ShiVa module

polyglot-visualnode

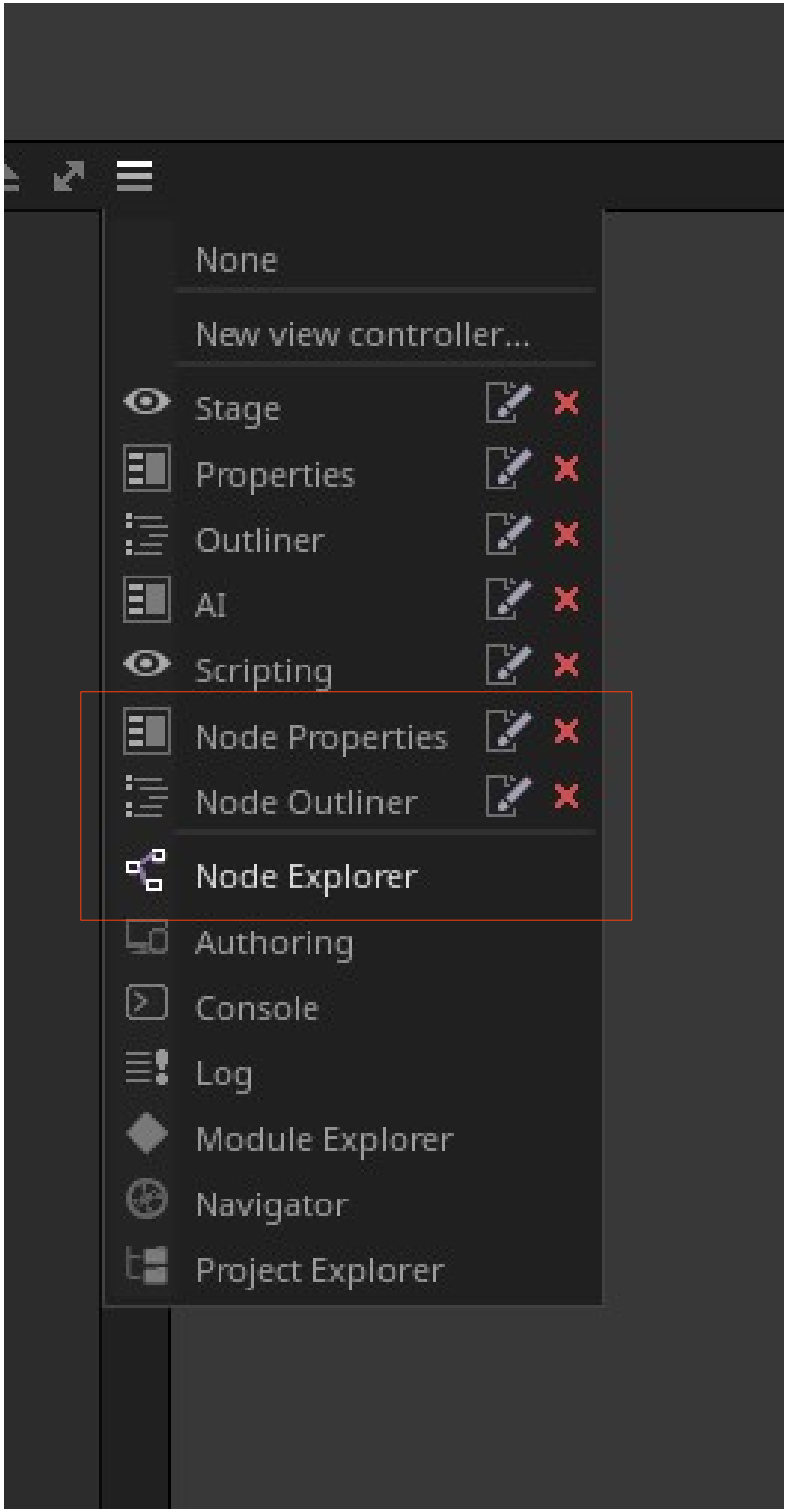
manual

The purpose of this module is data oriented editing as a property-node graph. By traversing this graph with logic you have the ability to (re-) generate things, like ShiVa-resources for example.

The design of the module adds the possibility to extend it with specific packages to add custom views/handling of nodes, which allows for specific higher level editing.

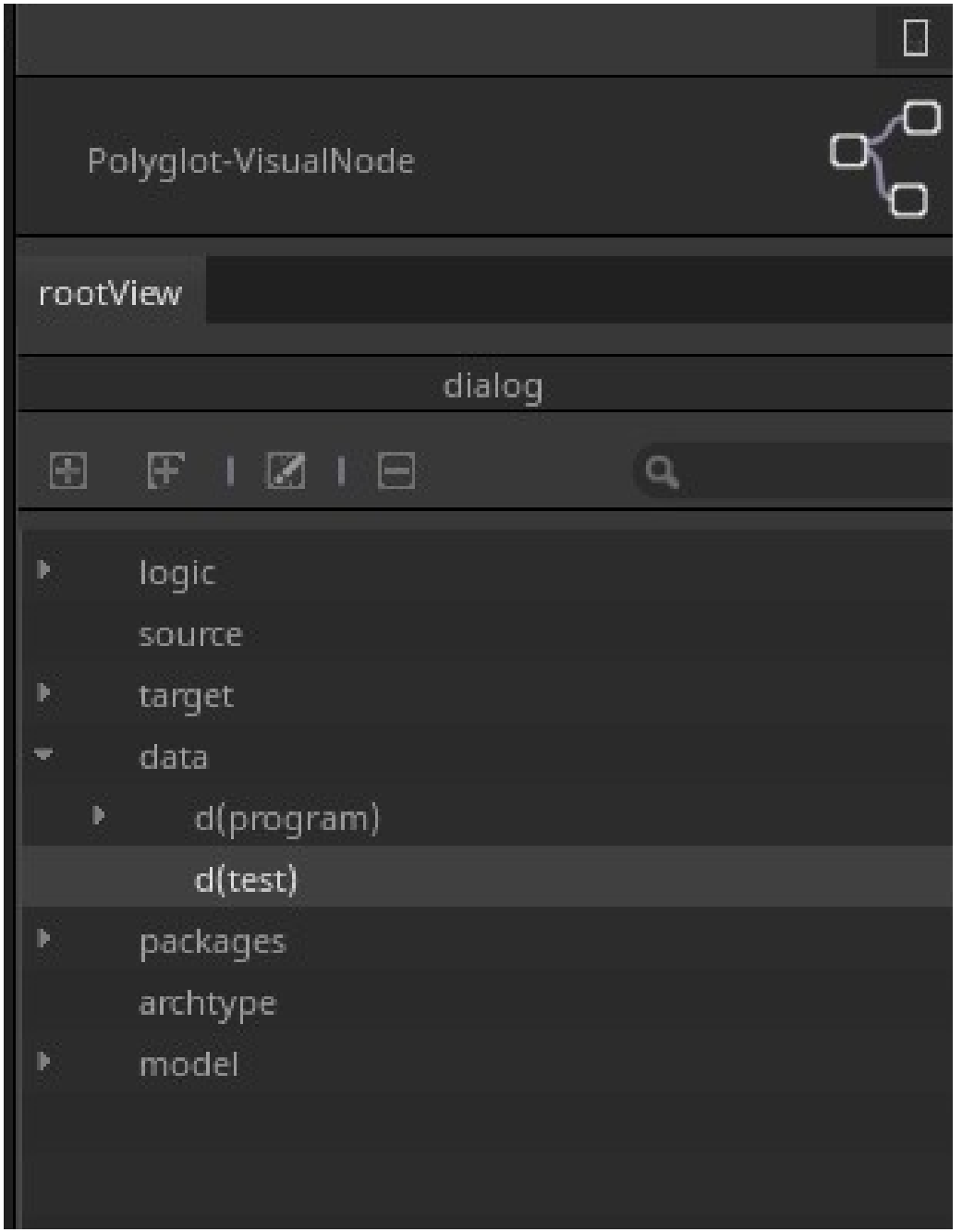
Views

- Node Explorer
- Node Properties
- Node Outliner
- Stage



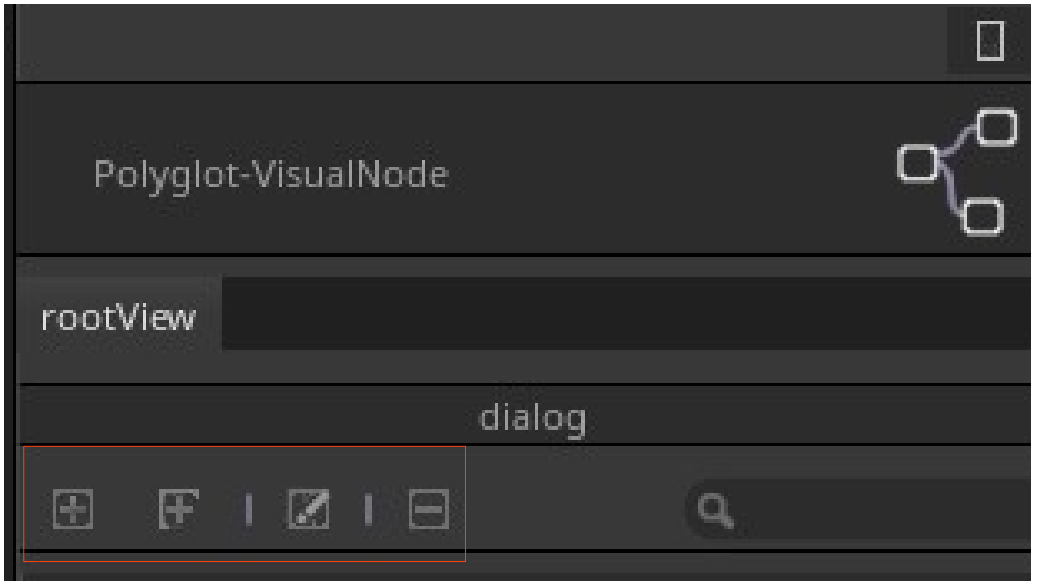
Node Explorer Tree

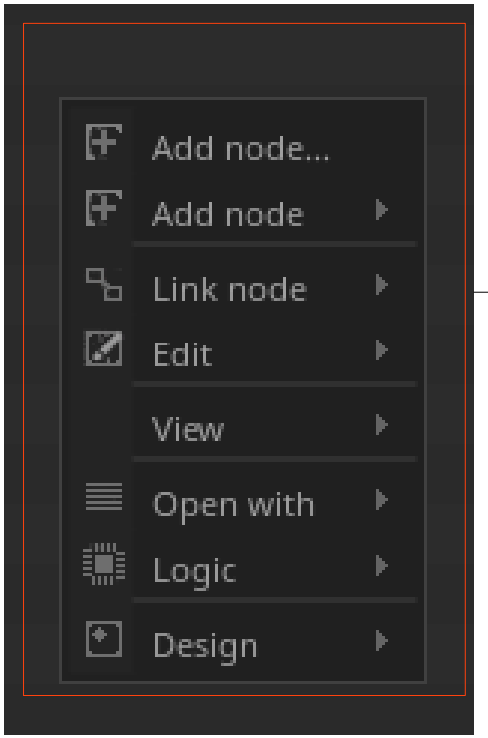
- Logic
  - Lua scripts
    - traversing nodes tree/linked
- Source
  - Definition of node types
- Target
  - Generated output
- Data
  - Storage for nodes
- Packages
  - VisualNode extensions
- Archtype
  - Bundle (of nodes, ..?)
- Model
  - Definition of nodes



Node Explorer Actions

- Add node
  - As sibling/child
  - Plus treeitem
- Link node
  - Between 2 selected nodes
- Copy & Paste Node
- Rename node
  - treeitem rename
- Delete treeitem
- Design
  - Node /Filter
  - Script /Export
- Package
  - (De)-Activated
- Logic
  - Run created lua scripts



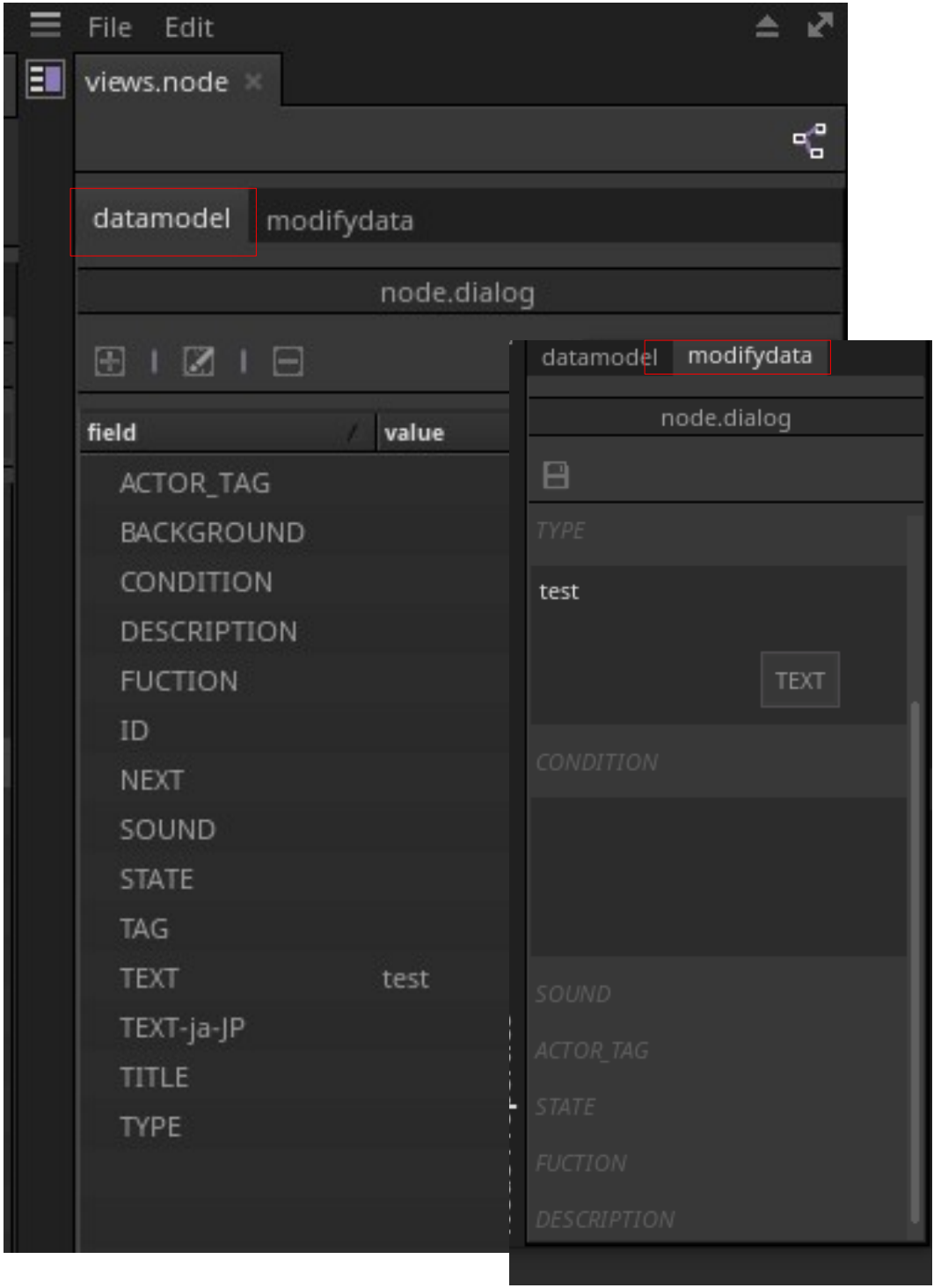


Treemenu

Node Properties

- type node is
  - e.g “node.dialog”
- datamodel

  Key value pairs
  - Create
  - Edit
  - Delete
- modifydata
  - Edit values

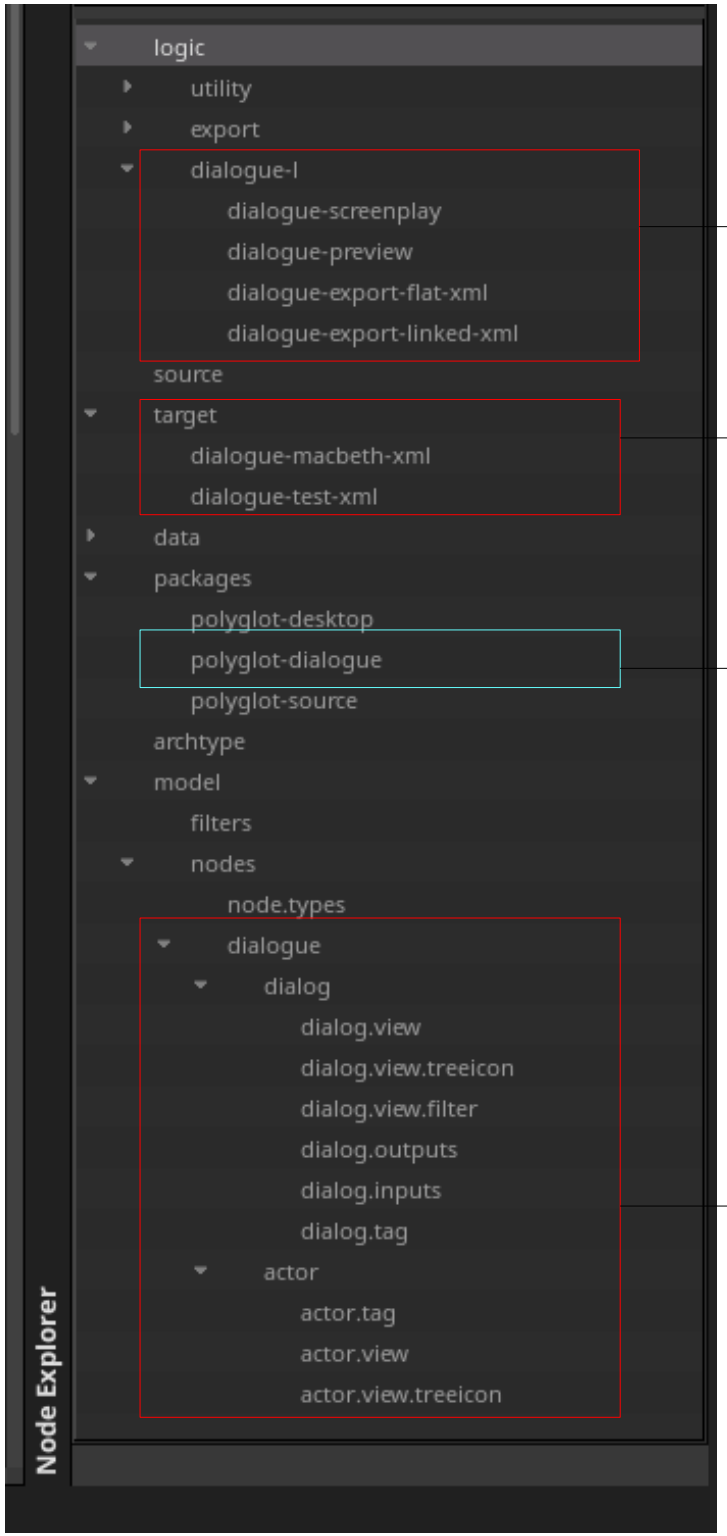


ShiVa module

polyglot-visualnode

Package Dialogue
(early stage)

Package Dialogue



Logic To
Export to xml
Preview export

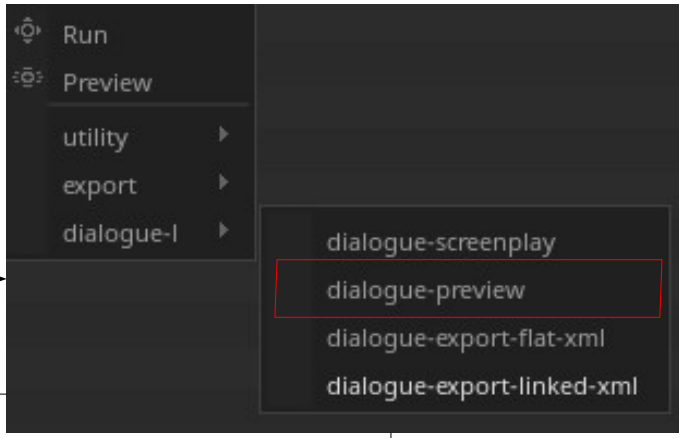
Exported data

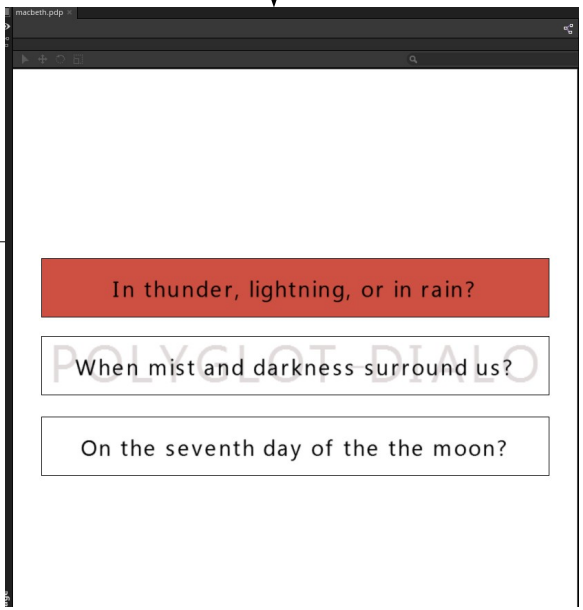
Treemenu

this
packag
e

Stage “Preview”
“.pdp extension”

Node definitions





ShiVa module

Polyglot-visualnode

Technical 'lua impl.'

Node

- Storage lua table in file
  - Node-definitions: have unique path -filename
  - Nodes: have content generated-id, same-content == same-id and generated-id == filename. no path
- Type def
  - Dot separated string: 'node[“x.y.z”]'
- Content
  - Key value pairs
- Content extended functionality
  - key value pairs
    - e.g. “.link” - <uid> to link 2 nodes

```
node["dialog"] =
{
["TITLE"] =
""
,
["NEXT"] =
""
,
["DESCRIPTION"] =
""
,
["TEXT"] =
"program"
,
}
```

Node Api

- *addNode*

snodeId = function CLP_Node.AddNode(self, sNodename, tNode, sNodeId, sRelativePath)

- *getNode*

snodeId = function CLP_Node.GetNode(self, sNodeId, sRelativePath)

- *getDifferenceFrom*

tKeyFields = CLP_Node.getDifferenceFrom(self, sNodeBaseFields, sOtherNodeFields)

- *isGenNode*

bGenNode = function CLP_Node.isGenNode(self, sNodeId, sNodename, tNodeData)

- Not yet implemented

function CLP_Node.intersection(self, sNodeId, sOtherNodeId)

function CLP_Node.complement(self, sNodeId, sOtherNodeId)

function CLP_Node.union(self, sNodeId, sOtherNodeId, kOptions)

function CLP_Node.difference(self, sNodeId, sOtherNodeId)

Node Traverse Api

- *walk(hTreeRoot, call, isRoot) --tree*
  - *You walk the tree explorer treeitems from the selected or every item*
- *walkLinked(hTreeRoot, call, isRoot) --node*
  - *You walk nodes connected with each other.*
- *Not yet implemented*
  - *other means of walking the graph*
    - *Visitor*
    - *..?*

Node Model Api

- *getDatamodel()*
  - *Gets all model definition to create a node from definition*
- *getSharedDatamodel()*
  - *Shared node storage*

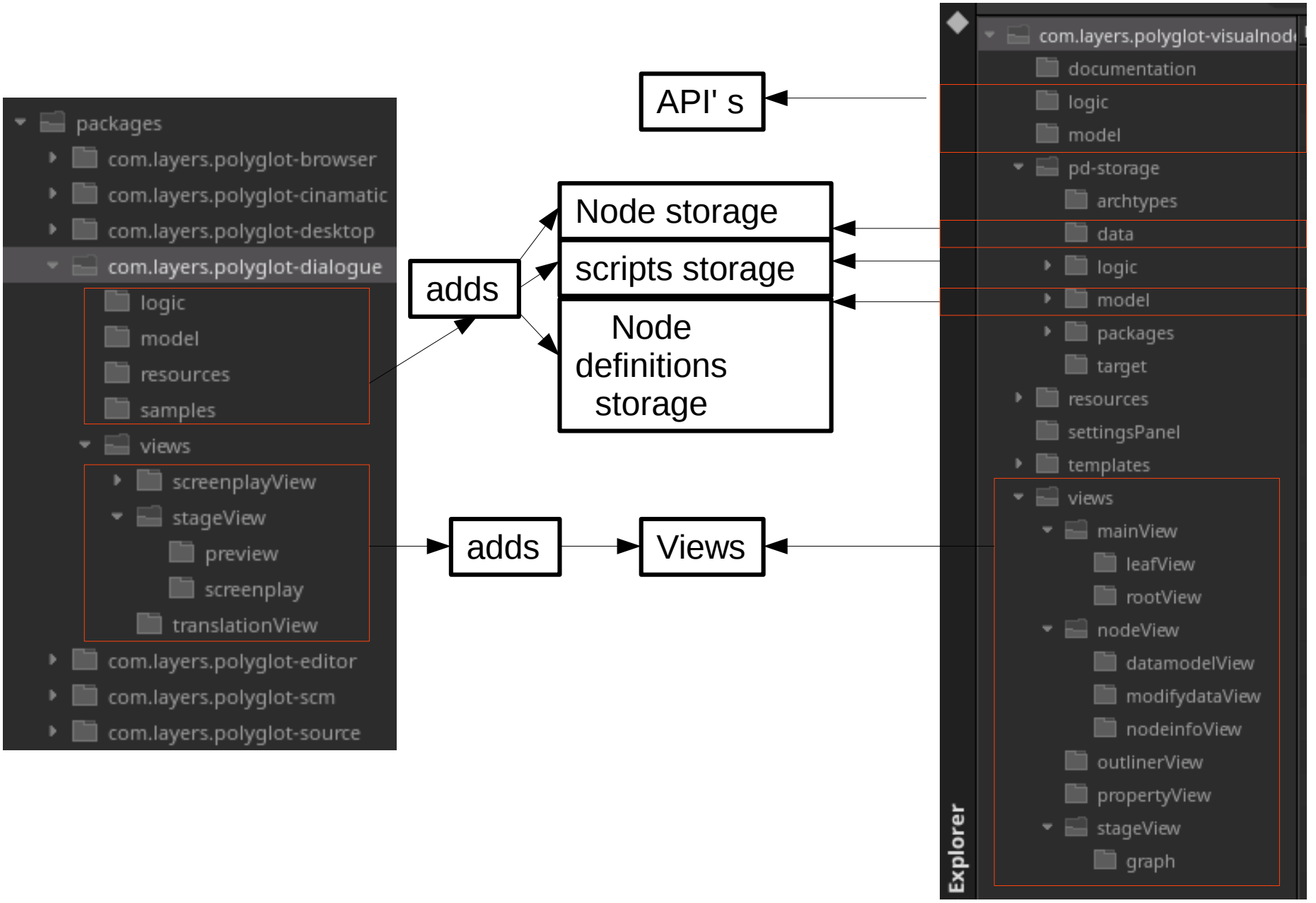
ShiVa module

polyglot-visualnode

Technical 'module'

Package structure

Module structure



- **Kown (high priority) issues**
- *“Node Definition” getting saved as node (generated-ids)*
- *Linked Nodes are not updated after editing a node*
- **Todo**
- *Rename node["dialog"] = {} → axiom[“node.dialog”] or other-name n/d/a/?*
- *Make the node API' s more like ShiVa API*
- *Do something use full with Stage view*
- *Cleanup Stage view code*
- *Tree (RootView/LeafView) is random saved (the order of keys can change with the same content)*
- *Tree (RootView/LeafView) is stored in custom format*
- *Tree (RootView/LeafView) are central and highly coupled*
- *The script/export templates are not user friendly*
- *...*