## Tutorial: Importing an existing project

The following tutorial takes you though the process of importing a 'legacy' application using the CDT.

Step 1: You want to create a single project that will reflect all of the components for the existing source tree.

1. Select **File > New > Project**.
2. For this tutorial, expand **C/C++** and select **C++ project**.  
     
   The **New Project** wizard displays. Click here to see an illustration (displayed in a separate window).
3. In the **Project name** field, type a name for the project.
4. Ensure that the **Use default location** option is not selected because here we will specify where the resources reside in the file system, in a location other than your workspace.  
   We assume this directory location already exists, and contains e.g. source files and makefile. If it does not exist, it will be created.  
   In this example, we will use an existing hello.cpp and its existing hello.mak makefile, located in the directory c:\brandon.  
     Click here to see an illustration (displayed in a separate window).
5. In the **Location** field, specify the path to the existing files for your project.
6. From the **Project types** list, expand **Makefile Project** and select **Empty Project.**.
   * To have sample source and a makefile created in your existing directory, select **Hello World C++ Project**
7. Make sure a toolchain is selected.
8. Click **Next**.
9. (Optional) On the **Select Configurations** page, select only the configurations you want created. Configurations display for each toolchain selected on the previous page of the wizard.
10. Click **Finish** to close the dialog box.  
    You will see the new project in the Project Explorer view. In addition, new 'dot' files have been created in your legacy project's root directory, these are CDT project files.
    * (Optional) To see these "hidden" files, open the view menu on the Project explorer view, and select "Customize View."  
      The view menu is activated via the small downward-pointing triangle in the upper right of the toolbar of the view.  
      Click here to see an illustration (displayed in a separate window).
    * In the **Available Customizations** window, on the **Filters** tab, uncheck ".\* resources."  
      Click here to see an illustration (displayed in a separate window).
    * The formerly hidden 'dot' files are now shown in the **Project Explorer**.  
      Click here to see an illustration (displayed in a separate window).
    * Hit **OK** to close the dialog.

Step 2: You are now ready to build your project.

1. To build your project, select **Project** > **Build Project**.  
   You can also hit the hammer icon  on the toolbar to build the selected project.
2. You may see an error e.g. "make: \*\*\* no rule to make target 'all'.  
   This is because, in our case, we didn't tell it where the makefile is, or its name.
   * Open the project properties (right mouse on project name in the **Project Explorer** view and select **Properties** at the bottom on the context menu).
   * On the C/C++ Build Page, on its **Builder Settings** tab, uncheck "Use default build command" and change the make command to make -f hello.mak since our makefile is not named the default makefile.  
     Click here to see an illustration (displayed in a separate window).
   * Click **OK** to close the **Project Properties** window.
   * Your project will probably rebuild. If not you can force a build via **Project** > **Build Project** or use the hammer icon .

After the project build completes, the results display in the **Console** view and new objects, such as binaries and includes, show in the **Project Explorer** view.  Click here to see an illustration (displayed in a separate window).

1. Review the results to ensure that there are no errors.  
      
   **Note:** By default, the indexer is set to **Fast indexer** for the project to parse your project in the same way that a compiler does; beginning with each compilation unit and parsing that file and all files that it includes, except that it parses each header file included in a project only once. This method provides the most accurate index information. For large projects using complex C++ code, this indexer can be slow. For example, if a header file is included and takes two compilation units, the parsing of the second unit reuses the results of parsing the first unit. This is similar to how precompiled headers work. The indexing of large projects using the Fast indexer uses fewer resources than the Full indexer, but the resulting index is not quite as accurate.  
   To change indexer settings, bring up the Indexer page in the Project Properties.  
   Click here to see an illustration (displayed in a separate window).

Step 3: You are now ready to run your application.

1. To run your application within the C/C++ Perspective, click **Run > Run Configurations...**  
     
   The **Run Configurations** dialog displays.
2. To create a run configuration, select **C/C++ Application** and click the **New** icon.
3. Give your run configuration a name by entering a new name in the **Name** field at the top.
4. On the **Main** tab, select the project by clicking **Browse** opposite the **Project** field.
   * From the Project Selection window, select the project and click **OK**.
5. Select the new **C/C++ Application** you want to execute by clicking **Search**.  
   In our case it didn't find the file because it's in a nonstandard location. Use the **Browse** button to select the executable. In our case also it's a nonstandard name, hello.e because that's what our makefile hello.mak built.  
   Click here to see an illustration (displayed in a separate window).
   * If you see the error "[Debugger]: No debugger available", select the Debugger tab and choose a valid debugger for your platform (e.g. gdb/mi).
6. Click **Run** to save the run configuration and run it.  
   Click here to see an illustration (displayed in a separate window).  
   The application runs in the **Console** view. The **Console** also shows which application it is currently running in the title bar. You can configure the view to display different elements, such as user input elements using different colors.

