



DVT Eclipse IDE

My First e Language Project

> The DVT Perspective



- Switch to the **DVT Perspective** from

menu Window > Open Perspective > Other... > DVT

- The DVT Perspective presents different Views (GUI components) around the Editor



Views

Types
Compile Order
Verification Hierarchy
Type Hierarchy

...

> The Project Location



- You typically create a project in a folder that contains the source code files

*It is not mandatory to create a project where the source files are
All "outside the project" sources will be presented under DVT Auto-Linked*

- DVT creates **.project** and **.dvt** in the project location folder

*The .project file indicates to Eclipse that a project exists in the folder
The .dvt folder contains various DVT specific project settings*

```
my_vip/  
  e/  
    core/  
    examples/
```



```
my_vip/  
  .project  
  .dvt  
  e/  
    core/  
    examples/
```

> The New Project Wizard



New Project

Project Location

Specify project location

Directory:

Project Name: Custom

If the directory doesn't exist, it will be created.

DVT doesn't copy any files or directories. A project corresponds to a directory on the disk and it shows its content as is. Any directory content changes done in DVT are visible outside. Any directory content changes done outside are visible in DVT upon refresh. Refresh must be explicitly triggered, unless Eclipse is configured to do it automatically.

You can also choose from [Predefined Projects](#).

Select one or more project natures: Verilog/SystemVerilog e Language VHDL

A specific language source files are compiled only if the project has the specific nature.

- Invoke the wizard

From menu File > New > DVT Project

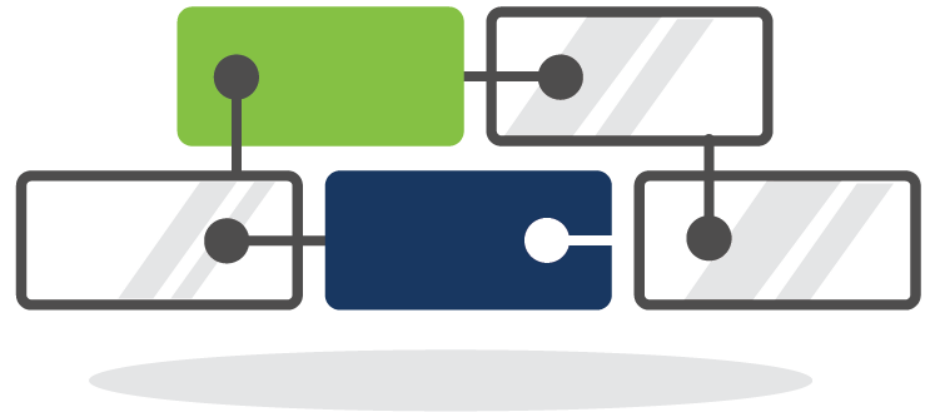
- Specify the project location
- Specify the project nature. If a project was already configured, that is if .project and .dvt already exist, the wizard recognizes the existing project

*You can open one of the **Predefined Projects**, if you want to see how an example project is configured*

> Build Configurations



- In order to provide advanced functionality (like hyperlinks, autocomplete, design and class hierarchy, error signaling, etc.) DVT analyzes the source code files in your project. This analysis process is called **build**
- In order to build, DVT uses the compilation arguments that you specify in a build file. The default build file is **.dvt/default.build**



> The .build File Syntax



- You can change the `.dvt/default.build`
- In a `.build` file you can specify:

Absolute paths or project root relative paths

System variables like `${var}` or `%var%`

`+dvt_setenv+SPECMAN_PATH=<path>` directives to configure the `SPECMAN_PATH`

`+define+<DEFINE>=<replacement>` or `-define <DEFINE>=<replacement>` directives

`-f <path>` or `-F <path>` to include a file containing more arguments


- For more options see: https://www.dvteclipse.com/documentation/e/Build_Configurations.html

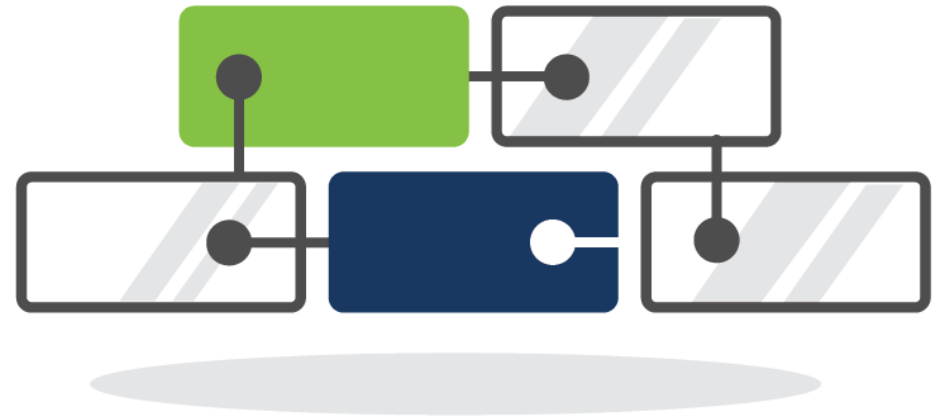
> Build the Project

- Building a project means compiling and indexing all the source files in order to provide hyperlinks, autocomplete, class browsing ...

Full Build or Rebuild = compile all using directives from the current build file

Incremental Build = compile changes. As you edit files, DVT incrementally builds the project.

-  ● After changing `.dvt/*.build`, you have to Rebuild the project from the toolbar button



> Check the Build



- The **Compile Order View** shows all the compiled files

*You can open the Compile Order View from menu Window > Show View > Compile Order
It is typically located on the right side of the Editor, behind the Outline View*

- The **Problems View** shows all the errors and warnings in your project

Build Errors: file not found, imported file not found ...

Syntax Errors: unrecognized action "whie" instead of "while" ...

Semantic Errors: duplicate declarations, extending non existing types ...

You can open the Problems View from menu Window > Show View > Problems

It is typically located below the Editor

> Features Overview [1]



- **Hyperlinks:** in the editor press Ctrl and hover struct names, method names, and in general any identifier. A hyperlink appears. Click the it to go to the definition. In addition to the hyperlink, a drop-down presents more options, for example Show Usages
- **Show Usages:** in the editor press Ctrl and hover an identifier. From the drop-down chose Show Usages to see all places were a variable, function, struct, macro etc. is used
- **Autocomplete:** in the editor Ctrl + Space triggers autocomplete. For example driver. <Ctrl+Space here> will show driver API
- **Quick Fixes:** in the editor on a line with errors Ctrl +1 pops-up quick fix proposals to correct typos, to declare missing variables etc.
- **Rename Refactoring:** place the cursor over an identifier and right click > Refactor > Rename or Shift+Alt+R to rename and update all usages across the entire project

> Features Overview [2]



- **Class Hierarchy:** place the cursor over a struct name and press F4 to see the OOP inheritance. Ctrl + T is a quick way to navigate the OOP hierarchy – it works for structs and methods/TCMs
- **Verification Hierarchy:** place the cursor over a unit name and press Shift + F6 to see the unit instance tree
- **All Structs/Units/Enums/...:** menu Window > Show View > Types
- **All Macros:** menu Window > Show View > Macros
- **All TODOs/FIXMEs:** menu Window > Show View > Tasks
- **Code Templates:** menu Window > Show View > Code Templates
- **To quickly find a struct or unit:** Ctrl + Shift + T
- **To quickly open a file:** Ctrl + I or Ctrl + Shift + R

> Features Overview [3]



- **Diagrams:** Right click on a struct to get UML diagrams
- **Semantic Search:** `Ctrl + H` to search for a struct, variable... both definitions and/or where it is used
- **Code Formatting:** press `Ctrl+Shift+F` to format the whole editor or selection
- **Toggle Comment:** `Ctrl + /` for current line or selection
- **Matching Begin – End:** double-click after curly or paren.
- **Column Selection:** `Shift+Ctrl+A` or from the main toolbar button
- **All Shortcuts:** press `Ctrl + Shift + L` to pop-up a list of all shortcuts



And many more, please contact support@amiq.com for a demo.

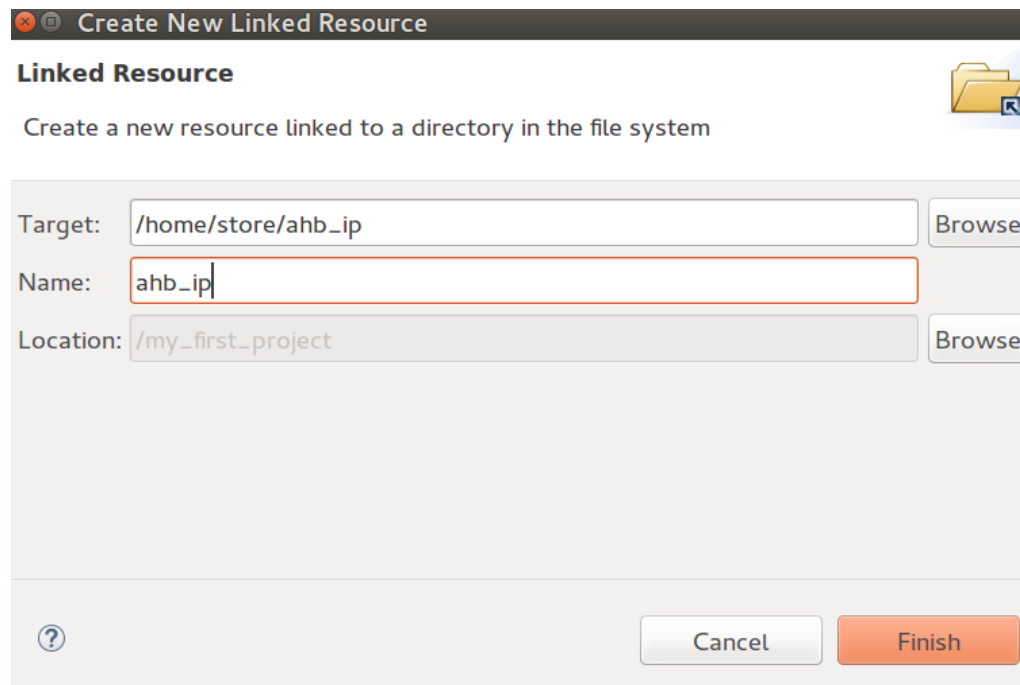
> Advanced: Linked Resources [1]



- Sometimes your source code is spread on the disk, not everything is under one “source code root” folder
- Files outside the Project Location folder (“external files”) are by default automatically brought during a full build under the **DVT Auto-Linked virtual folder**
- What you see there are the actual files, any change will be visible on the disk and the other way around
- The hierarchy under DVT Auto-Linked may be too deep. To bring the files “closer” you can:
 - Use **+dvt_auto+link_root+alias=/path/to/folder** directives in **default.build**
 - Use **Linked Resources**. Linked Resources are logical entities, no additional files are created on your disk. Their definition is stored in the **.project** file

> Advanced: Linked Resources [2]

Right click on Project > **New** > **DVT Linked Resource**, specify the target location and linked resource name



Create New Linked Resource

Linked Resource

Create a new resource linked to a directory in the file system

Target: /home/store/ahb_ip

Name: ahb_ip

Location: /my_first_project

Rebuild the project in order to avoid duplicate files in the newly created folder and DVT Auto-Linked

> More Information



- Demo Movies: <https://www.youtube.com/user/dvteclipse>
- Step by step basic tutorial: https://www.dvteclipse.com/documentation/e/Getting_Started.html
 - Please contact us for more training materials
- Features with snapshots: https://www.dvteclipse.com/documentation/e/Tips_and_Tricks.html
- User Guide: <https://www.dvteclipse.com/documentation/e/index.html>
- Datasheet: https://dvteclipse.com/datasheet/AMIQ_DVT_Datasheet.pdf



Mail to support@amiq.com