Smart TV SDK PNaCl IDE Tutorial

Published 2014-10-28 | (Compatible with SDK 5.0,5.1 and 2014 models)

> This tutorial will show you how to use Smart TV SDK PNaCl IDE to create / setting / build / run & debug PNaCl projects.

Contents

Prerequisites

To use Smart TV PNaCl IDE, you'll need the JDK / Eclipse & CDT / Python / PNaCl ToolChain and the Google Chrome browser, and you also need to set some environment variables.

Getting the JDK

Download and install the appropriate JDK package for your platform from **http://www.java.com**.

Getting the Eclipse & CDT

The PNaCl IDE has already included the Eclipse & CDT, if you want to get them by yourself, you can get them from **http://www.eclipse.org**.

Install Python

The PNaCl ToolChain depends on the Python 2.6 or 2.7, and needs to add the Python executable path to the OS's environment variable (PATH).

On Mac/Linux, Python is probably preinstalled. Run the command "python -V" in a terminal window, and make sure that the version of Python you have is 2.6.x or 2.7.x (if it's not, upgrade to one of those versions).

On Windows, you may need to install Python. Go to **http://www.python.org/download** and select the latest 2.x version. In addition, be sure to add the Python directory (for example, C:\python27) to the PATH environment variable. After you've installed Python, run the command "python V" in a Command Prompt window and verify that the version of Python you have is 2.6.x or 2.7.x.

Note that Python 3.x is not yet supported.

PS: On the Mac, you need to install the make command on your system before you can build and run the examples.

Getting the PNaCl ToolChain

Download the SDK update utility tool (nacl_sdk.zip) from **https://developers.google.com/native-client/dev**. Then you can use the tool to install the PNaCl ToolChain :

Unzip the SDK update utility

On Mac/Linux, run the command "unzip nacl_sdk.zip" in a Terminal window, Go to the nacl_sdk directory and run naclsdk command with the "list" parameter to see a list of available bundles. The SDK includes a separate bundle for each version of Chrome/Pepper, currently, pepper_31 is recommended.

```
$ cd nacl_sdk
```

```
$ ./naclsdk list
```

Run naclsdk with the "update" parameter to download particular bundles that are available.

```
$ ./naclsdk update pepper_31
```

On Windows, right-click on the .zip file and select "Extract All…". A dialog box will open; enter a location and click "Extract".Go to the nacl_sdk directory and run naclsdk with the "list" parameter to see a list of available bundles. The SDK includes a separate bundle for each version of Chrome/Pepper, currently, pepper_31 is recommended.

```
> cd nacl_sdk
```

```
> naclsdk list
```

Run naclsdk with the "update" parameter to download particular bundles that are available.

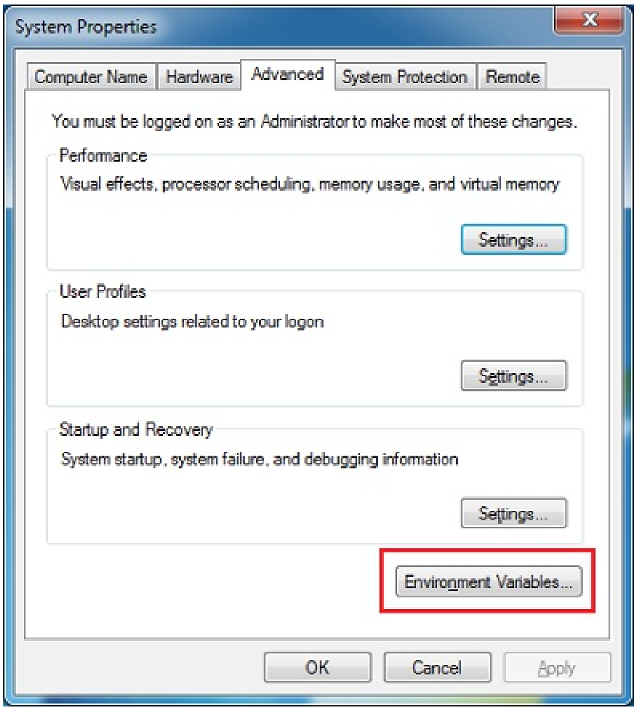
```
> naclsdk update pepper_31
```

Check Pepper C API to find out more about toolchains used on Samsung devices.

After install the ToolChain, we need to set the environment variables :

On windows

1. Set **NACL_SDK_ROOT** ('location-where-you-installed-the-SDK'/pepper_31) in User variables
2. Add **NACL_SDK_ROOT/tools** to Path
3. Add **NACL_SDK_ROOT/toolchain/win_pnacl/bin** to Path



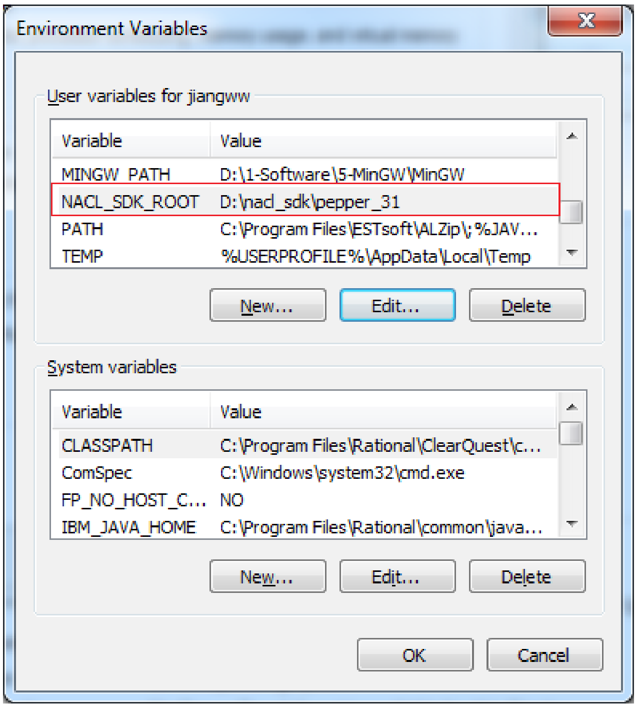
You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your logon

System startup, system failure, and debugging information

Figure : Set Windows Environment Variables



Variable	Value
MINGW_PATH	D:\1-Software\5-MinGW\MinGW
NACL_SDK_ROOT	D:\nacl_sdk\pepper_31
PATH	C:\Program Files\ESTsoft\ALZip\;%JAV...
TEMP	%USERPROFILE%\AppData\Local\Temp

Variable	Value
CLASSPATH	C:\Program Files\Rational\ClearQuest\c...
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
IBM_JAVA_HOME	C:\Program Files\Rational\common\java...

Figure : Set Windows Environment Variables

On Linux

1. Add **NACL_SDK_ROOT ('location-where-you-installed-the-SDK'/pepper_31)** to PATH and export NACL_SDK_ROOT
2. Add **NACL_SDK_ROOT/tools** to Path
3. Add **NACL_SDK_ROOT/toolchain/linux_pnacl/bin** to Path

On Mac

1. Create file: /etc/launchd.conf
2. Get $PATH value from terminal, here using Path_Value
3. Input setenv NACL_SDK_ROOT 'location-where-you-installed-the-SDK'/pepper_31
4. Input setenv PATH location-where-you-installed-the-SDK>/pepper_31/toolchain/mac_pnacl/bin:Path_Value
5. Save and close /etc/launchd.conf, then restart Mac
   PS: If there is no make command, in /etc/launchd.conf add setenv PATH 'location-where-you-installed-the-SDK'/pepper_31/tools:'location-where-you-installed-the-SDK'/pepper_31/toolchain/mac_pnacl/bin:Path_Value.

Install and Configure the Google Chrome

This is optional, but it could help to run and debug the native client module. And in order to make it work, the Chrome version must be equal to or greater than the PNaCl ToolChain version. For example, if the pepper_31 is used, the Chrome version must be 31 or greater.

1. Enable the Native Client flag in Chrome. (Native Client is enabled by default for applications distributed through the Chrome Web Store. To run Native Client applications that are not distributed through the Chrome Web Store, e.g., applications that you build and run locally, you must specifically enable the Native Client flag in Chrome.)
   Type about : flags in the Chrome address bar and scroll down to **Native Client**.
   If the link below **Native Client** says **Disable**, then Native Client is already enabled and you don't need to do anything else.
   If the link below **Native Client** says **Enable**, click the **Enable** link, scroll down to the bottom of the page, and click the **Relaunch Now** button. All browser windows will restart when you relaunch Chrome.

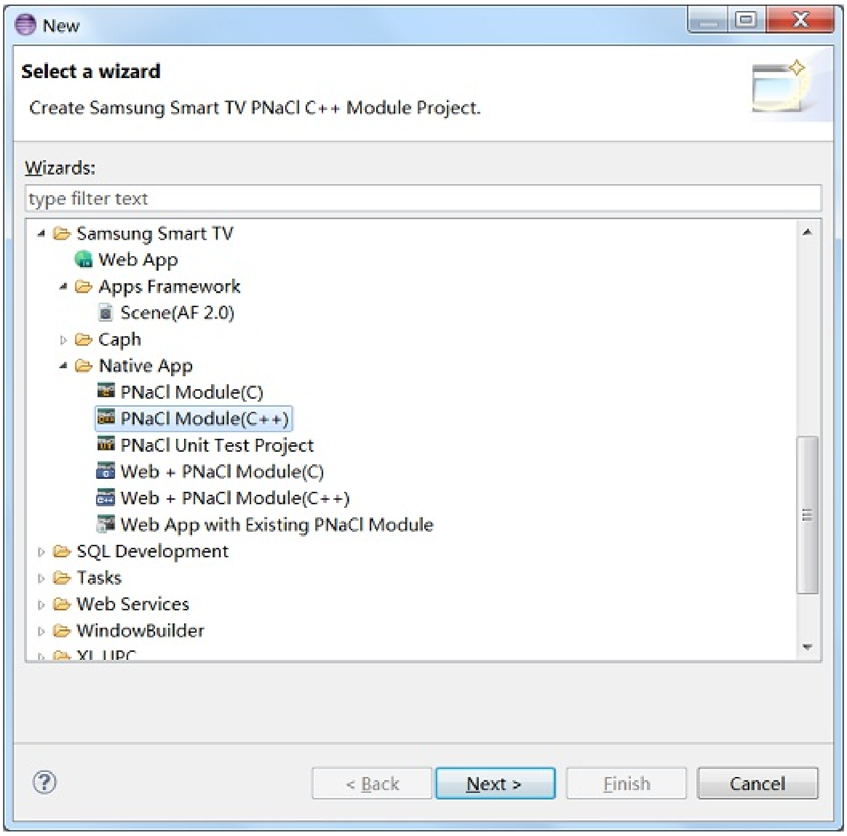
2. Disable the Chrome cache. (Chrome caches resources aggressively; you should disable the cache whenever you are developing a Native Client application in order to make sure Chrome loads new versions of your application.)
   Open Chrome's developer tools by clicking the menu icon ≡ and choosing **Tools -> Developer tools**.

Click the gear icon ⚙ in the bottom right corner of the Chrome window.

Under the **General** settings, check the box next to **Disable cache**.

Create Samsung Smart TV PNaCl App Projects

There are 6 different entrances for PNaCl project creation, shown as the following figure :



Select a wizard

Create Samsung Smart TV PNaCl C++ Module Project.

Figure : PNaCl Project Wizard

But actually, there are only 2 types of PNaCl projects: the PNaCl Module project, including the first 2 entrances; and the Web + PNaCl Module project, including the rest 3.

The PNaCl Module project means that this wizard just guides to create the PNaCl module without Web part (HTML, JS, CSS, .etc). The PNaCl module can be developed by both C and C++, as the toolchain might be different for these 2 languages, 2 different entrances are created for C and C++ respectively.

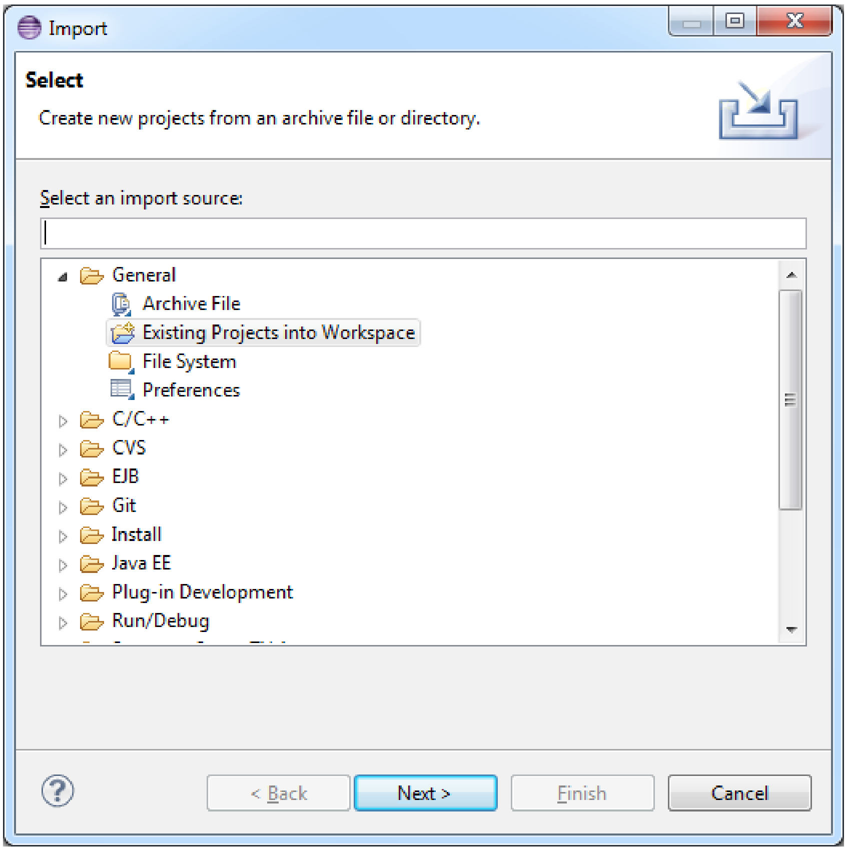
The Web + PNaCl project means that this wizard will guide to create both the Web part (HTML, JS, CSS, .etc) and PNaCl module. Same reason as above, 2 different entrances for 2 different languages are provided. Unlike others, the 'Import PNaCl Module into Web' wizard guide to import an existing PNaCl module into a web project. The PNaCl module might be developed with the above 2 wizards, and there's no need for developers to do any C/C++ coding jobs for this kind of project.

This user guide will create a PNaCl Module(C++) and a Web + PNaCl Module(C++) for example, after building the PNaCl Module(C++) project, the final output (the PNaCl Module) with .pexe extension could be used to be imported into a Web part using the last wizard introduced above.

As the creation process for PNaCl(C) projects are very similar with the PNaCl(C++) projects, it will be omitted in this guide.

The examples are all based on the template. The only difference between that with template and without template is that the wizard with template will provide a source file of the PNaCl module, and some commonly used APIs will be listed in the file.

Of course the SDK supports importing "Existing Projects into Workspace".



Select

Create new projects from an archive file or directory.

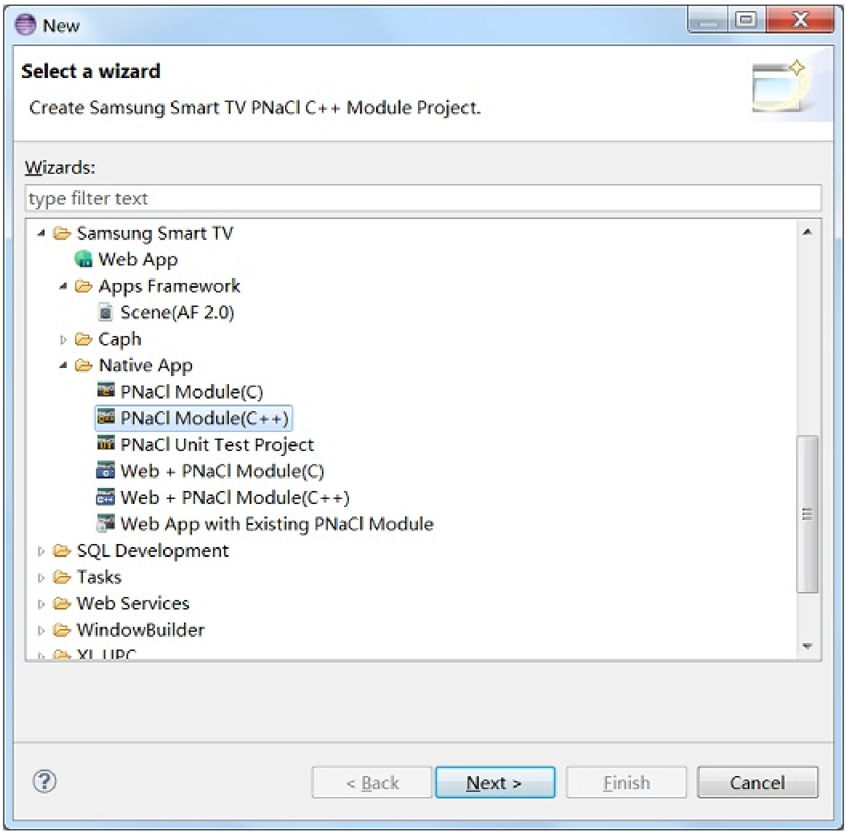
Figure : Import Existing Projects into Workspace

But Notice: SDK doesn't support perspective automatically switch to 'C/C++' perspective. Developers have to click C/C++ to do that to get the C/C++ like projects tools.

Create PNaCl Module(C/C++)

> Note
>
> A PNaCl Module(C) project's creation processes are very similar with that of PNaCl(C++), here only the PNaCl(C++) project will be discussed.

1. Open the new wizard dialogue, find the 'Samsung Smart TV Apps \ PNaCl Apps' folder, expand it and select the **PNaCl Module(C++)** item, then press **Next**(or double click the item) :

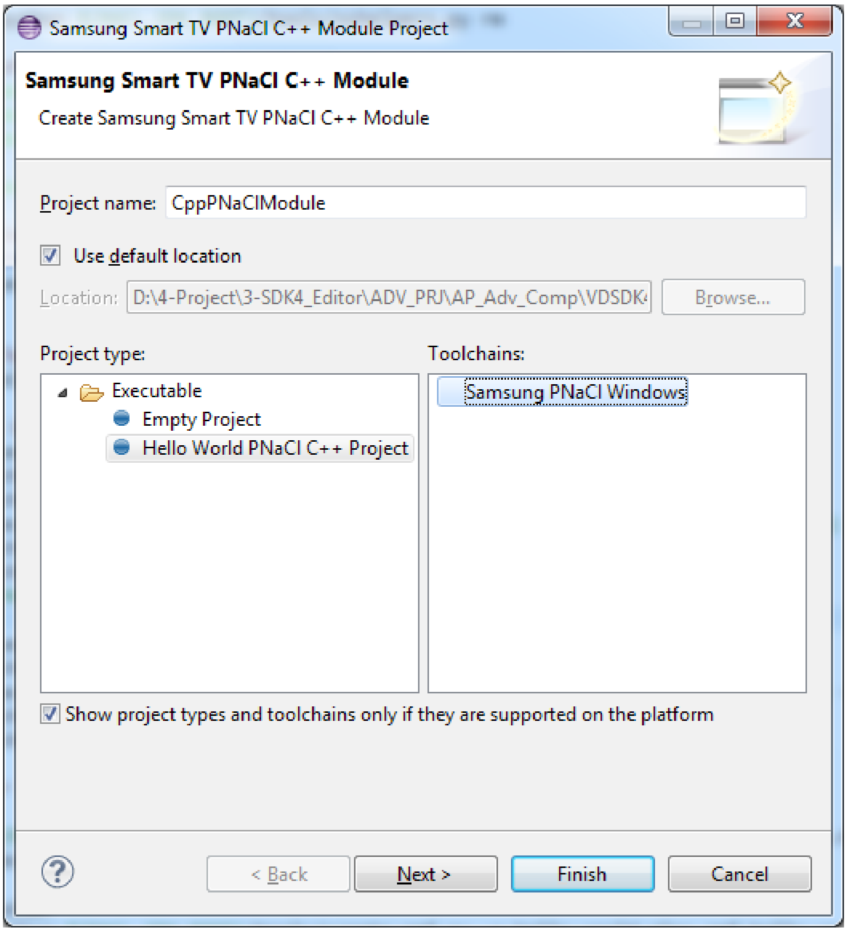


Select a wizard

Create Samsung Smart TV PNaCl C++ Module Project.

Figure : Select the Wizard

2. Fill in the **Project name**, and in the **Project Type** section, as mentioned above, the **Hello World PNaCl C++ Project** is selected; and in the **Toolchains** section, only the supported toolchain is displayed.
   Developers can press **Finish** button to complete the creation directly, or they can press **Next** to configure the configuration for this project.



Samsung Smart TV PNaCl C++ Module

Create Samsung Smart TV PNaCl C++ Module

Project name: CppPNaClModule

☑ Use default location

Location: D:\4-Project\3-SDK4_Editor\ADV_PRJ\AP_Adv_Comp\VDSDK4

Project type: Hello World PNaCl C++ Project

Toolchains: Samsung PNaCl Windows

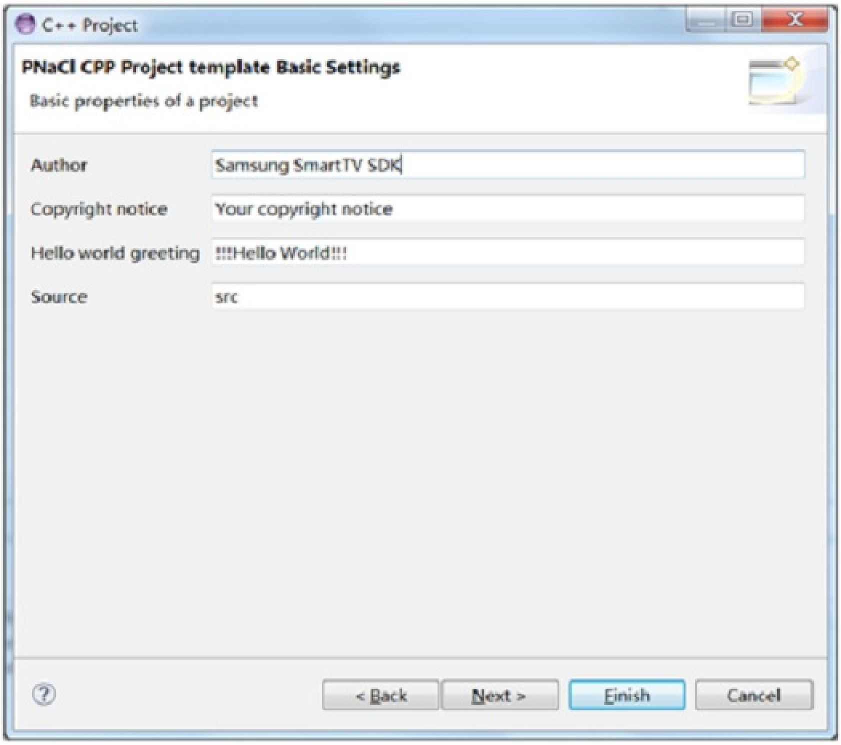
☑ Show project types and toolchains only if they are supported on the platform

Figure : Do the PNaCl Module Settings

3. If developers press **Next** instead of **Finish**, the following dialogue will be displayed. Developers can input their own

information here and they would be last memorized by the Eclipse System.

But notice: this dialogue is only for template project, and in this dialogue, developers can also choose to 'Next' or 'Finish' directly.



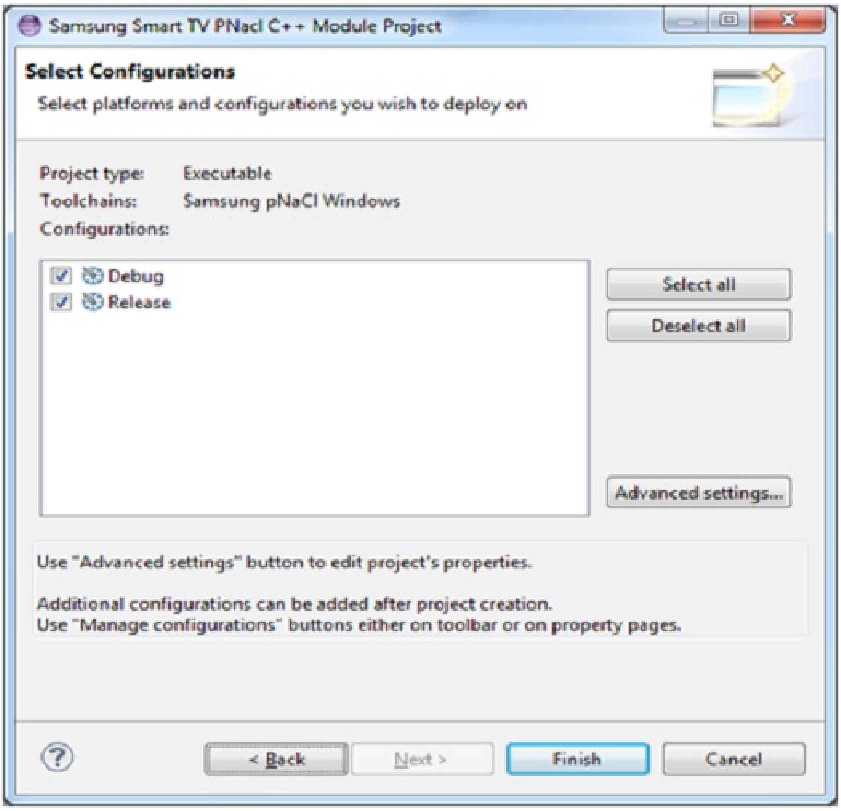
PNaCl CPP Project template Basic Settings

Basic properties of a project

Author: Samsung SmartTV SDK
Copyright notice: Your copyright notice
Hello world greeting: !!!Hello World!!!
Source: src

Figure : Input the Template Project Information

4. If developers press **Next** instead of **Finish**, the following dialogue will be displayed and by default, the **Debug** and **Release** configurations are both supported, which means developers can build both the **Debug** version and **Release** version of the project according to which is set to be active by developers. If either **Debug** or **Release** is selected, it means only the selected configuration is supported and developers are not able to change it anymore.



Select Configurations

Select platforms and configurations you wish to deploy on

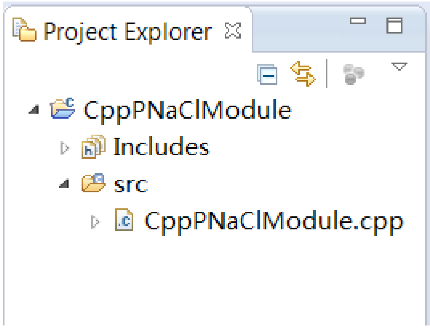
Project type: Executable
Toolchains: Samsung pNaCl Windows
Configurations:
☑ Debug
☑ Release

Use "Advanced settings" button to edit project's properties.

Additional configurations can be added after project creation.
Use "Manage configurations" buttons either on toolbar or on property pages.

Figure : Configure the Build Type

5. After pressing **Finish**, a PNaCl Module (C++) Project with template has been created in the current workspace. The

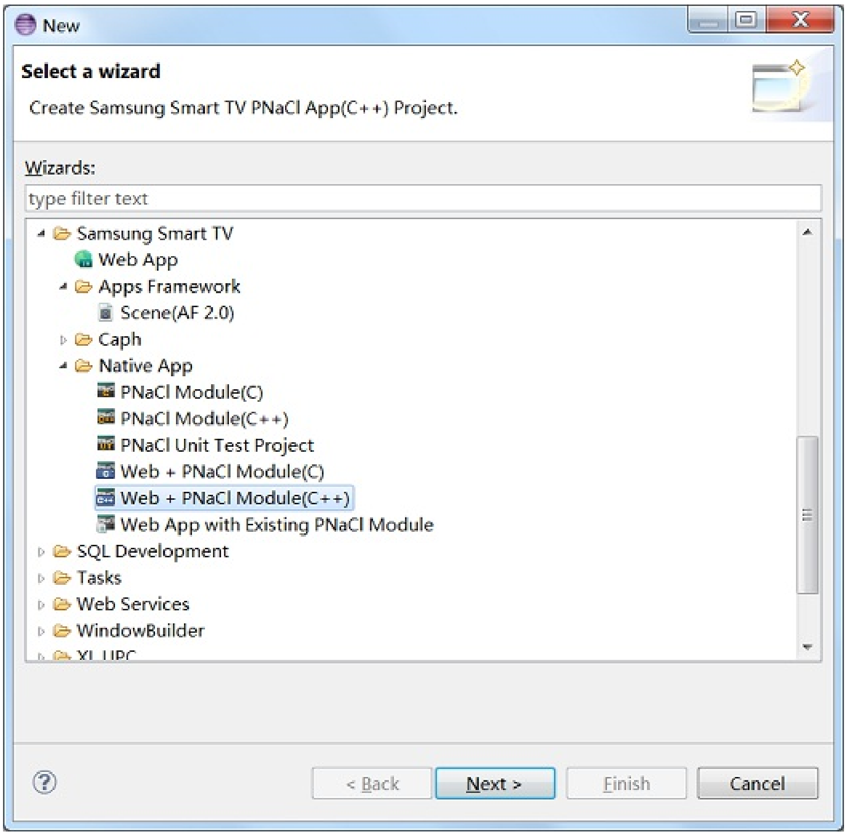


following figure displays the project architecture. **Figure** : PNaCl Module(C++) Project's Architecture

Create Web+PNaCl Module(C/C++)

> Note
>
> A Web + PNaCl Module(C) project's creation processes are very similar with that of Web + PNaCl(C++), here only the Web + PNaCl(C++) project will be discussed.

1. Open the new wizard dialogue, find the **Samsung Smart TV Apps \ PNaCl Apps** folder, expand it and select the **Web + PNaCl Module(C++)** item, then press **Next**(or double click the item) :

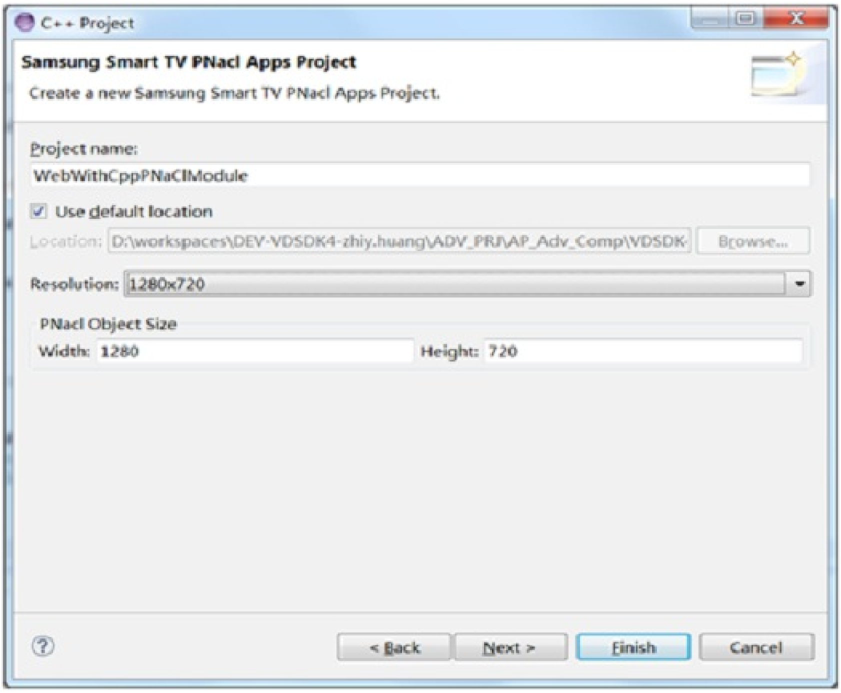
   

   Select a wizard

   Create Samsung Smart TV PNaCl App(C++) Project.

   Figure : Select the Wizard

2. The first wizard page is for the Web part. Developers should input the Project Name, set the location (by default, the current workspace is set), choose the web part's resolution and the PNaCl Object size. The PNaCl object's size can never exceed the web part's resolution.
   Developers can press **Finish** button to complete the creation directly, or they can press **Next** to configure the configuration for this project.


C++ Project

Samsung Smart TV PNacl Apps Project

Create a new Samsung Smart TV PNacl Apps Project.

Project name: WebWithCppPNaClModule

☑ Use default location

Location: D:\workspaces\DEV-VDSDK4-zhiy.huang\ADV_PRJ\AP_Adv_Comp\VDSDK

Resolution: 1280x720

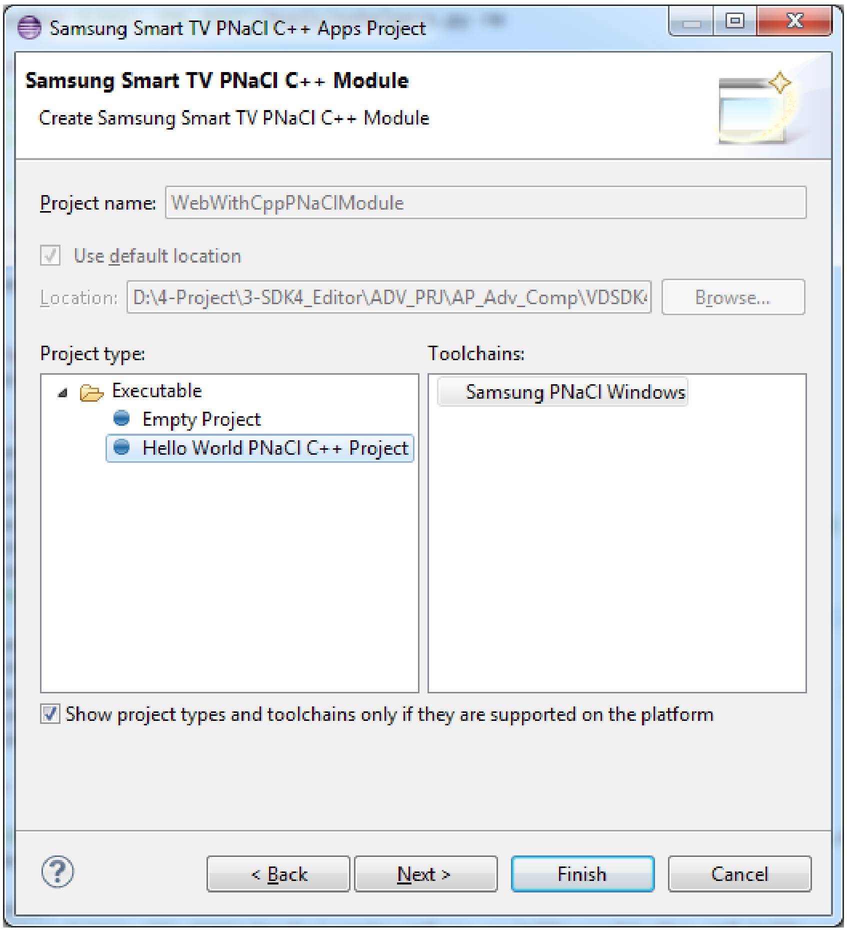
PNacl Object Size

Width: 1280

Height: 720

Figure : Do the Web Settings

3. If developers choose **Next** instead of **Finish**, they will go to the next wizard page which enters the PNaCl Module part. The Project Name which has been set in the previous page cannot be modified here, and the rest part of this page is the same as above, and it will be skipped.


Samsung Smart TV PNaCl C++ Apps Project

Samsung Smart TV PNaCl C++ Module

Create Samsung Smart TV PNaCl C++ Module

Project name: WebWithCppPNaClModule

☑ Use default location

Location: D:\4-Project\3-SDK4_Editor\ADV_PRJ\AP_Adv_Comp\VDSDK

Project type:

- Executable
  - Empty Project
  - Hello World PNaCl C++ Project

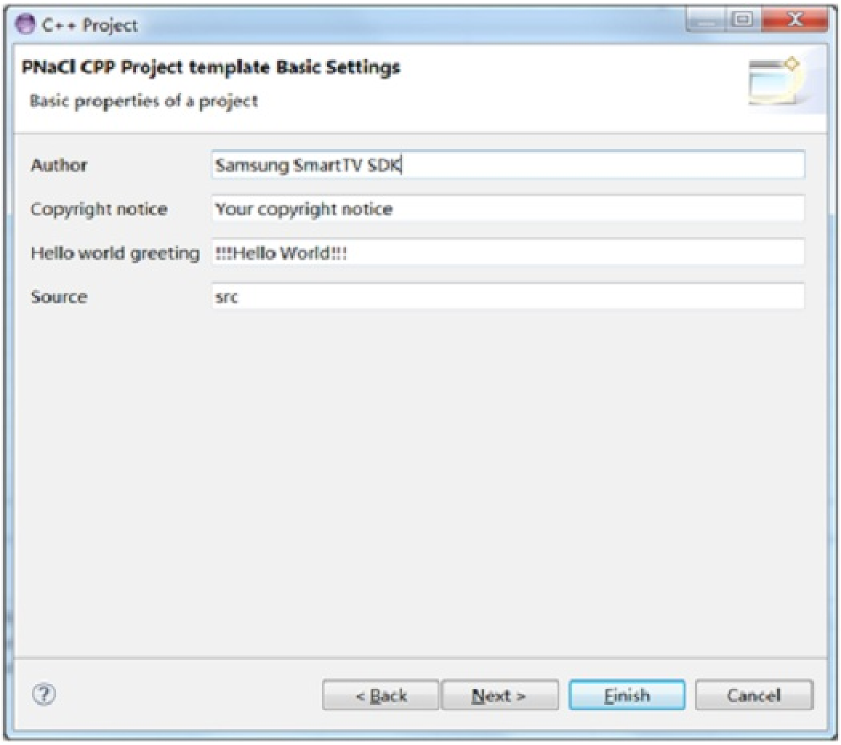
Toolchains:

Samsung PNaCl Windows

☑ Show project types and toolchains only if they are supported on the platform

Figure : Do the PNaCl Module Settings

4. If developers press **Next** instead of **Finish**, the following dialogue will be displayed. Developers can input their own information here and they would be last memorized by the Eclipse System.
   But notice: this dialogue is only for template project, and in this dialogue, developers can also choose to **Next** or **Finish** directly.


C++ Project
PNaCl CPP Project template Basic Settings
Basic properties of a project
?
< Back
Next >
Finish
Cancel

Author: Samsung SmartTV SDK

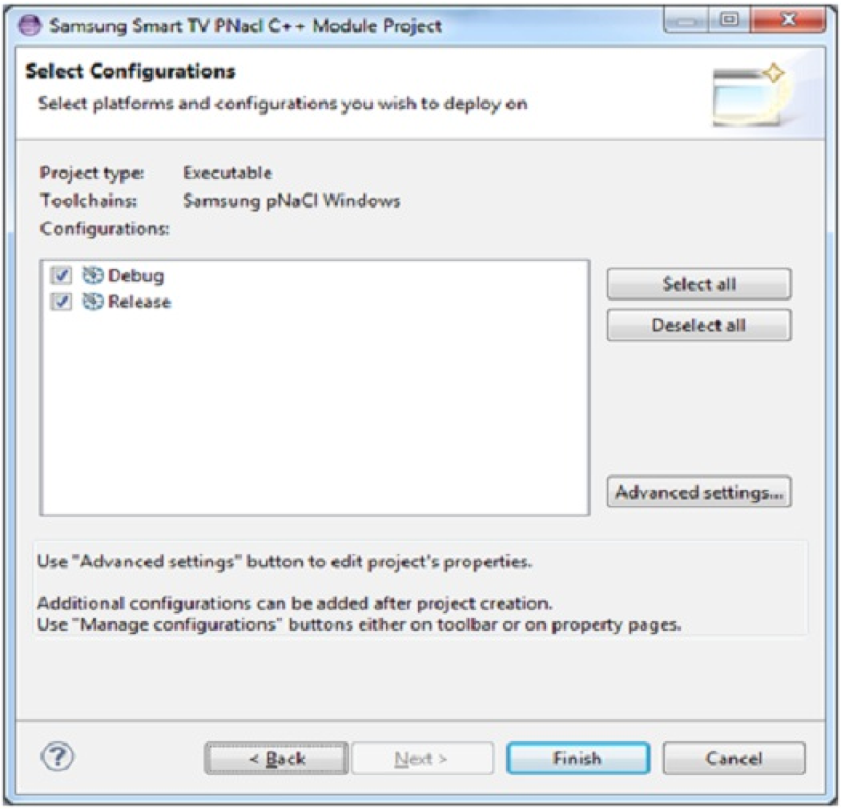
Copyright notice: Your copyright notice

Hello world greeting: !!!Hello World!!!

Source: src

Figure : Input the Template Project Information

5. If developers press **Next** instead of **Finish**, the following dialogue will be displayed and by default, the 'Debug' and 'Release' configurations are both supported, which means developers can build both the 'Debug' version and 'Release' version of the project according to which is set to be active by developers. If either 'Debug' or 'Release' is selected, it means only the selected configuration is supported and developers are not able to change it anymore.


Samsung Smart TV PNacl C++ Module Project
Select Configurations
Select platforms and configurations you wish to deploy on
Select all
Deselect all
Advanced settings...
?
< Back
Next >
Finish
Cancel

Project type: Executable

Toolchains: Samsung pNaCl Windows

Configurations:

☑ Debug

☑ Release

Use "Advanced settings" button to edit project's properties.

Additional configurations can be added after project creation.
Use "Manage configurations" buttons either on toolbar or on property pages.

Figure : Configure the Build Type

6. After pressing **Finish**, a Web + PNaCl Module (C++) Project with template has been created in the current workspace. Compared to the PNaCl Module (C++), more files are created, such as the HTML, Javascript, CSS and other configuration files.

   The following figure displays the project architecture.

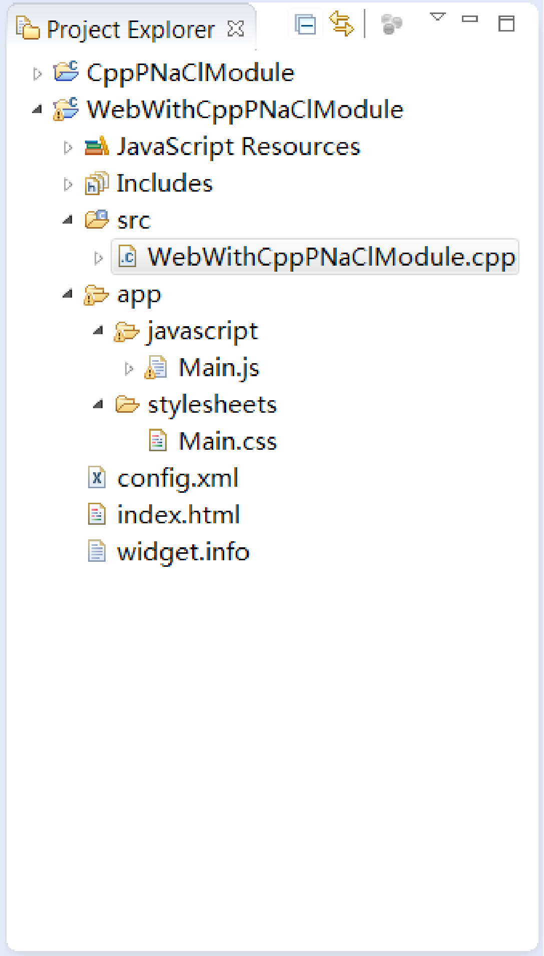
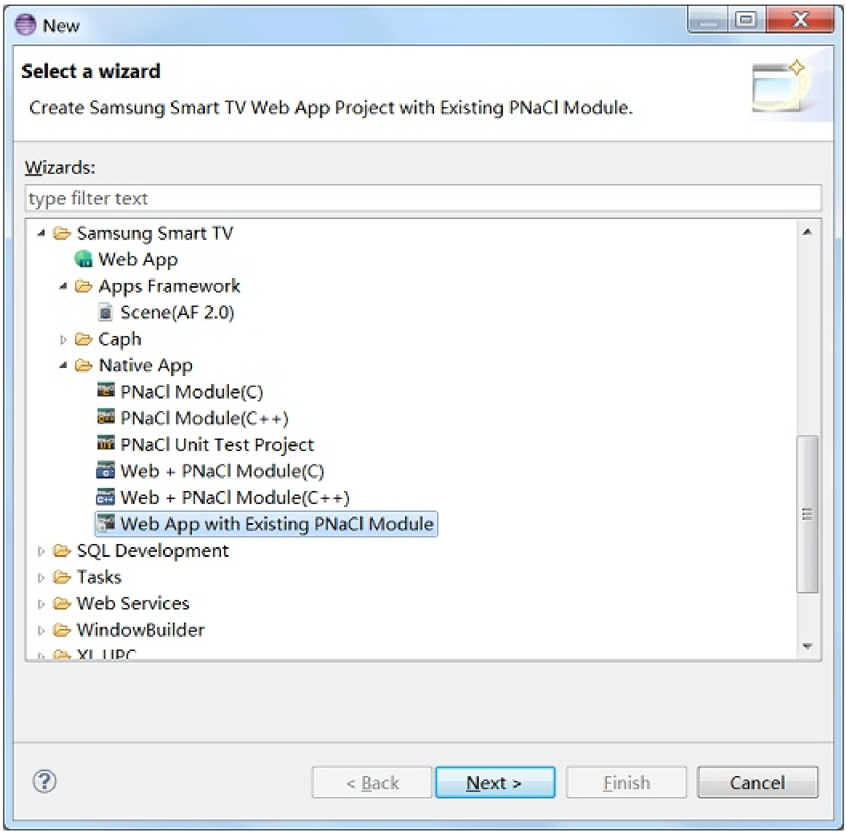


Figure : Web + PNaCl Module(C++) Project's Architecture

Create Web App with Existing PNaCl Module

After building the PNaCl Module project, the final output as with .pexe extension could be obtained, which could be used to import into Web.

1. Open the new wizard dialogue, find the **Samsung Smart TV Apps \ PNaCl Apps** folder, expand it and select the **Web App with Existing PNaCl Module** item, then press **Next**(or double click the item) :

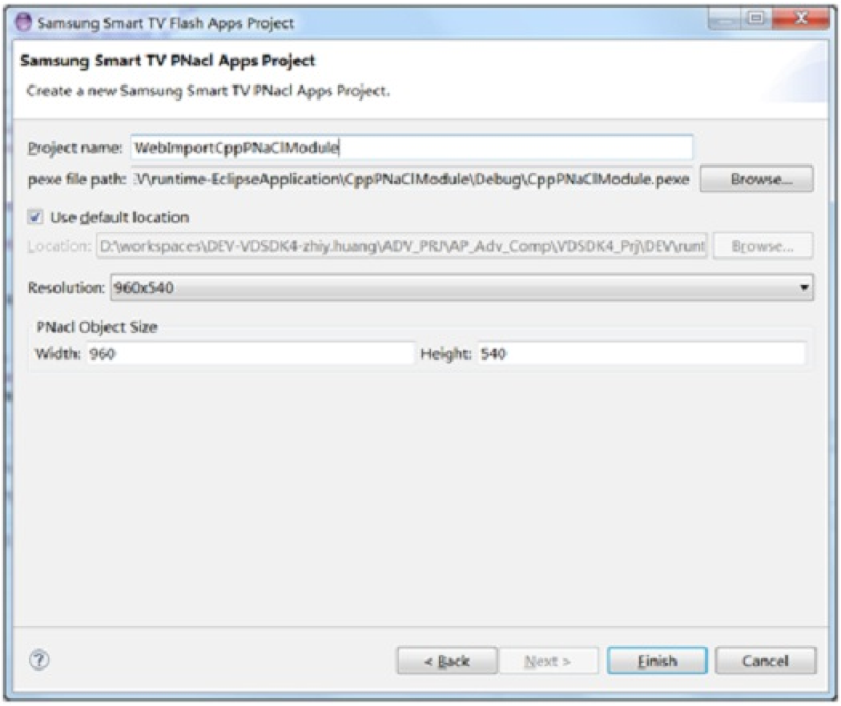


Select a wizard

Create Samsung Smart TV Web App Project with Existing PNaCl Module.

Figure : Select the Wizard

2. Fill in the **Project name**, and browse to select the PNaCl Module (.pexe file) to be imported into the project. After selecting the resolution of the project and confirm the PNaCl object size, developers can press **Finish** to complete the project



Samsung Smart TV PNacl Apps Project

Create a new Samsung Smart TV PNacl Apps Project.

Project name: WebImportCppPNaClModule

pexe file path: :\V\runtime-EclipseApplication\CppPNaClModule\Debug\CppPNaClModule.pexe

☑ Use default location

Location: D:\workspaces\DEV-VDSDK4-zhiy.huang\ADV_PRJ\AP_Adv_Comp\VDSDK4_Prj\DEV\runt

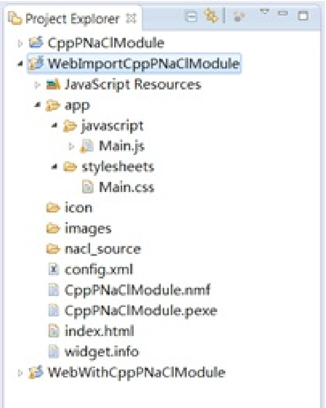
Resolution: 960x540

PNacl Object Size

Width: 960 Height: 540

creation : **Figure** : Do the Web Settings

3. The Project is almost the same as the Web + PNaCl Module project, except for the source files of the PNaCl Module are

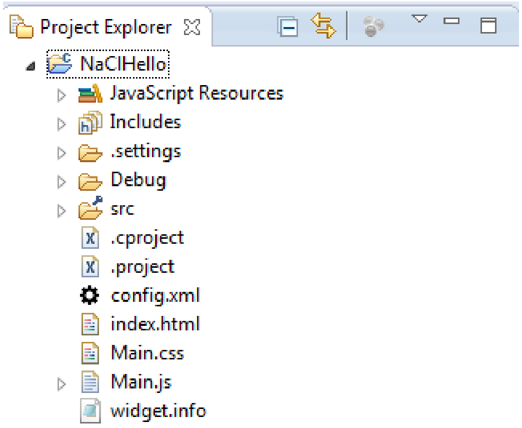


not used : **Figure** : Project Architecture

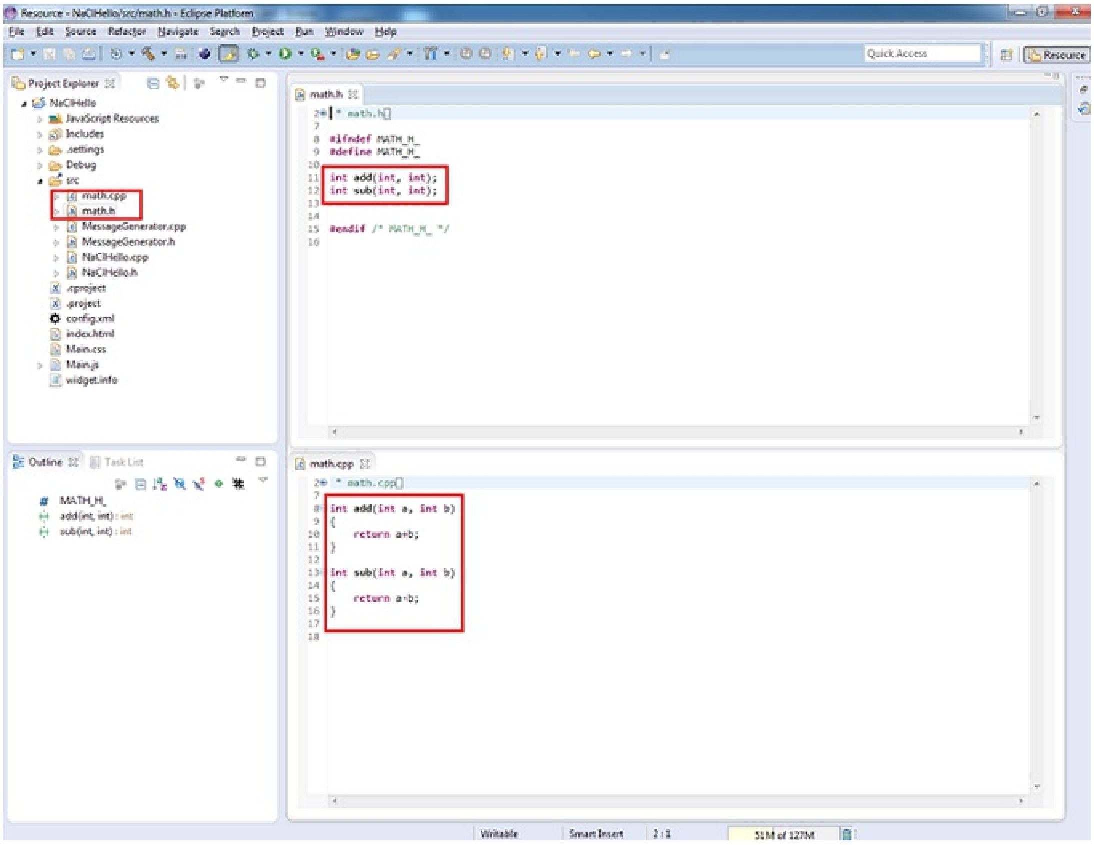
Create PNaCl Unit Test Project

The PNaCl tool chain has been upgraded to pepper_31, and one of the new features is the Unit Test based on Google C++ Testing Framework.

The following sections show how developers could do Unit Test for normal PNaCl projects.

1. 

   Create a PNaCl project which needs to be unit tested : **Figure** : Create PNaCl Project
2. Choose the File/Class/Function to be unit tested :



```
 * math.h

#ifndef MATH_H_
#define MATH_H_

int add(int, int);
int sub(int, int);


#endif /* MATH_H_ */
```

```
 * math.cpp

int add(int a, int b)
{
    return a+b;
}

int sub(int a, int b)
{
    return a-b;
}
```

Figure : Choose Test File/Class/Function

3. Create a PNaCl Unit Test Project using the wizard :

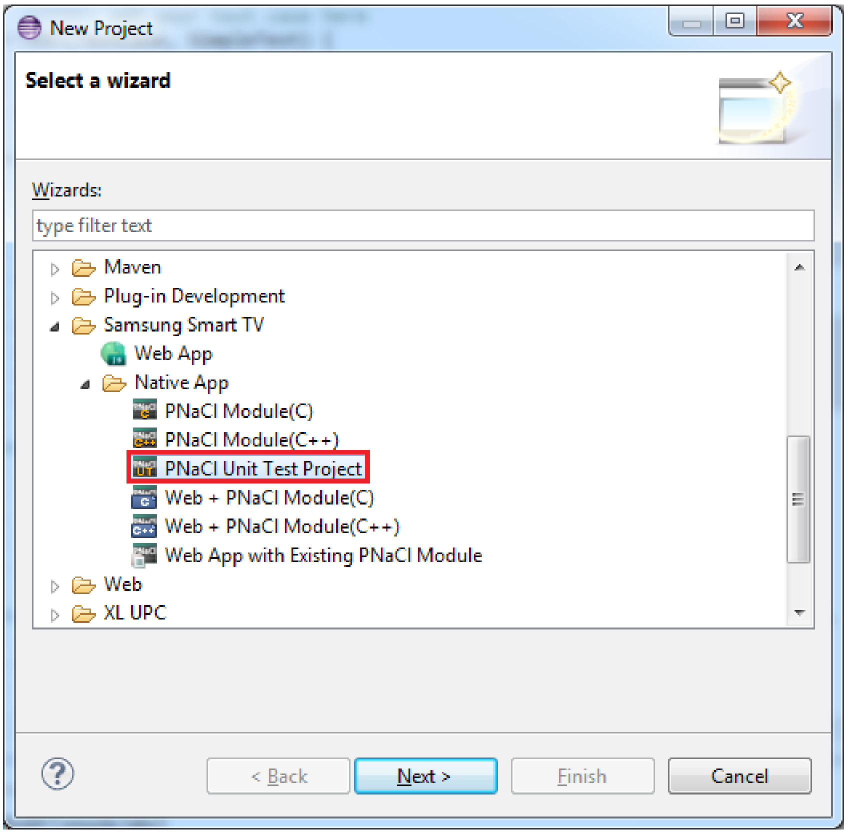
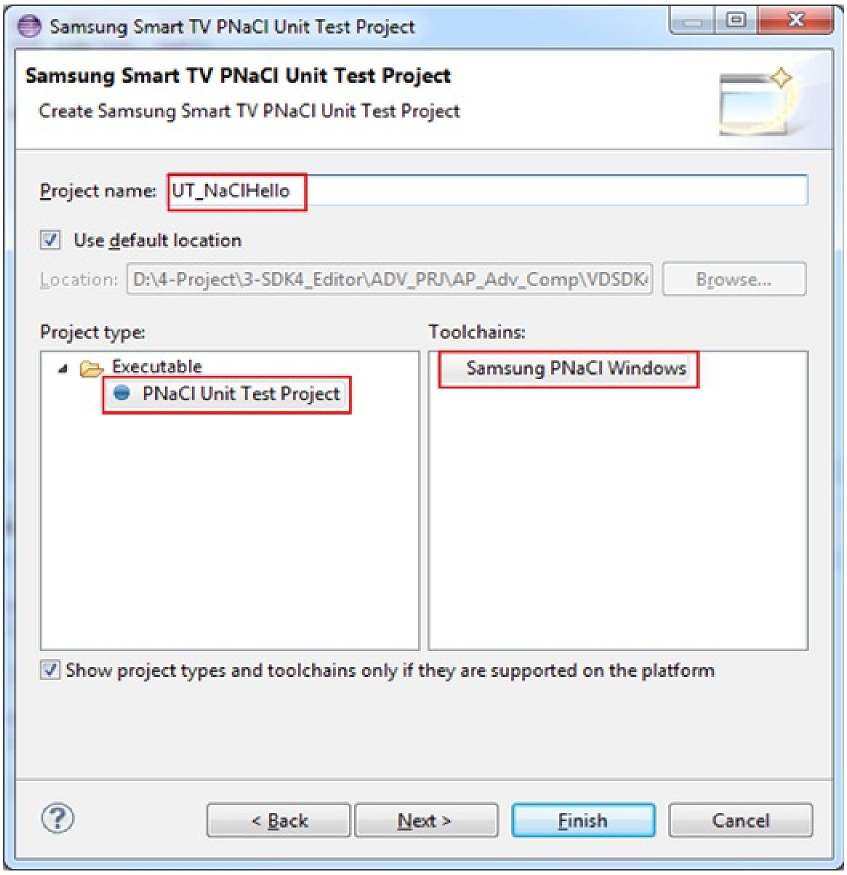


Figure : Create Unit Test Project

4. Set the project name and choose the right tool chain :


Samsung Smart TV PNaCl Unit Test Project
Samsung Smart TV PNaCl Unit Test Project
Create Samsung Smart TV PNaCl Unit Test Project
Browse...
< Back
Next >
Finish
Cancel

Project name: UT_NaClHello

☑ Use default location

Location: D:\4-Project\3-SDK4_Editor\ADV_PRJ\AP_Adv_Comp\VDSDK4

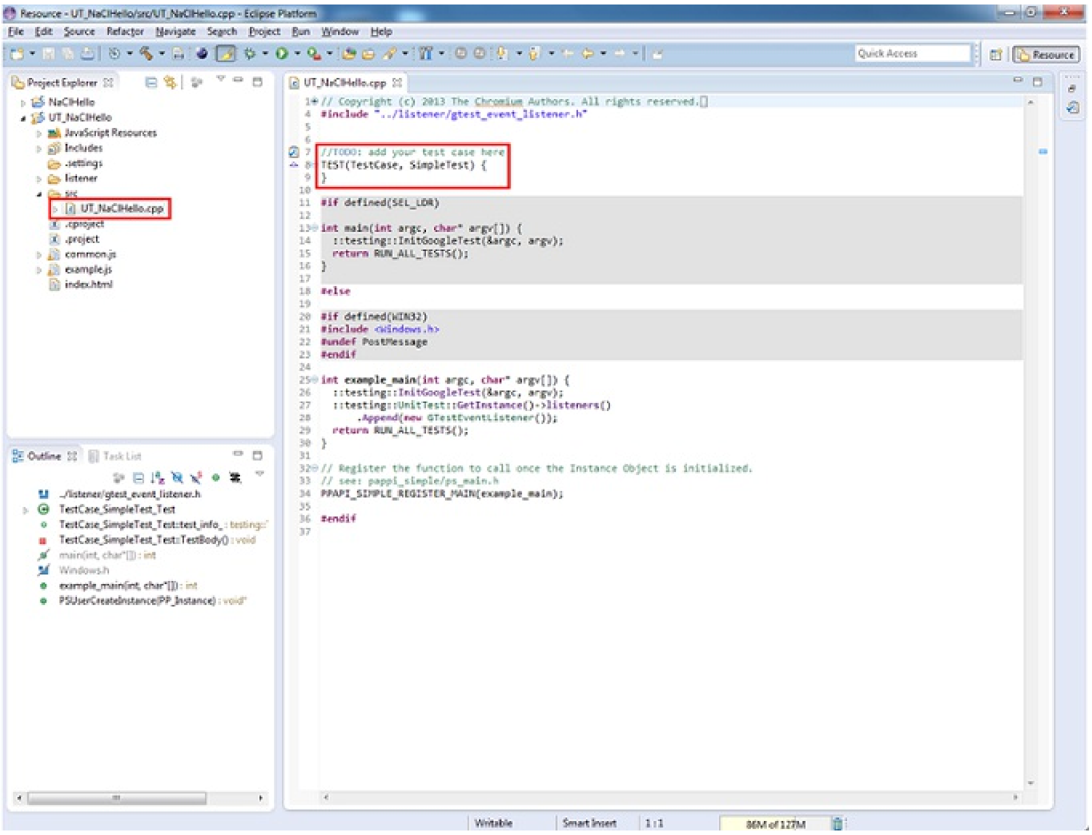
Project type:
- Executable
  - PNaCl Unit Test Project

Toolchains:
- Samsung PNaCl Windows

☑ Show project types and toolchains only if they are supported on the platform

Figure : Choose Tool Chain

5. When finished, the Unit Test project is created, and the template is generated automatically :



```
// Copyright (c) 2013 The Chromium Authors. All rights reserved.
#include "../listener/gtest_event_listener.h"


//TODO: add your test case here
TEST(TestCase, SimpleTest) {
}

#if defined(SEL_LDR)

int main(int argc, char* argv[]) {
  ::testing::InitGoogleTest(&argc, argv);
  return RUN_ALL_TESTS();
}

#else

#if defined(WIN32)
#include <Windows.h>
#undef PostMessage
#endif

int example_main(int argc, char* argv[]) {
  ::testing::InitGoogleTest(&argc, argv);
  ::testing::UnitTest::GetInstance()->listeners()
      .Append(new GTestEventListener());
  return RUN_ALL_TESTS();
}

// Register the function to call once the Instance Object is initialized.
// see: pappi_simple/ps_main.h
PPAPI_SIMPLE_REGISTER_MAIN(example_main);

#endif
```

Figure : Generate Template

PNaCl Project Development

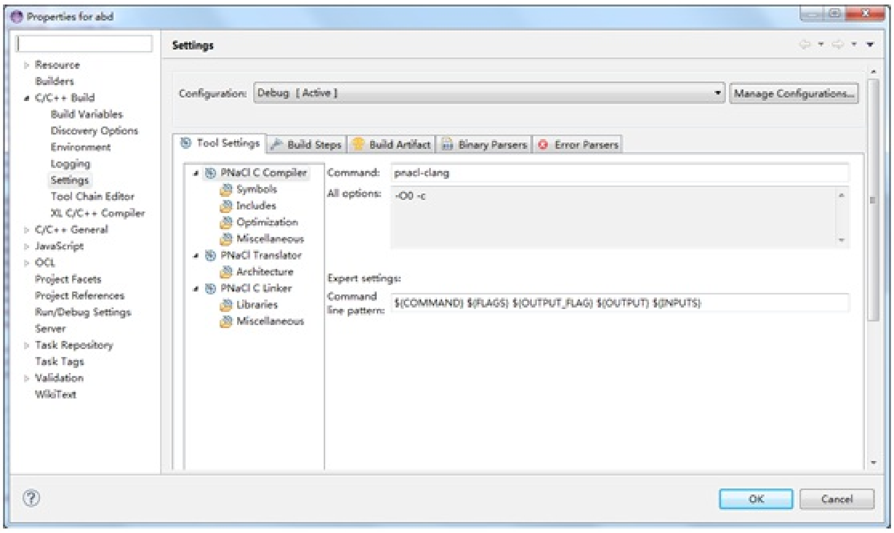
Following the above steps, developers are able to create their own Smart TV App Projects with PNaCl Module. Except the 'Imported PNaCl Module Project', the other 4 kinds of projects' C/C++ source code to be built before the whole project could work. In this section, the 'Build' related contents would be discussed.

Project setting

Use the Tool Settings properties tab to customize the tools and tool options used in the build configuration. To reach PNaCl compiler tool settings page, developers can follow the path as :

Right click current **Samsung PNaCl project -> select 'Properties' ('Alt +Enter') -> 'C/C++ Build' -> 'Settings'**.

The following picture is an overview of tools settings properties :



Configuration: Debug [Active]

Command: pnacl-clang

All options: -O0 -c

Expert settings:

Command line pattern: ${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT} ${INPUTS}

Figure : PNaCl Project Tools Settings

As shown in the picture, the left pane displays a list of tools and their option categories and the right gives the options that developers can modify for the selected tool. This list of options changes according to which option category developers have selected for a specific tool in the left pane.

The following table display the flags listed in the settings windows, which are commonly used.

Tool	Options	Description
PNaCl C/C++ Compiler		**Compile source code and generate PNaCl object file (.po).**
	Symbols	This page can define or undefined symbols for compiler.
	Includes	This page can add include path for current compiler tool. **You may need to add include path to your toolchain (check ‘location-where-you-installed-NaCl-SDK’/pepper_<version>/include on your disk).**
	Optimization	This page can set optimization flags for current compiler tool.
	Miscellaneous	This page can set other flags for current compiler tool.
PNaCl Translator		**Translate .pexe(architecture independent) file to .nexe (architecture dependent) files**
	Architecture	This page can select different architecture for current translator tool.
	Miscellaneous	This page can set other flags for current compiler tool. You may need to update the options, for example when you encounter pepper_25 compilator error: Unrecognized option: –allow-llvm-bitcode-input, remove the option.
PNaCl C/C++ Linker		**Link different PNaCl object files(.po) to PNaCl executable file(.pexe)**
	Libraries	This page can add link libraries and library search path for current linker tool. You may need to add library search path to your toolchain (check ‘location-where-you-installed-NaCl-SDK’/pepper_<version>/lib/pnacl/<> on your disk).
	Miscellaneous	This page can set other flags for current link tool.

> Note
>
> Samsung PNaCl C++ project's compiler tool is different from C project, but their options are the same. So if compile errors comes with “no header file”, developers should add the additional include paths with the “Includes” option of the compiler; if compile errors comes with “no lib file”, developers should check the additional lib paths and lib name with the “Libraries” option of the linker.

Build the Project

After following the steps mentioned above to create either the PNaCl Module Project or the Web + PNaCl Module Project, developers should build the project to get the final output. In order to build the project, developers should take the following steps :

1. In the Project Explorer view, select the project to be built :
2. Right click and select Build Project, or click the build icon on the toolbar.

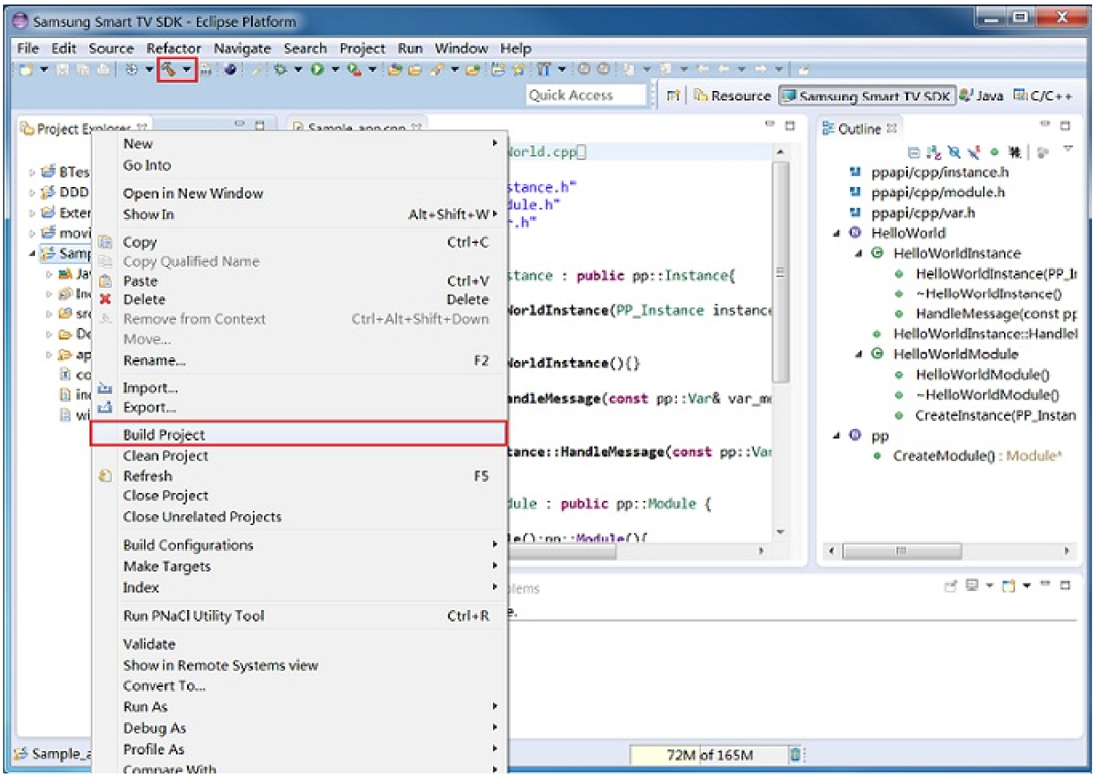
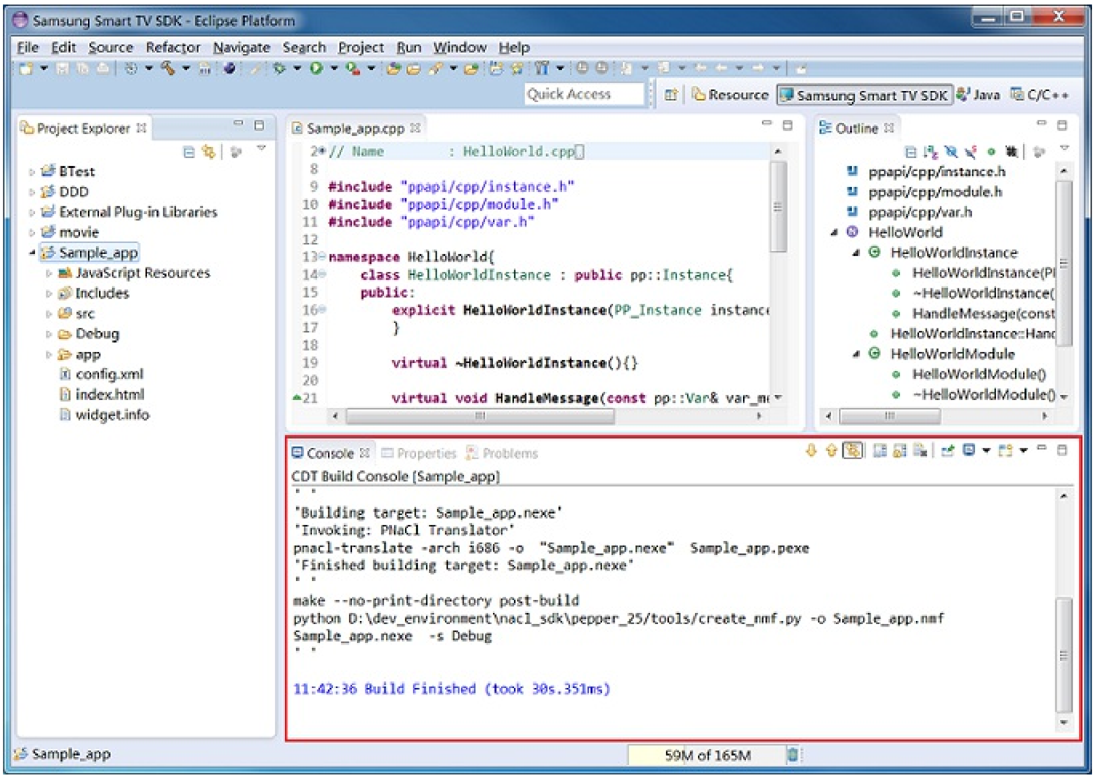


Figure : PNaCl Project Build

3. Developers can see in the Console view the output and results of the build command. Click its tab to bring the view forward if it is not currently visible. If for some reason it is not present, developers can open it by selecting **Window > Show View > Console**.



```
// Name        : HelloWorld.cpp

#include "ppapi/cpp/instance.h"
#include "ppapi/cpp/module.h"
#include "ppapi/cpp/var.h"

namespace HelloWorld{
    class HelloWorldInstance : public pp::Instance{
    public:
        explicit HelloWorldInstance(PP_Instance instanc
        }

        virtual ~HelloWorldInstance(){}

        virtual void HandleMessage(const pp::Var& var_me
```

CDT Build Console [Sample_app]

```
' '
'Building target: Sample_app.nexe'
'Invoking: PNaCl Translator'
pnacl-translate -arch i686 -o  "Sample_app.nexe"  Sample_app.pexe
'Finished building target: Sample_app.nexe'
' '
make --no-print-directory post-build
python D:\dev_environment\nacl_sdk\pepper_25/tools/create_nmf.py -o Sample_app.nmf
Sample_app.nexe  -s Debug
' '

11:42:36 Build Finished (took 30s.351ms)
```

Figure : PNaCl Project Build Output

4. Developers can also switch to Problems view to check if there's any problems reported.

Run the Project

The PNaCl project can run both in the Samsung Smart TV Emulator or the Google Chrome browser.

1. Run in Samsung Smart TV Emulator

   If the PNaCl project created by developer is a 'Web + PNaCl Module(C/C++)' Project or 'Import PNaCl Module into Web' Project, then developers are able to run it in the Samsung Smart TV Emulator.

   To run the project, developers should follow below steps :

   1.1 In the Project Explorer view, select the project and right click on it.
   1.2 Select Run As -> Smart TV Application, Smart Hub page with all Apps in workspace will be shown, then select the project to run.

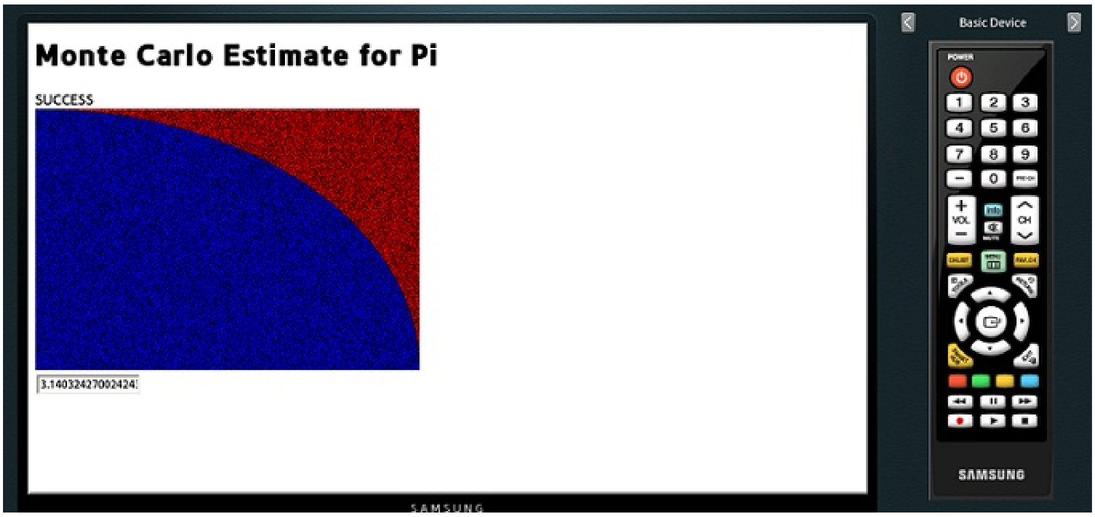


Figure : Run PNaCl App Project in Emulator

2. Run in Chrome Browser

   If the PNaCl project created by developer belongs to PNaCl module(C/C++), developers can run it in Google Chrome browser after the web part is properly configured (Its structure could be quite different from that of Samsung Smart TV PNaCl Project).

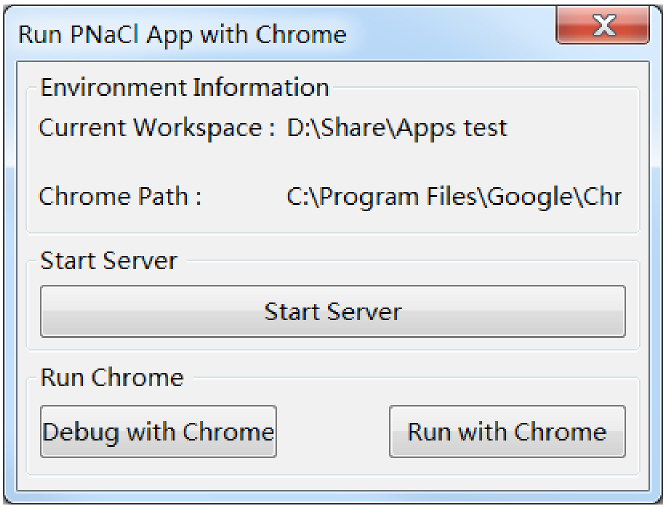
And the follow links could be referred :

- https://developers.google.com/native-client/devguide/tutorial
- https://developers.google.com/native-client/devguide/devcycle/running

After the environment variables, toolchains and chrome browser being properly configured, developers can use the SDK to make the 'Run in Chrome Browser' much more convenient. In order to run project in Chrome Browser, actually, 2 steps should be taken:

2.1 Start a web server;
2.2 Start the Chrome Browser;

Developers can click the toolbar with icon (or) to start a dialogue, where they can start the web server and start the Chrome Browser with different parameters. The dialogue layout is as follows :



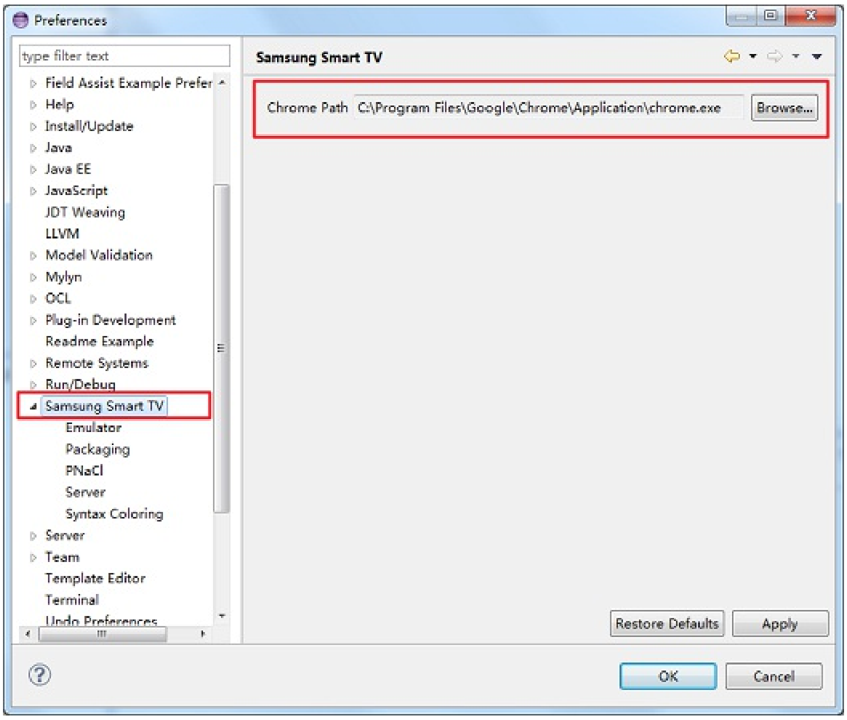
Current Workspace : D:\Share\Apps test

Chrome Path : C:\Program Files\Google\Chr

Figure : Run PNaCl Project in Chrome

The following table explains the details of the dialogue :

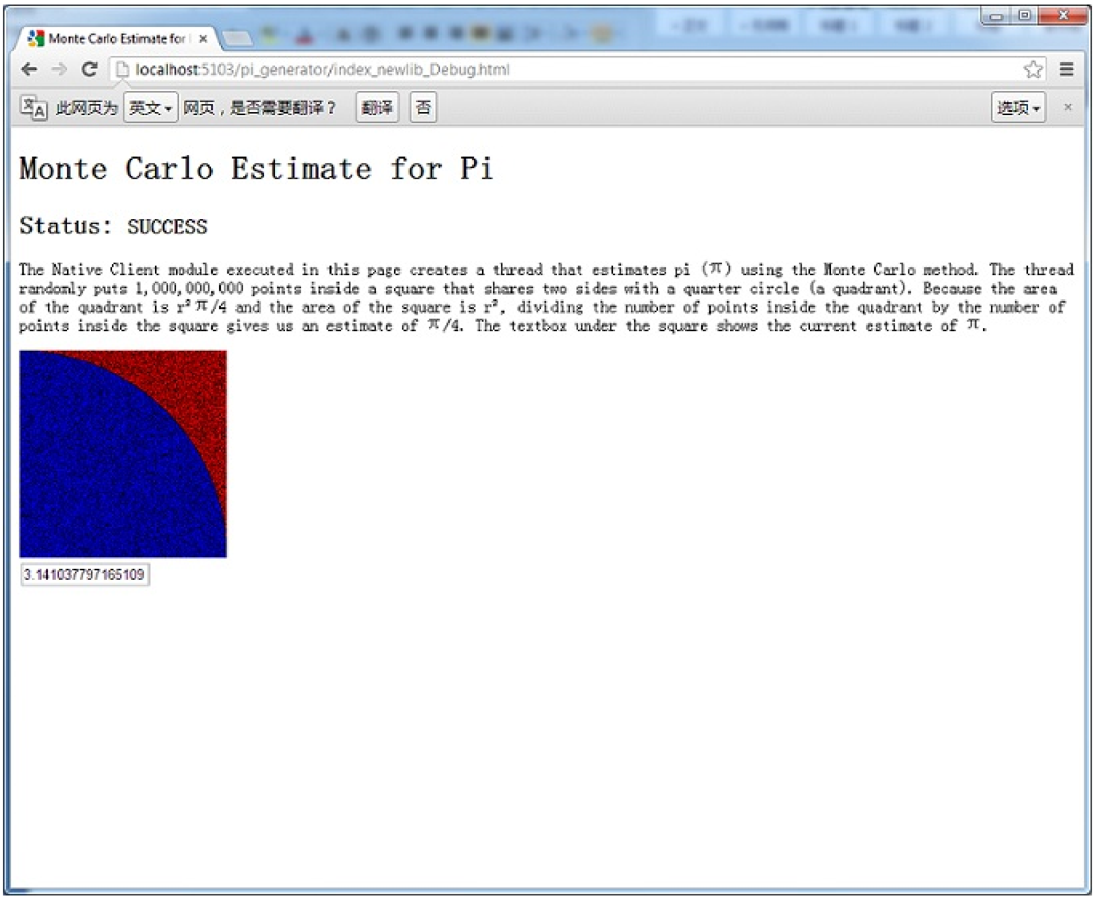
Items	Description
Current Workspace	Display the current workspace location.
Chrome Path	Display the chrome executable path in current OS. It could be set in the **Preferences->Samsung Smart TV**. If it has not been set, a warning message with red color will be displayed.
Start Server	This button help to start the web server using the Python Script. Notice:The script should be copied into the current workspace
Debug with Chrome	Start Chrome Browser with 'Support NaCl Debug' parameter.
Run with Chrome	Start Chrome Browser without 'Support NaCl Debug' parameter.



Chrome Path C:\Program Files\Google\Chrome\Application\chrome.exe

Figure : Chrome Path Settings in SDK Preference Dialogue

The following picture displays the PNaCl Project Running in Chrome.



localhost:5103/pi_generator/index_newlib_Debug.html

Monte Carlo Estimate for Pi

Status: SUCCESS

The Native Client module executed in this page creates a thread that estimates pi (π) using the Monte Carlo method. The thread randomly puts 1,000,000,000 points inside a square that shares two sides with a quarter circle (a quadrant). Because the area of the quadrant is $r^2\pi/4$ and the area of the square is r^2, dividing the number of points inside the quadrant by the number of points inside the square gives us an estimate of π/4. The textbox under the square shows the current estimate of π.

3.141037797165109

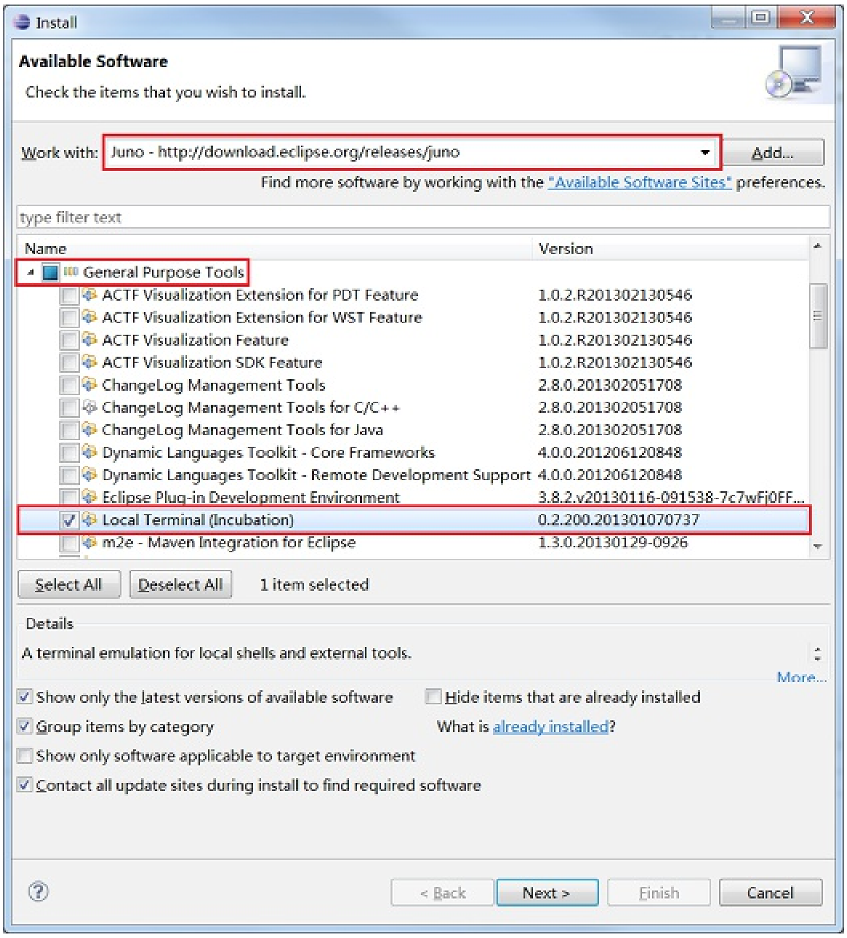
Figure : Run PNaCl Project in Chrome

Utility Tools

The PNaCl Utility Tools depends on the Local Terminal plug-in, install this plug-in if your Eclipse hasn't installed it. Follow these steps to install it :

1. Select Help -> Install New Software..., select the Juno's update site.
2. Choose the 'General Purpose Tools'->'Local Terminal' in Eclipse item.

3.



Available Software

Check the items that you wish to install.

Work with: Juno - http://download.eclipse.org/releases/juno

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
■ General Purpose Tools	
☐ ACTF Visualization Extension for PDT Feature	1.0.2.R201302130546
☐ ACTF Visualization Extension for WST Feature	1.0.2.R201302130546
☐ ACTF Visualization Feature	1.0.2.R201302130546
☐ ACTF Visualization SDK Feature	1.0.2.R201302130546
☐ ChangeLog Management Tools	2.8.0.201302051708
☐ ChangeLog Management Tools for C/C++	2.8.0.201302051708
☐ ChangeLog Management Tools for Java	2.8.0.201302051708
☐ Dynamic Languages Toolkit - Core Frameworks	4.0.0.201206120848
☐ Dynamic Languages Toolkit - Remote Development Support	4.0.0.201206120848
☐ Eclipse Plug-in Development Environment	3.8.2.v20130116-091538-7c7wFj0FF...
☑ Local Terminal (Incubation)	0.2.200.201301070737
☐ m2e - Maven Integration for Eclipse	1.3.0.20130129-0926

1 item selected

Details

A terminal emulation for local shells and external tools.

☑ Show only the latest versions of available software
☐ Hide items that are already installed
☑ Group items by category
What is already installed?
☐ Show only software applicable to target environment
☑ Contact all update sites during install to find required software

Install this Plug-in. **Figure** : Install Local Terminal Plug-in

After your IDE installed the Terminal plug-in, you can run the PNaCl Utility Tools :

1. Create a project by Project Wizard.
2. Right Click this project, then choose the ‘Run PNaCl Utility Tools’ in the context menu (or press Ctrl+R).

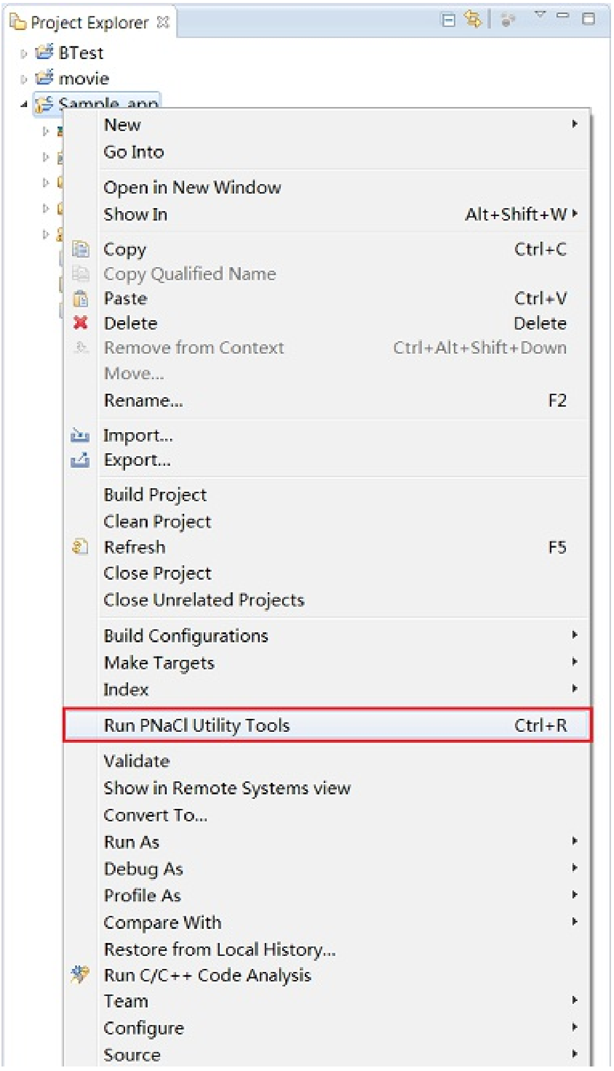
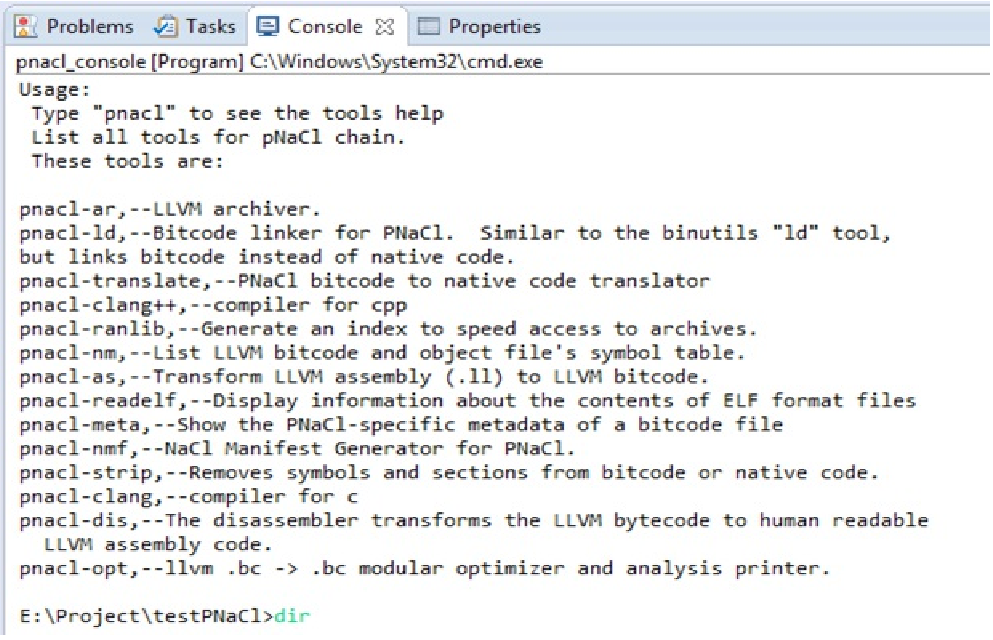


Figure : Run PNaCl Utility Tools

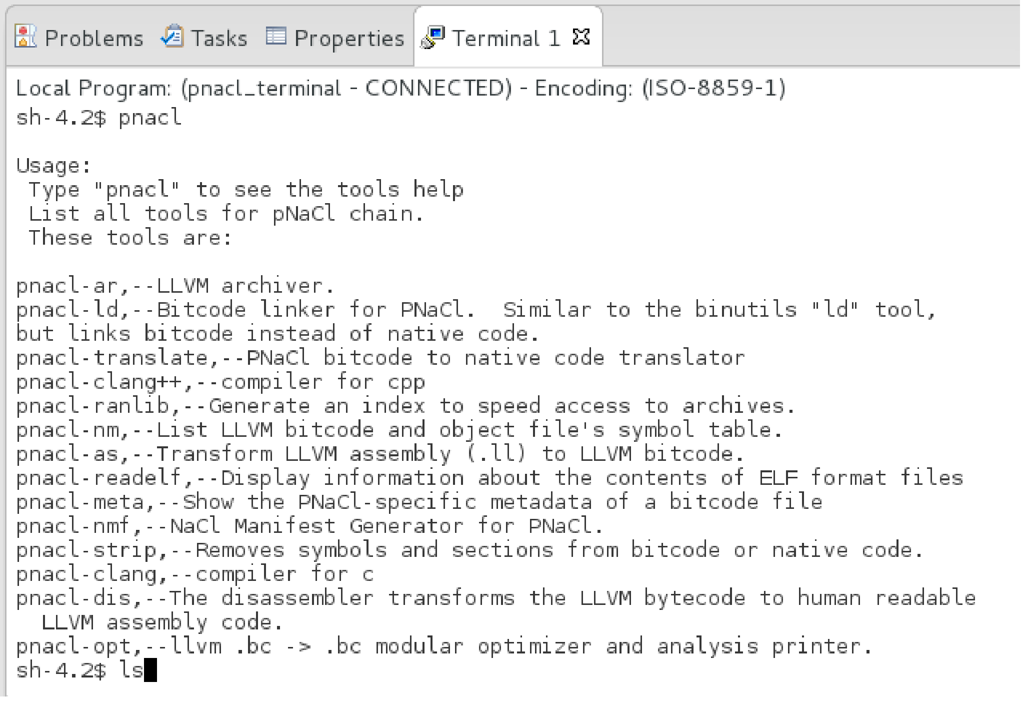
Use the PNaCl Utility Tools :

1. When execute the ‘Run PNaCl Utility Tools’ command, the help information will be printed in Console (Windows) or Terminal(Linux/Mac).
2. In this help information, each illustration for PNaCl tools will be listed as follows.
3. Input these PNaCl commands for detailed information like ‘pnacl-clang --help’, also input ‘pnacl’ command to see all PNaCl tools illustrations again.



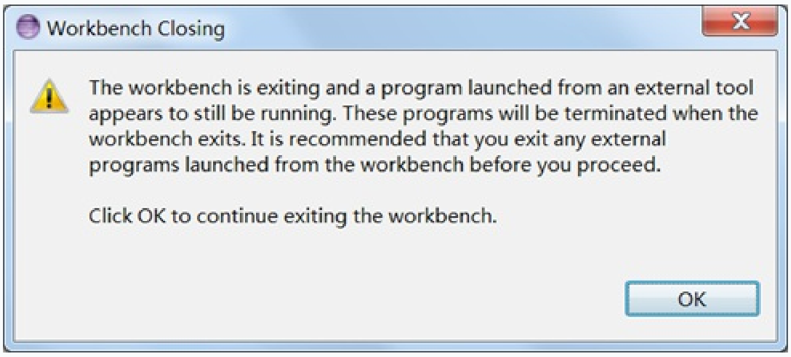
```
Usage:
 Type "pnacl" to see the tools help
 List all tools for pNaCl chain.
 These tools are:

pnacl-ar,--LLVM archiver.
pnacl-ld,--Bitcode linker for PNaCl.  Similar to the binutils "ld" tool,
but links bitcode instead of native code.
pnacl-translate,--PNaCl bitcode to native code translator
pnacl-clang++,--compiler for cpp
pnacl-ranlib,--Generate an index to speed access to archives.
pnacl-nm,--List LLVM bitcode and object file's symbol table.
pnacl-as,--Transform LLVM assembly (.ll) to LLVM bitcode.
pnacl-readelf,--Display information about the contents of ELF format files
pnacl-meta,--Show the PNaCl-specific metadata of a bitcode file
pnacl-nmf,--NaCl Manifest Generator for PNaCl.
pnacl-strip,--Removes symbols and sections from bitcode or native code.
pnacl-clang,--compiler for c
pnacl-dis,--The disassembler transforms the LLVM bytecode to human readable
  LLVM assembly code.
pnacl-opt,--llvm .bc -> .bc modular optimizer and analysis printer.

E:\Project\testPNaCl>dir
```

Figure : Use Utility Tools (Windows)



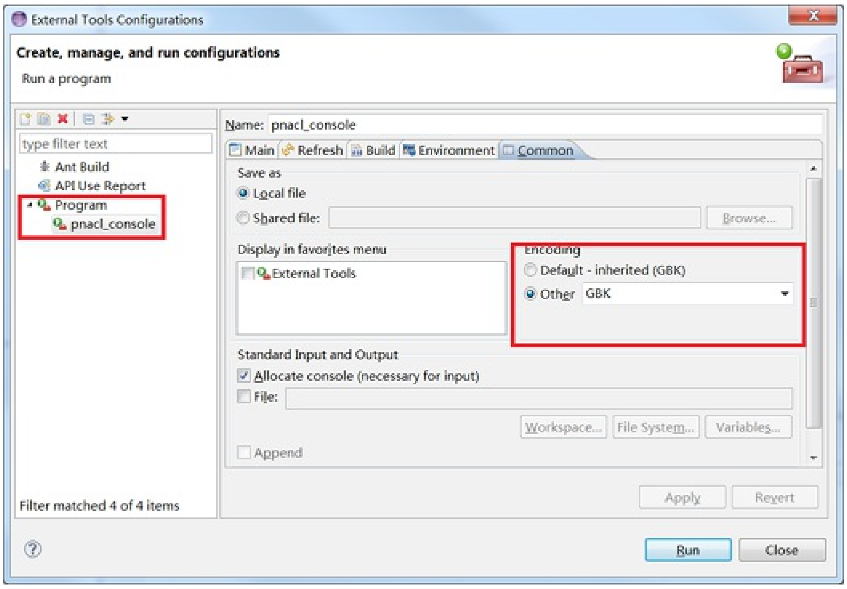
```
Local Program: (pnacl_terminal - CONNECTED) - Encoding: (ISO-8859-1)
sh-4.2$ pnacl

Usage:
 Type "pnacl" to see the tools help
 List all tools for pNaCl chain.
 These tools are:

pnacl-ar,--LLVM archiver.
pnacl-ld,--Bitcode linker for PNaCl.  Similar to the binutils "ld" tool,
but links bitcode instead of native code.
pnacl-translate,--PNaCl bitcode to native code translator
pnacl-clang++,--compiler for cpp
pnacl-ranlib,--Generate an index to speed access to archives.
pnacl-nm,--List LLVM bitcode and object file's symbol table.
pnacl-as,--Transform LLVM assembly (.ll) to LLVM bitcode.
pnacl-readelf,--Display information about the contents of ELF format files
pnacl-meta,--Show the PNaCl-specific metadata of a bitcode file
pnacl-nmf,--NaCl Manifest Generator for PNaCl.
pnacl-strip,--Removes symbols and sections from bitcode or native code.
pnacl-clang,--compiler for c
pnacl-dis,--The disassembler transforms the LLVM bytecode to human readable
  LLVM assembly code.
pnacl-opt,--llvm .bc -> .bc modular optimizer and analysis printer.
sh-4.2$ ls
```

Figure : Use Utility Tools (Linux/Mac)

> Note
>
> Please do not forget to Terminate (🟥) the Console (Windows) or Disconnect ( ) the Terminal (Linux/Mac) after you complete your work in the Console/Terminal, because they take up some of the background resource, and if you do not Terminate/Disconnect them when you exit your SDK IDE, you may get this Warning dialogue:
>
> 
>
>
> The workbench is exiting and a program launched from an external tool appears to still be running. These programs will be terminated when the workbench exits. It is recommended that you exit any external programs launched from the workbench before you proceed.
>
> Click OK to continue exiting the workbench.
>
> **Figure**: Warning not exit program

Set the Console encoding (Windows) :

If your Console display is garbled, close it, then select Run->External Tools->External Tools Configurations..., select the Program->pnacal_console, on the right, choose the Common tab, in the Encoding section, select or input your current OS' encoding, (for example: GBK for the Chinese) :



Name: pnacl_console

● Local file

● Other GBK

☑ Allocate console (necessary for input)

Figure : Set the Console encoding

> Note
>
> If your External Tools Configurations dialog does not have the Program->pnacal_console node, please close the dialog and run the PNaCl Utility Tools, then re-open this dialog.

Set the Terminal encoding (Linux/Mac) :

If your Terminal display is garbled, press the ‘Settings’ button on the Terminal view :

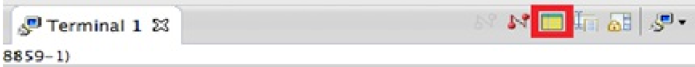
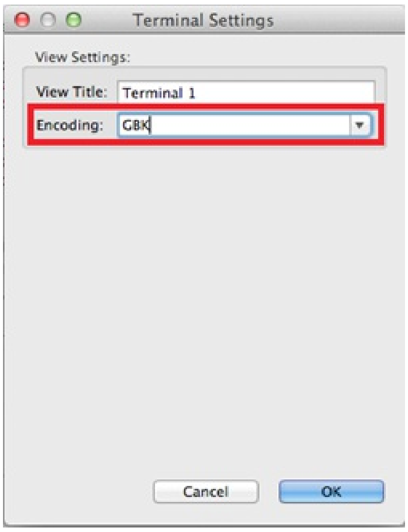


Figure : The Terminal toolbar

In the Terminal Setting dialog ,specify the encoding the same as your current OS’ :



View Title: Terminal 1

Encoding: GBK

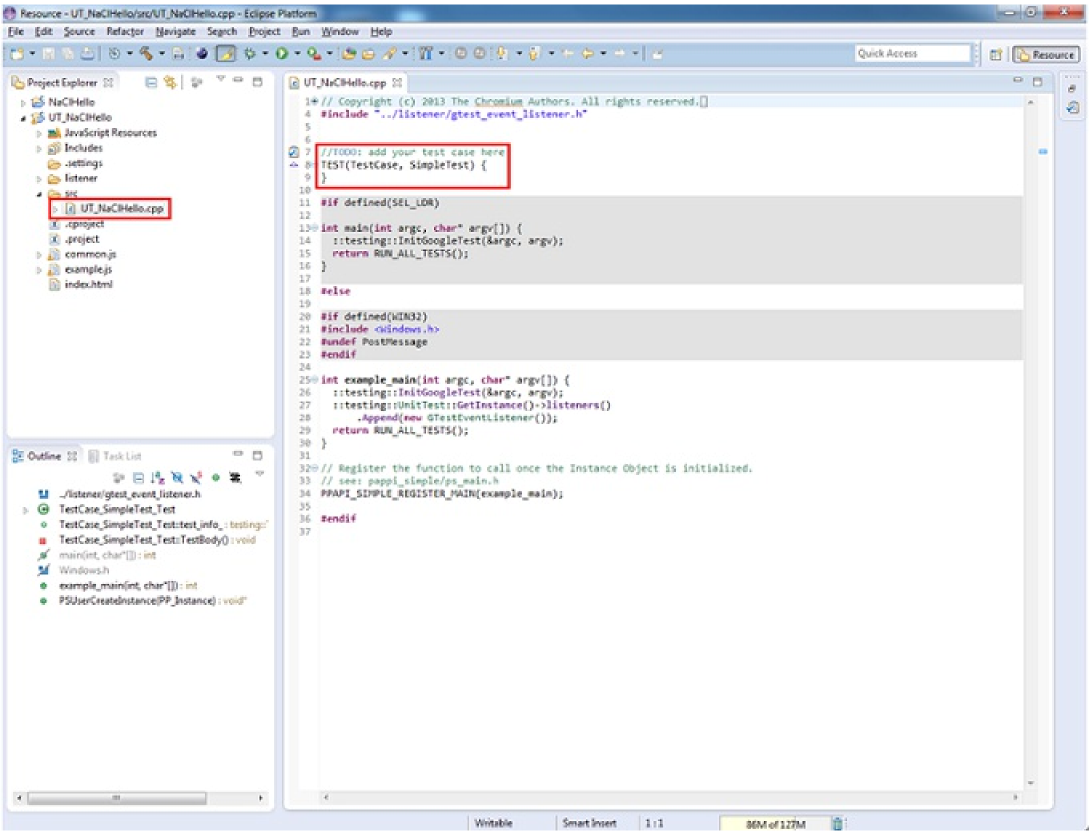
Figure : Set the Terminal encoding

PNaCl Unit Test

The PNaCl tool chain has been upgraded to pepper_31, and one of the new features is the Unit Test based on Google C++ Testing Framework.

The following sections show how developers could do Unit Test for normal PNaCl projects.

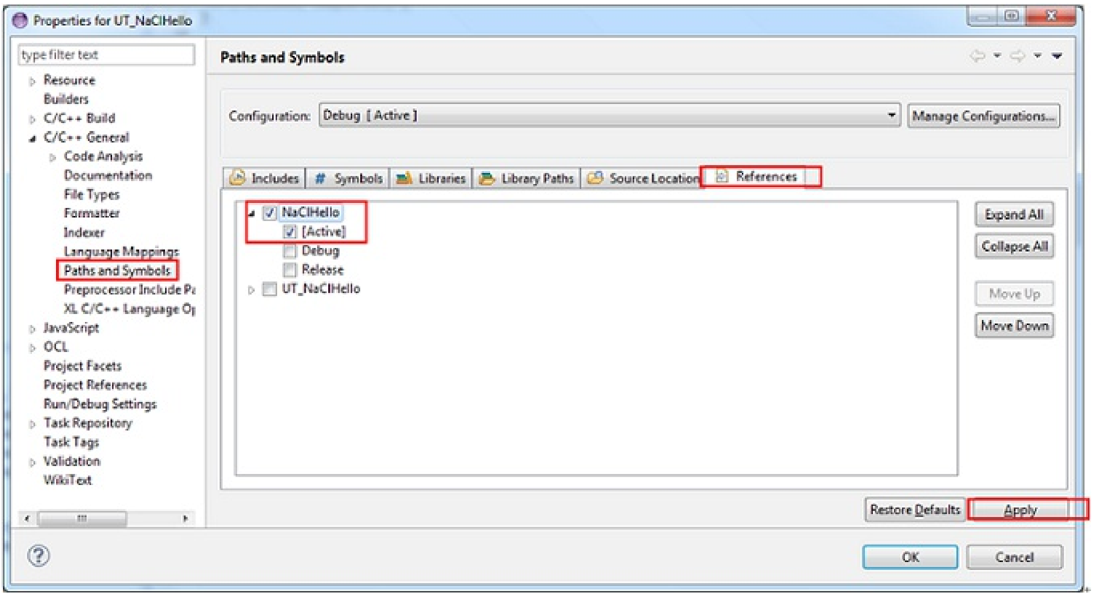
1. Follow the Create PNaCl Unit Test Project to create the unit test project :



```
// Copyright (c) 2013 The Chromium Authors. All rights reserved.
#include "../listener/gtest_event_listener.h"


//TODO: add your test case here
TEST(TestCase, SimpleTest) {
}

#if defined(SEL_LDR)

int main(int argc, char* argv[]) {
  ::testing::InitGoogleTest(&argc, argv);
  return RUN_ALL_TESTS();
}

#else

#if defined(WIN32)
#include <Windows.h>
#undef PostMessage
#endif

int example_main(int argc, char* argv[]) {
  ::testing::InitGoogleTest(&argc, argv);
  ::testing::UnitTest::GetInstance()->listeners()
      .Append(new GTestEventListener());
  return RUN_ALL_TESTS();
}

// Register the function to call once the Instance Object is initialized.
// see: pappi_simple/ps_main.h
PPAPI_SIMPLE_REGISTER_MAIN(example_main);

#endif
```

Figure : Generate Template

2. Build relationships between the Unit Test project and its target project, Set the target project as the Unit Test Project's reference in project Properties :

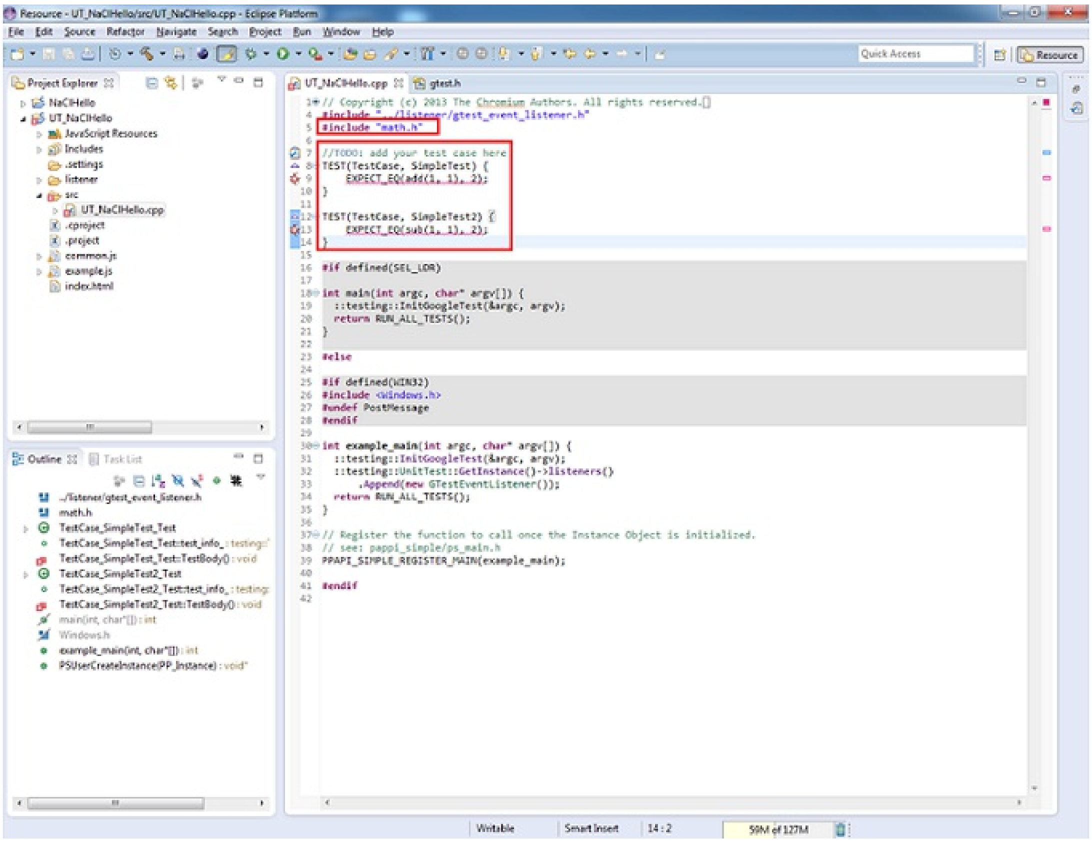


Configuration: Debug [Active]

- ☑ NaClHello
  - ☑ [Active]
  - ☐ Debug
  - ☐ Release
- ☐ UT_NaClHello

Figure : Set Project to Reference

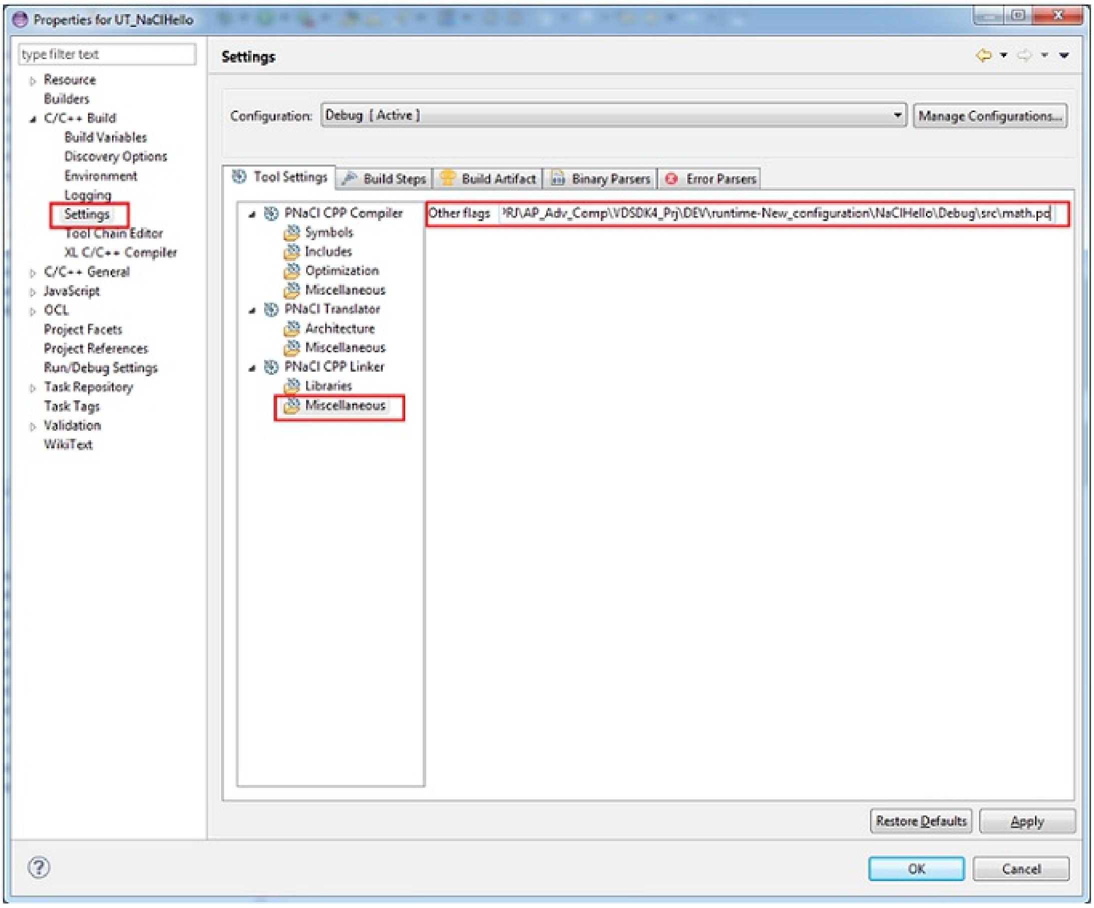
3. Add the prototype of the functions (or just include the .h file) to be tested, Then add the test case :


Resource - UT_NaClHello/src/UT_NaClHello.cpp - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Resource
Project Explorer
NaClHello
UT_NaClHello
JavaScript Resources
Includes
.settings
listener
src
UT_NaClHello.cpp
.cproject
.project
common.js
example.js
index.html
Outline
Task List
../listener/gtest_event_listener.h
math.h
TestCase_SimpleTest_Test
TestCase_SimpleTest_Test::test_info_ : testing::
TestCase_SimpleTest_Test::TestBody() : void
TestCase_SimpleTest2_Test
TestCase_SimpleTest2_Test::test_info_ : testing:
TestCase_SimpleTest2_Test::TestBody() : void
main(int, char*[]) : int
Windows.h
example_main(int, char*[]) : int
PSUserCreateInstance(PP_Instance) : void*
UT_NaClHello.cpp
gtest.h
Writable
Smart Insert
14 : 2
59M of 127M

```
// Copyright (c) 2013 The Chromium Authors. All rights reserved.
#include "../listener/gtest_event_listener.h"
#include "math.h"

//TODO: add your test case here
TEST(TestCase, SimpleTest) {
    EXPECT_EQ(add(1, 1), 2);
}

TEST(TestCase, SimpleTest2) {
    EXPECT_EQ(sub(1, 1), 2);
}

#if defined(SEL_LDR)

int main(int argc, char* argv[]) {
  ::testing::InitGoogleTest(&argc, argv);
  return RUN_ALL_TESTS();
}

#else

#if defined(WIN32)
#include <Windows.h>
#undef PostMessage
#endif

int example_main(int argc, char* argv[]) {
  ::testing::InitGoogleTest(&argc, argv);
  ::testing::UnitTest::GetInstance()->listeners()
      .Append(new GTestEventListener());
  return RUN_ALL_TESTS();
}

// Register the function to call once the Instance Object is initialized.
// see: pappi_simple/ps_main.h
PPAPI_SIMPLE_REGISTER_MAIN(example_main);

#endif
```

Figure : Add Test Case

4. Add the additional include path where the .h file locates , and the additional object file to the linker (math.h and math.po for this example) :

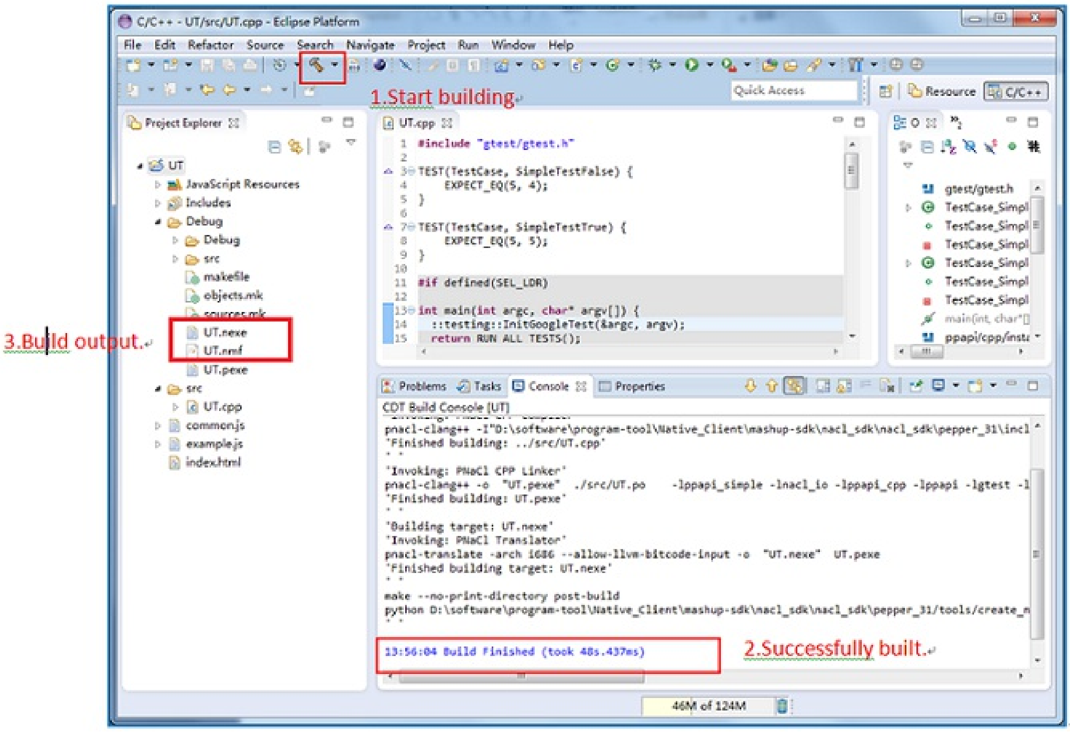


Configuration: Debug [Active]

Other flags `RJ\AP_Adv_Comp\VDSDK4_Prj\DEV\runtime-New_configuration\NaClHello\Debug\src\math.po`

Figure : Add Additional Object File

5. Build the Unit Test project with the following steps :



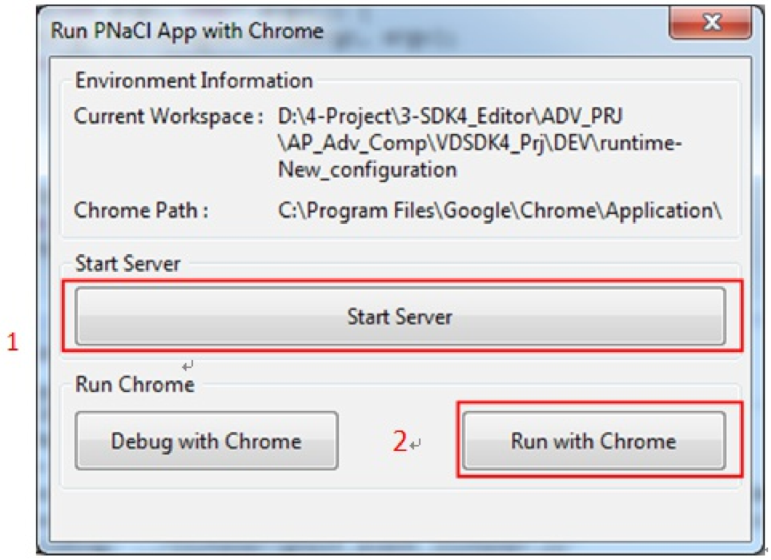
```
#include "gtest/gtest.h"

TEST(TestCase, SimpleTestFalse) {
    EXPECT_EQ(5, 4);
}

TEST(TestCase, SimpleTestTrue) {
    EXPECT_EQ(5, 5);
}

#if defined(SEL_LDR)

int main(int argc, char* argv[]) {
  ::testing::InitGoogleTest(&argc, argv);
  return RUN_ALL_TESTS();
```

CDT Build Console [UT]

```
pnacl-clang++ -I"D:\software\program-tool\Native_Client\mashup-sdk\nacl_sdk\nacl_sdk\pepper_31\incl
'Finished building: ../src/UT.cpp'
' '
'Invoking: PNaCl CPP Linker'
pnacl-clang++ -o  "UT.pexe"  ./src/UT.po    -lppapi_simple -lnacl_io -lppapi_cpp -lppapi -lgtest -l
'Finished building: UT.pexe'
' '
'Building target: UT.nexe'
'Invoking: PNaCl Translator'
pnacl-translate -arch i686 --allow-llvm-bitcode-input -o  "UT.nexe"  UT.pexe
'Finished building target: UT.nexe'
' '
make --no-print-directory post-build
python D:\software\program-tool\Native_Client\mashup-sdk\nacl_sdk\nacl_sdk\pepper_31/tools/create_n
' '

13:56:04 Build Finished (took 48s.437ms)
```

Figure : Build Unit Test Project

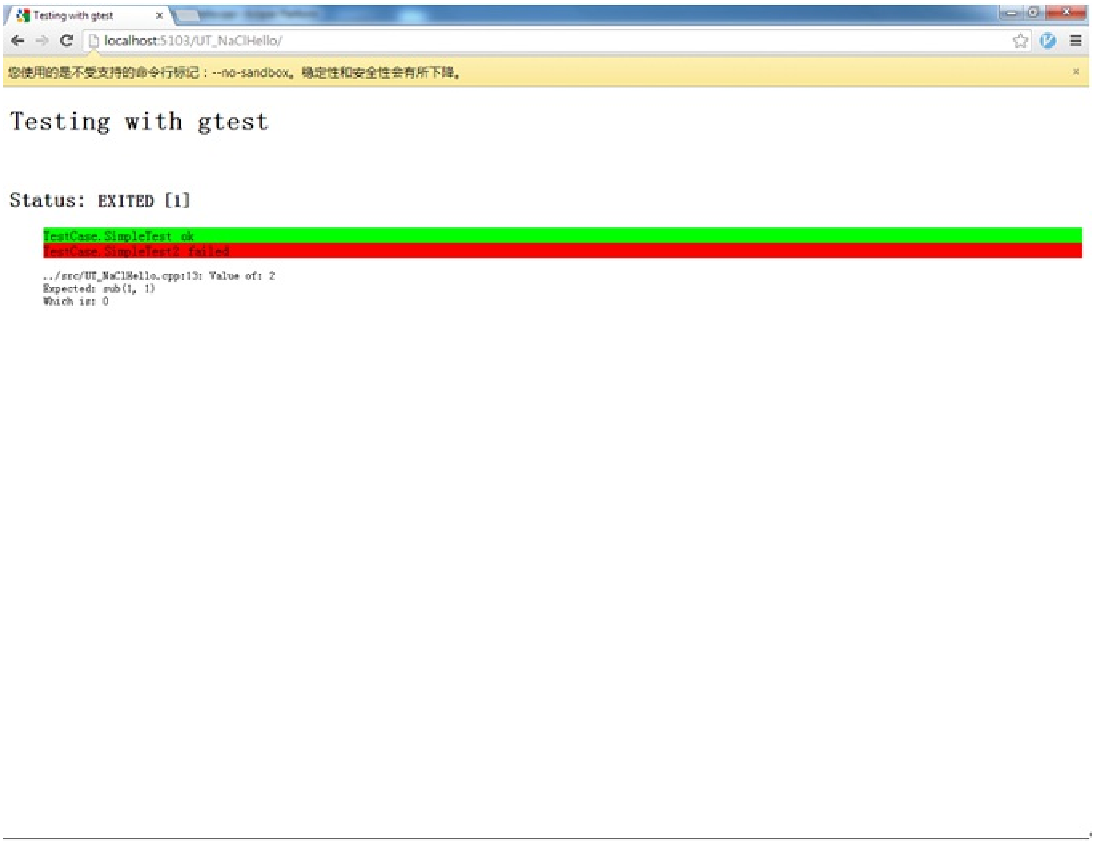
6. Run the Unit Test project in Chrome and check the result. Obviously, the first test case is passed, while the second is failed



Current Workspace : D:\4-Project\3-SDK4_Editor\ADV_PRJ\AP_Adv_Comp\VDSDK4_Prj\DEV\runtime-New_configuration

Chrome Path : C:\Program Files\Google\Chrome\Application\

: **Figure** : Run Unit Test Project



您使用的是不受支持的命令行标记：--no-sandbox。稳定性和安全性会有所下降。

Testing with gtest

Status: EXITED [1]

```
TestCase.SimpleTest ok
TestCase.SimpleTest2 failed

../src/UT_NaClHello.cpp:13: Value of: 2
Expected: sub(1, 1)
Which is: 0
```

Figure : Show Cases Result

Related Documents

How to create sample PNaCl application