Integrating Software Parts for Project #2

1. Go to Lecture #4 page, click on Navigate Directory Trees, right click on Navigate.zip and select save target as selecting some directory (I'm choosing C:\temp\Integrate). Now right click on the zip file and select extract all, taking the default path.
2. Repeat this process with Template Inserter Demo and Parser Folder
3. Now start visual studio, click on top menu item File and select new and Project.
4. Now in the New Project dialog select Other Project Types -> Visual Studio Solutions, type Project2 for the name and set the location to the download path, e.g., in my case that will be C:\temp\Integrate.
5. Pull each of the folders: Navigate, Parser-Fall11, and VisualStudioDemo-Fall11 into the Project2 folder. I renamed VisualStudioDemo-Fall11 to TextInserter for convenience.
6. Now, right click on the solution at the top of the Solution Explorer view and select: add->existing project, browse to the c:\temp\Integrate\Navigate folder and select the Navigate project. That will copy the project and all its files into the Project2 folder.
7. Repeat for the Parser and TextInserter projects.
8. Select the Navigate project, right-click and select properties. Select the Application Tab and set the Output type to Class Library and build the project by right clicking on the project and selecting rebuild.
9. For convenience later let's change the Executive and Navigate namespaces to Project2 in the files Executive.cs and Navigate.cs.
10. Finally, add a new C# console application project called Executive.
11. Open the Navigate Test.cs in the Visual Studio editor by double clicking on it in Solution Explorer. Now select all the code inside its main function, copy (cntrl-C) to the clipboard and open the Executive and paste the copied code into its main function.
12. Add using System.IO; at the end of the using statements in the Executive.
13. Change the statement Navigate.go(path, "*.*"); to Navigate.go(path, "*.cs");
14. Right-click on the References tab in the Executive project and select add reference. In the dialog select projects-Navigate and click ok.
15. right click on the Executive project, select properties, click on the debug tab and insert ../../.. in the Command Line Arguments text box.
16. Build and Run the Executive by clicking on Debug->Start without Debugging.
17. Note that your executive has found all the C# files on the input path.

Now, let's change a few names to make things look more like the packages in your OCD.

1. First, let's rename Navigate to FileFinder, rename Parser to Locator, but leave TextInserter as is.
2. Instead of displaying the files it finds, we want FileFinder to return a list of files to publish. To do that we will make a few changes to Navigator.go(). First remove all the Console.write statements. Let's add a List<string> files member to hold the files as Navigator finds them.
3. In order to access this member data we have to remove the static qualifier from go()'s return type, which means we will have to create an instance of Navigator in our Executive.

4. Whenever go encounters files it saves them in the List<string> files member using its addRange member.
5. Now we have to add a function for the executive to retrieve the files. We'll call it getFiles(), and have it return a List<string>.
6. After we make these changes let's simply let the Executive display the found files.

All the remaining steps I'll leave for you to carry out, but let's summarize them:

1. Let's create a Publish package that will use TextInserter to make a webpage. First, in the TextInserter Project let's use the add->new item->class selection and name the package Publish.cs.
2. Add to our Executive file loop a call to Publishe's makePage function (I just made that up). Use TextInserter to implement most of the makePage functionality.
3. Now use locator to find the important parts of the newly created page and pass its table to MakeClickable to insert spans at the appropriate places.
4. Add an output package to build the directory page.
5. Test as you go and test more here.