**Contoso Jobs**

**Lab 0. Explore the Contoso Jobs UWP App**

In this Lab you will explore an existing app called Contoso Jobs. Through the other tutorials in this series you will modify this application to improve it as a Surface Hub application. At this point the application will work on a Surface Hub, yet there are many areas in which it can be improved to become an optimized Surface Hub app.

As you follow the steps in this exercise you will discover the features of the existing app that need to be changed.

**Instructions:**

1. Open and test the existing application.   
   Find the folder named **0. Start Contoso Jobs**  
   In this folder open the **Contoso Jobs.sln** solution file in **Visual Studio 2015**
2. In Visual Studio run the application, **Debug | Start Debugging (F5)**  
   (ensure the target is set to x86 or x64)
3. The Contoso Jobs application should deploy and run on your dev machine.  
   You will see the Windows App has three columns; To Do, WIP (work in progress), and Done.

The idea is that each new Job (or task) that gets created will start in the To Do list. When a job is started it should be moved to the Work In Progress list, and when completed a job should be moved to the Done list. This way you can manage jobs to be done, make sure you are not trying to do too much at the same time, and track what has been completed. At a high level this follows the principles of Kanban.

1. From the Hamburger menu in the top left of the app you can create a new Job. You will notice this displays a modal dialog, where you can enter the Title and Description of the new Job. The Hamburger menu is going to be hard to reach for multiple people at a Surface Hub, this needs to be replaced.

The modal dialog to create a new Job is blocking other people from working with the Job board at the same time. A new way for creating jobs will be needed.

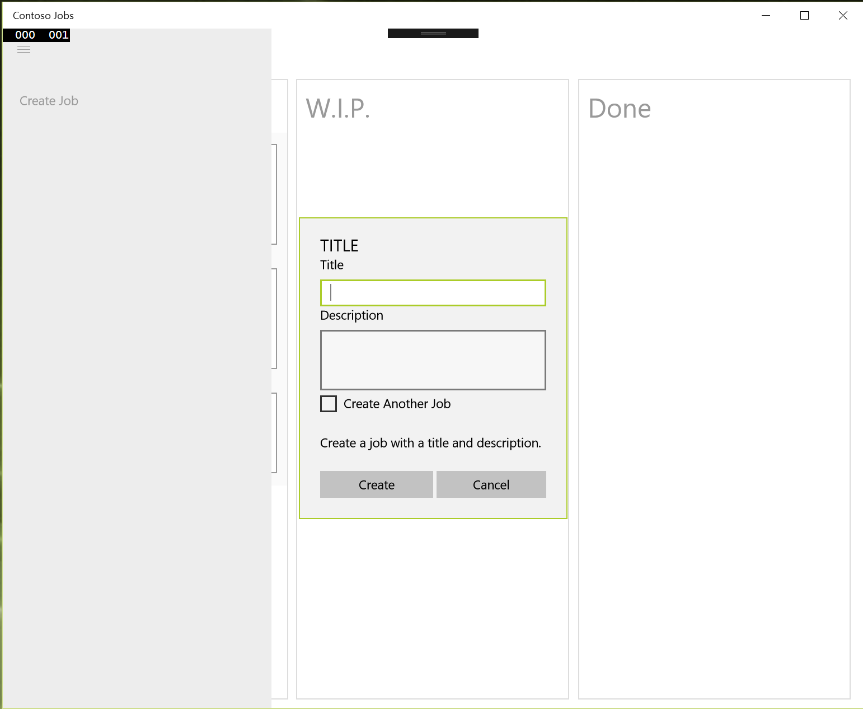


Figure Create a new Job

1. When you create the Job, it gets placed in the To Do list. All the Jobs in the To Do list have a ‘play’ button, this can be used to start the Job and move it into the WIP column. Take note that the Play button has a tool tip when you hover your mouse over it. With a touch interface there is no hover, the tooltip will need to be replaced.

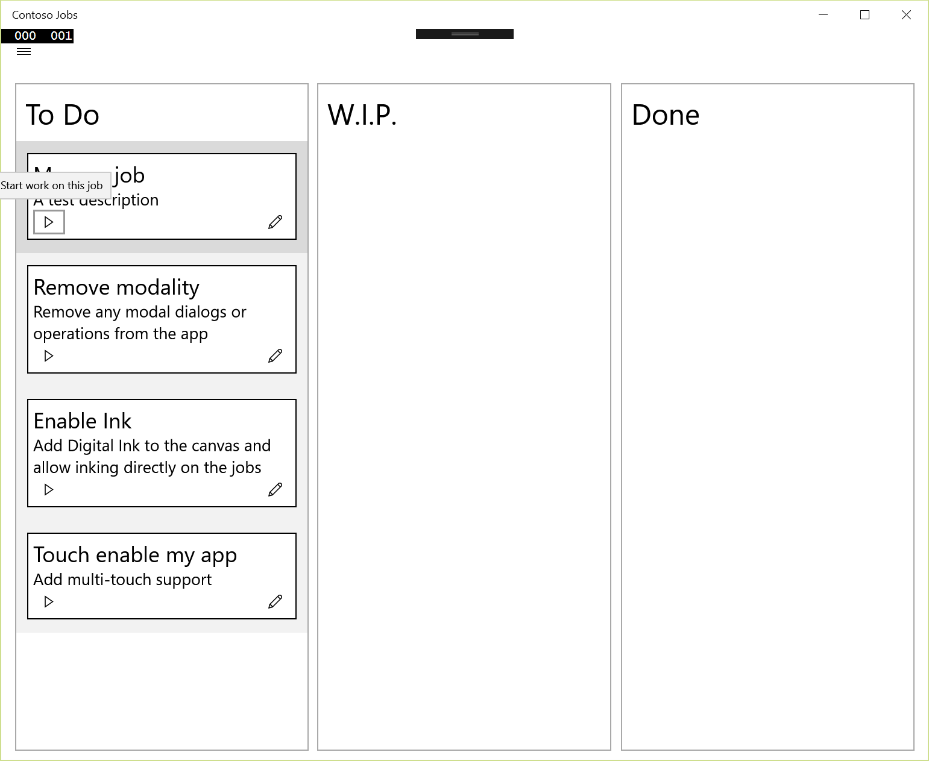


Figure Start a Job

1. On the right hand bottom corner of each Job is an edit button. This allows a Job to be edited. When you press this button, a modal dialog is displayed to allow the Job to be edited. This blocks any other user from working on the App at the same time. This will have to be changed.

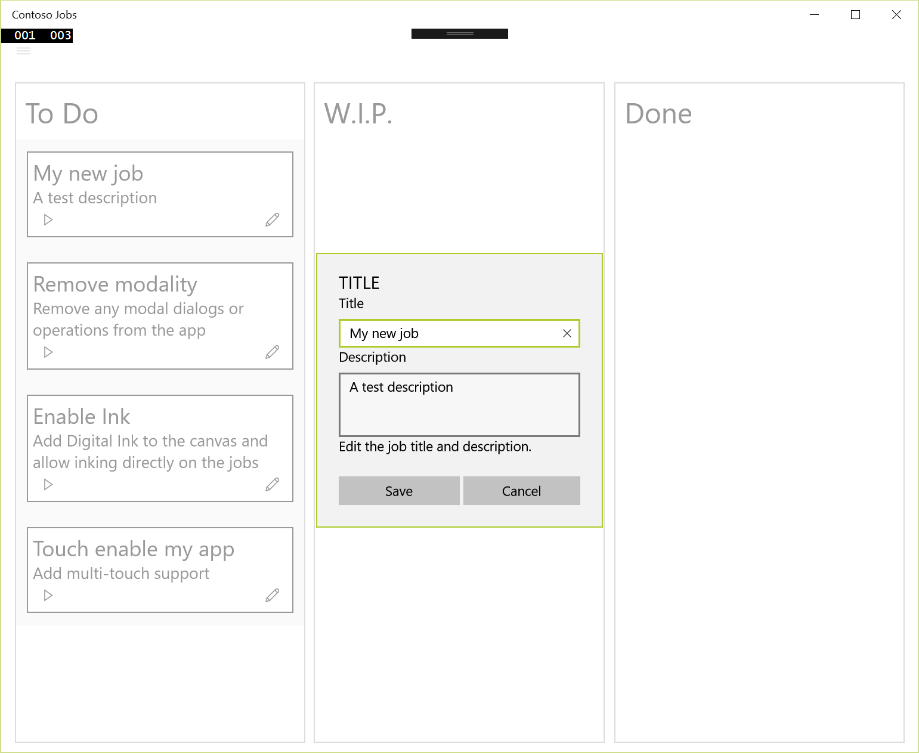


Figure Edit Job

1. When a Job is completed, the Stop button in the Job can be clicked and it will move to the Done list. Notice that again the Stop button has a tooltip when you hover over it, this will never be shown in a touch interface, as there is no concept of hover in touch.

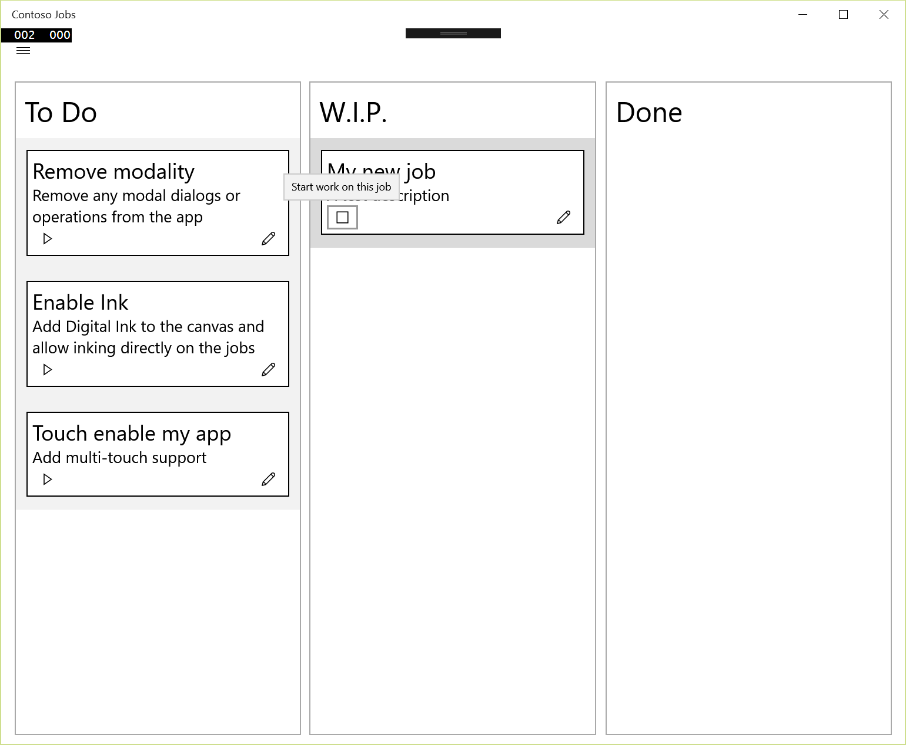


Figure Tooltip for Stop button

1. When you close the app and open it again, the Jobs you have created and their status will be reloaded. Open the Common folder in the Solution, and open the file called XmlService.cs.   
   This is how the Jobs are saved and loaded.

In the SaveJobs method you can see that the Jobs are saved in the local folder

StorageFolder storageFolder;

storageFolder = ApplicationData.Current.LocalFolder;

var jobFile = await storageFolder.CreateFileAsync(JobsFile,

CreationCollisionOption.ReplaceExisting);

using (var writeStream = await jobFile.OpenStreamForWriteAsync())

{

new XmlSerializer(jobs.GetType()).Serialize(writeStream, jobs);

}

On Surface Hub this will be an issue as this folder will not be maintained between sessions. We will need to change where Jobs are saved.

In this lab you have seen how the current app works and some of the areas that are going to need work.

In the following tutorials you will address these issues as well as adding new features to enhance this application and make it a better Surface Hub app.

**Deploying the project to a Surface Hub (optional):**

1. The Surface Hub should be on the same subnet as the development computer.
2. Ensure the Surface Hub is connected through an Ethernet connection as this will improve the speed of the deployment.
3. Ensure the Surface Hub is in Developer Mode.
   1. From the Surface Hub's **Start** menu, open the Settings app.

**Note** Administrative privileges are required to access the Settings app. Also, this is currently an irreversible change without a factory reset.

* 1. Navigate to **Update & security > For developers**.
  2. Choose **Developer mode** and accept the warning prompt.
  3. Restart the Surface Hub

1. Obtain the IP address of the Surface Hub from the upper right hand information box on the start screen.
2. On the deployment machine, right click on the project in the Solution Explorer window.
3. Go to the tab 'Debug'
4. Inside Debug, go to 'Start Options' and change the Target Device to 'Remote Machine' and enter the IP address of the Surface Hub.
5. The solution can now be deployed directly to the Surface Hub.