (left) has the black color, the local version (right) has the red color, and the result (below) currently shows the original color.
      1. 
      2. Figure
      3. Using Merge Tool to reconcile differences
  22. Let’s go with server version. Select the checkbox to the left of the body **color** property choice that uses **black** (in the left pane).
      1. 
      2. Figure
      3. Manually merging changes
  23. In the Merge Toolbar, click the **Accept Merge** button.
      1. 
      2. Figure
      3. Accepting the merge that we made manually

To give feedback please write to [VSKitFdbk@Microsoft.com](mailto:VSKitFdbk@Microsoft.com)

Copyright © 2016 by Microsoft Corporation. All rights reserved.