NEURO4J.ORG

Hello, World!

PDF version available online at http://static.neuro4j.org/download/doc/studio/HelloWorld.pdf

ABOUT THIS GUIDE

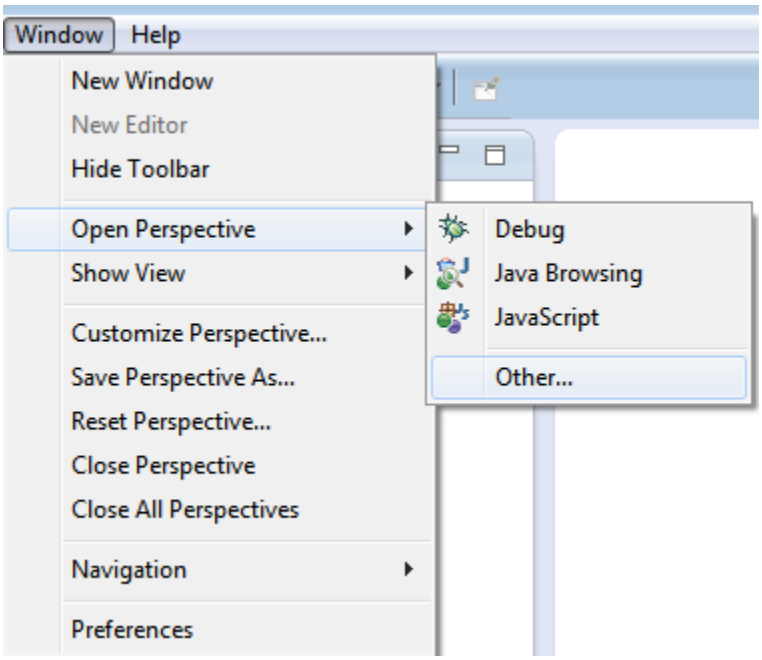
This guide describes how to create first flow with Neuro4j Studio.

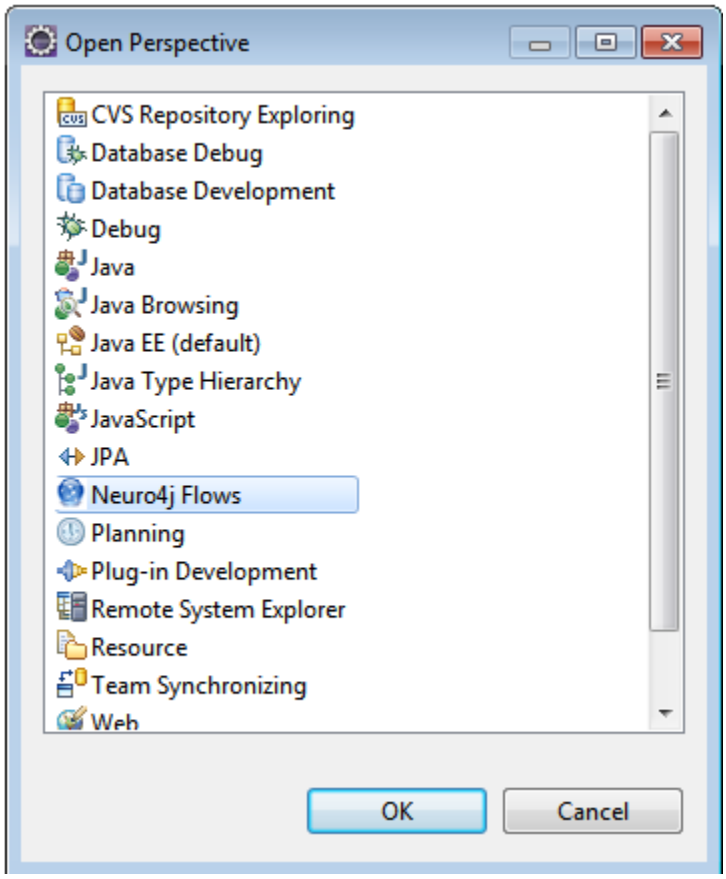
To install Neuro4j Studio use document http://static.neuro4j.org/download/doc/studio/StudioInstalationGuide.pdf

Online html version available at http://neuro4j.org/docs/wf/helloworld

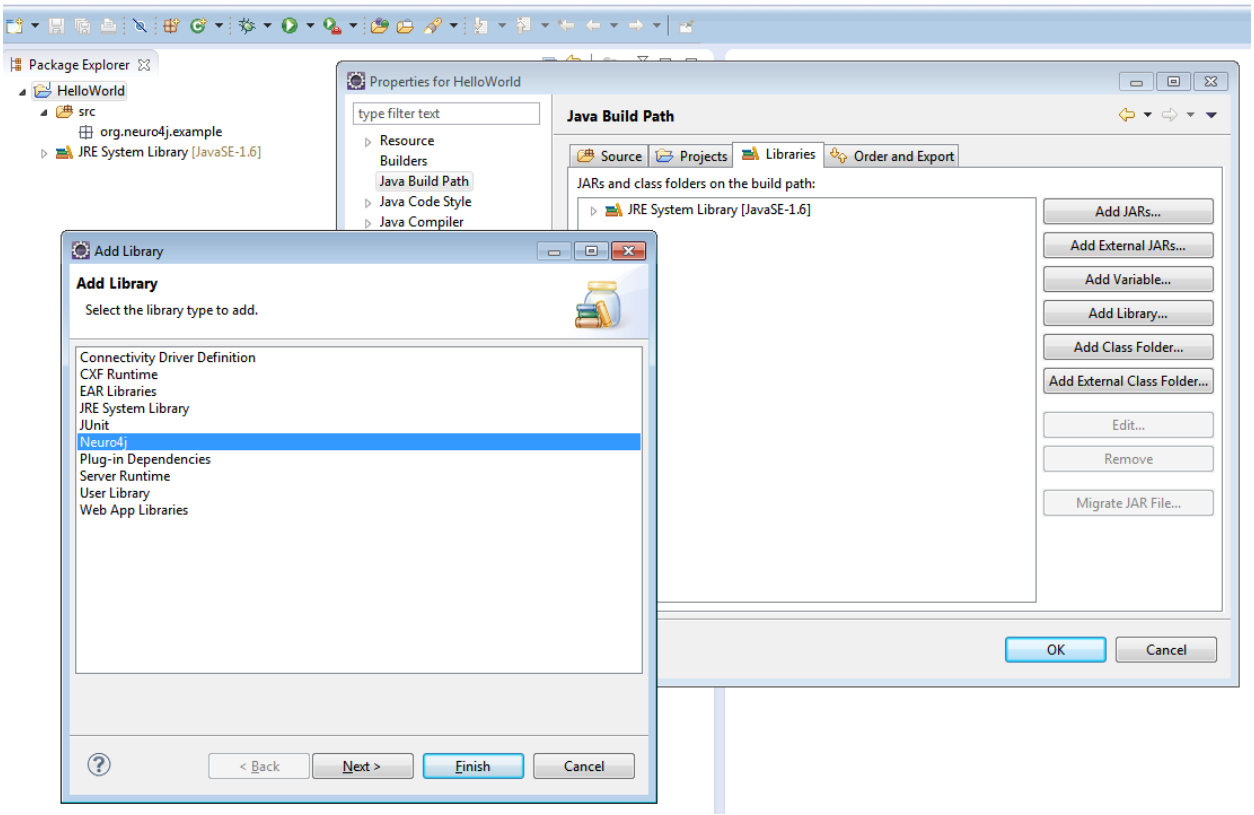
STEPS

1) Run Eclipse and create new “Java project”;
2) Switch to “Neuro4j Flows” perspective;

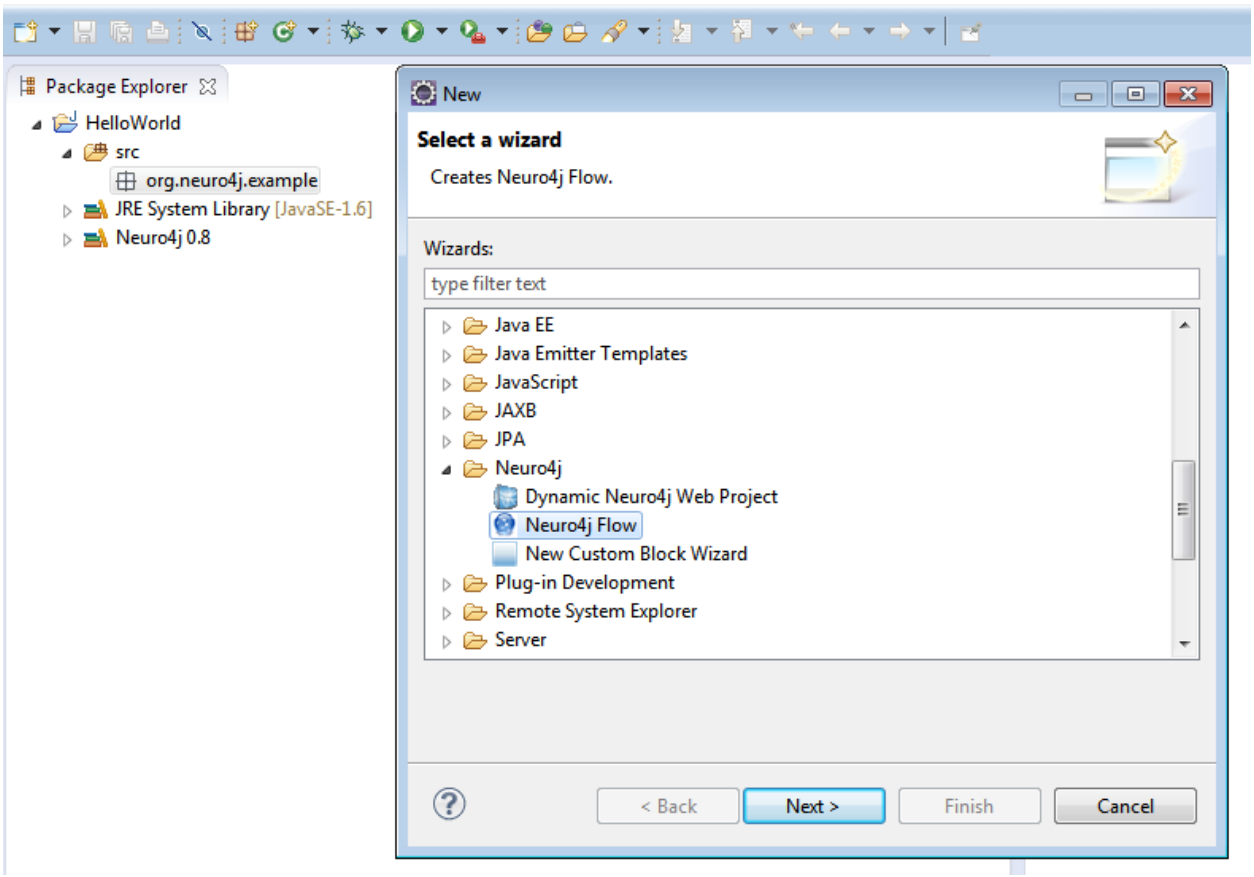




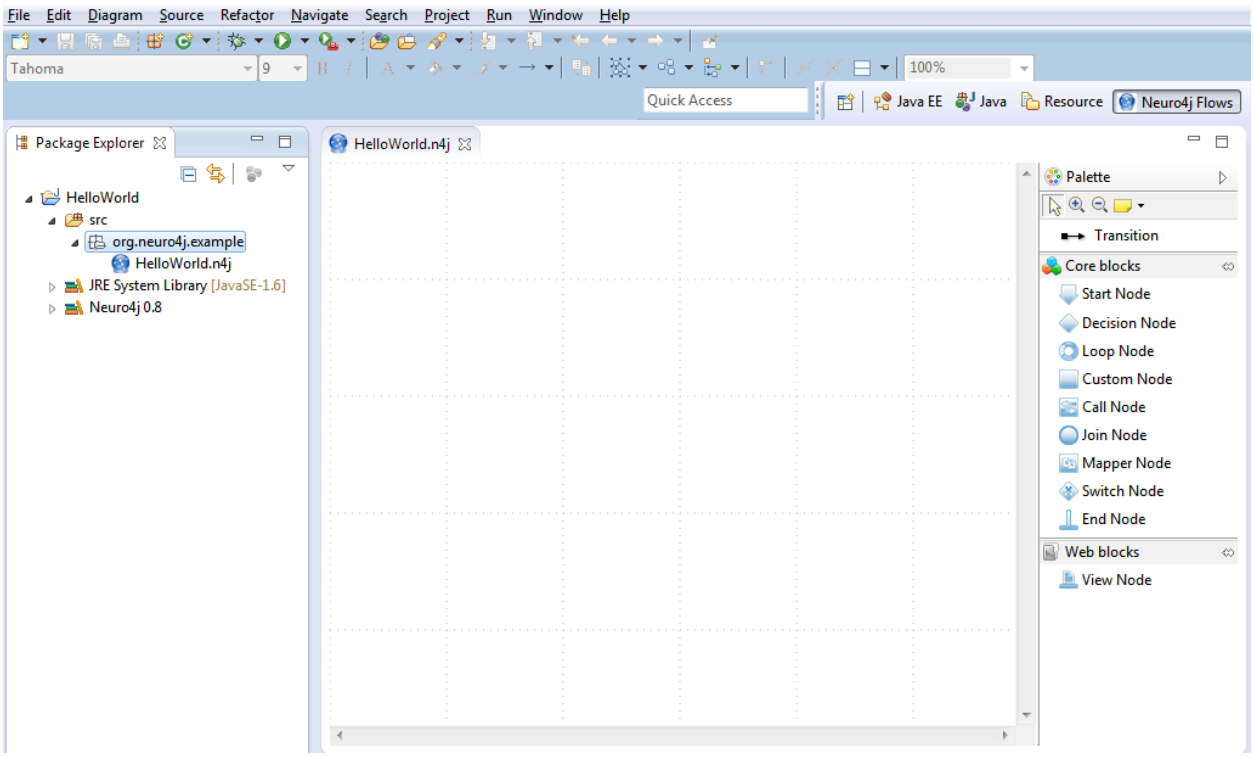
3) Create new package “org.neuro4j.example”
4) Add Neuro4j Library to project;



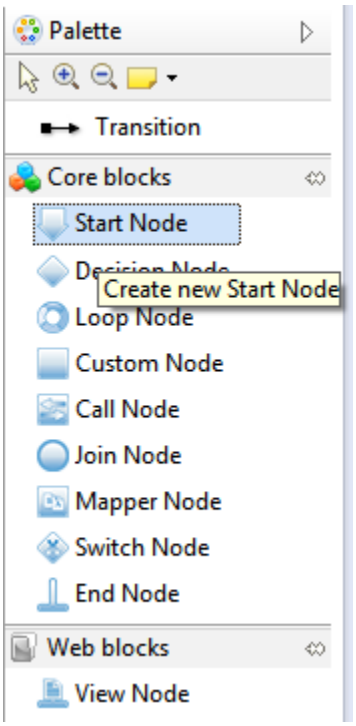
5) Select package “org.neuro4j.example”->Right-Click->New->Other
6) Select “Neuro4j Flow” in Category “Neuro4j” and click “Next”



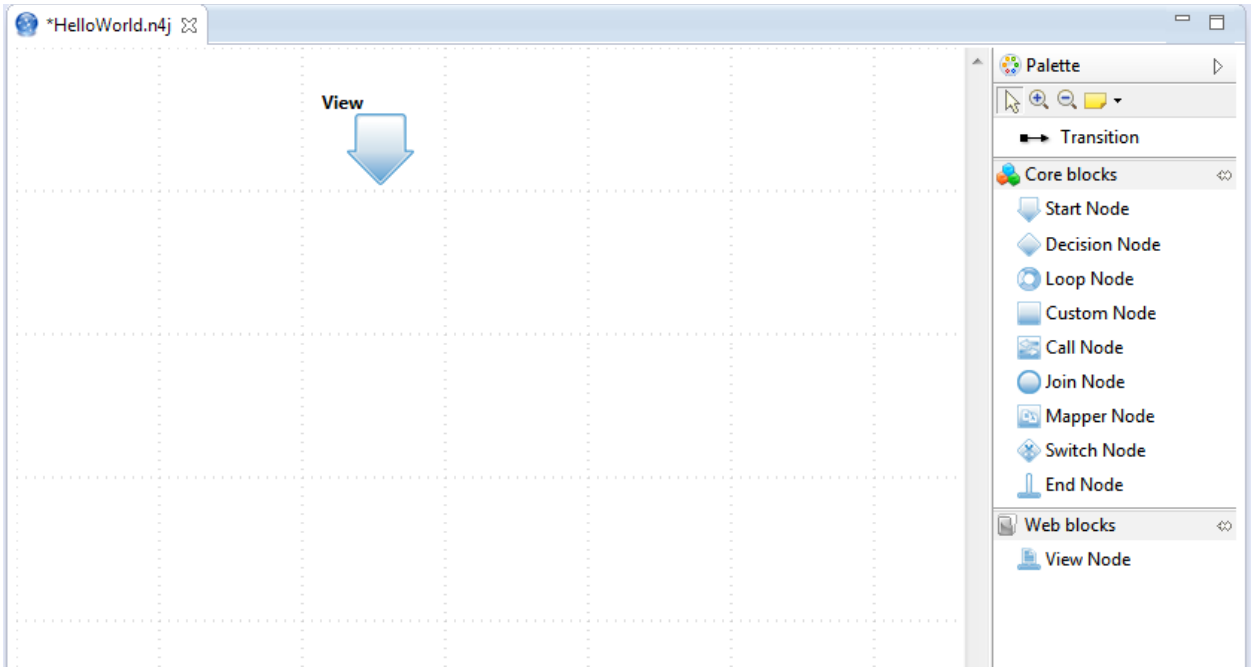
7) Put name “HelloWorld.n4j” and click “Finish”



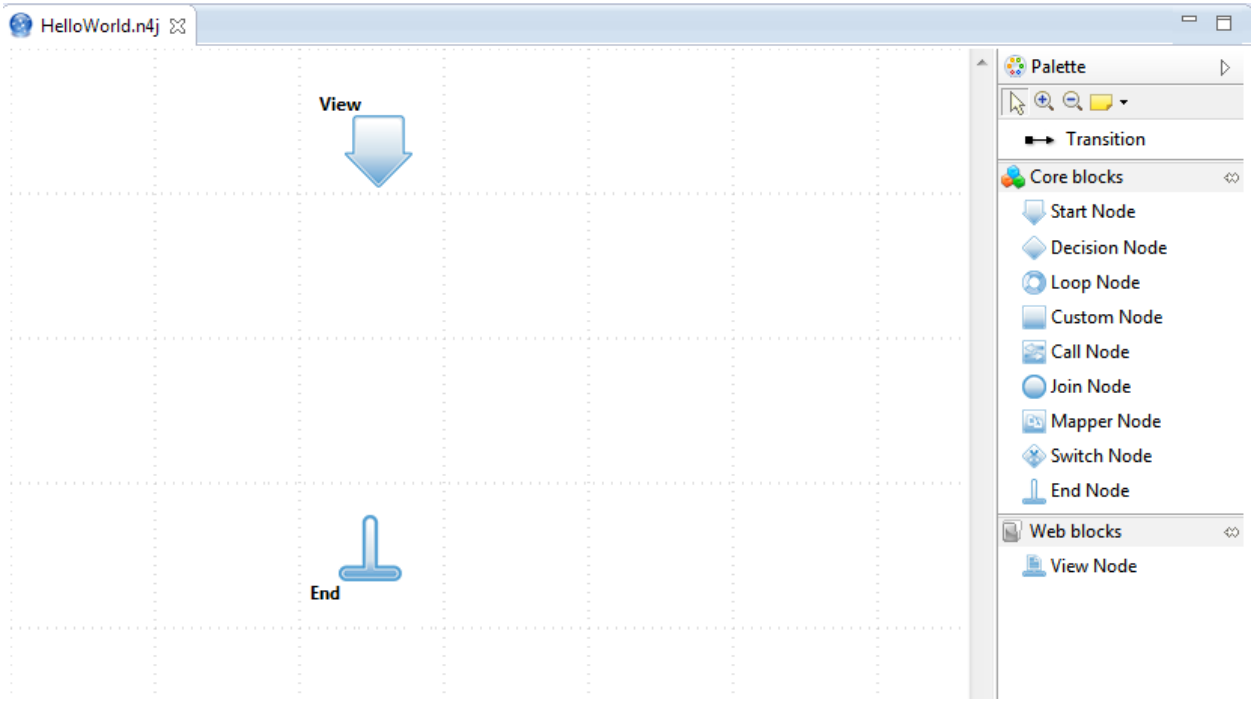
8) Click on Start Node



9) Click on Diagram and Set name – "View".

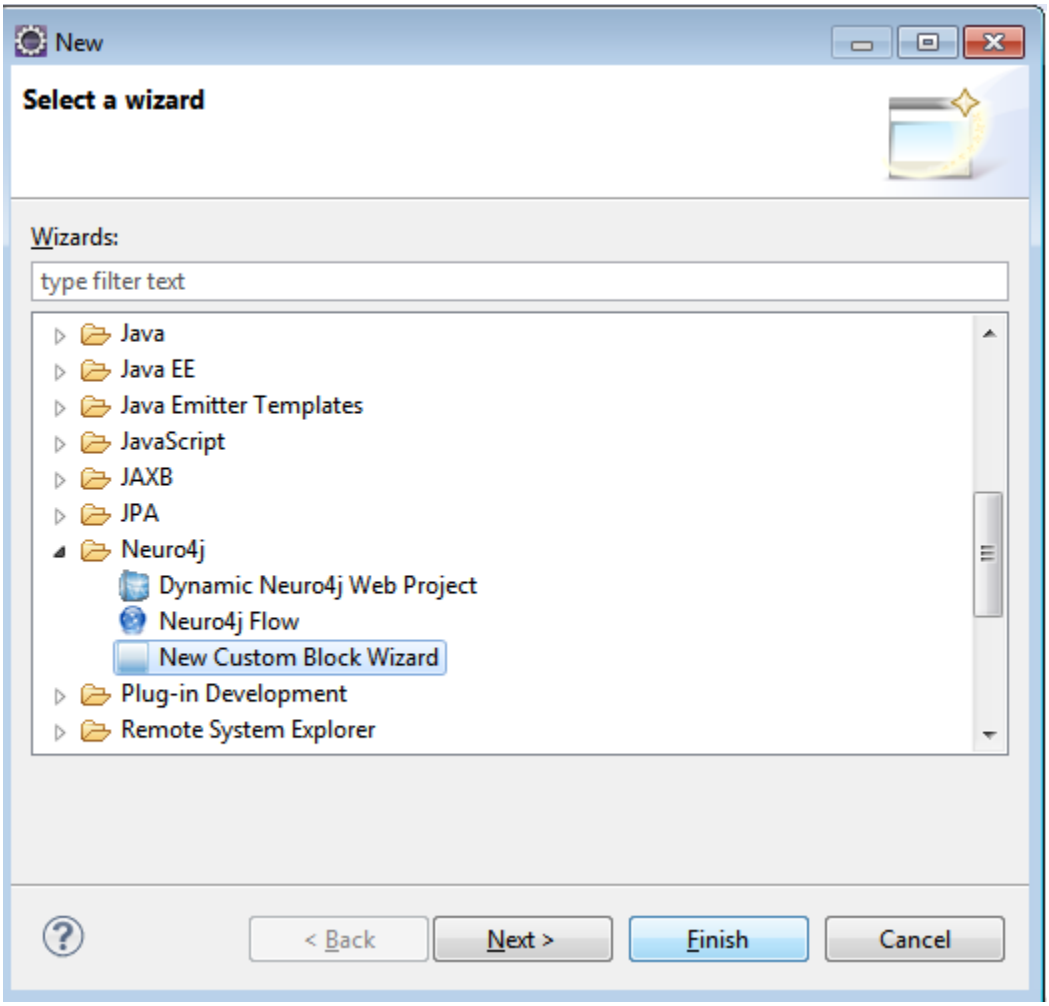


10) Add “End Node” to Diagram;

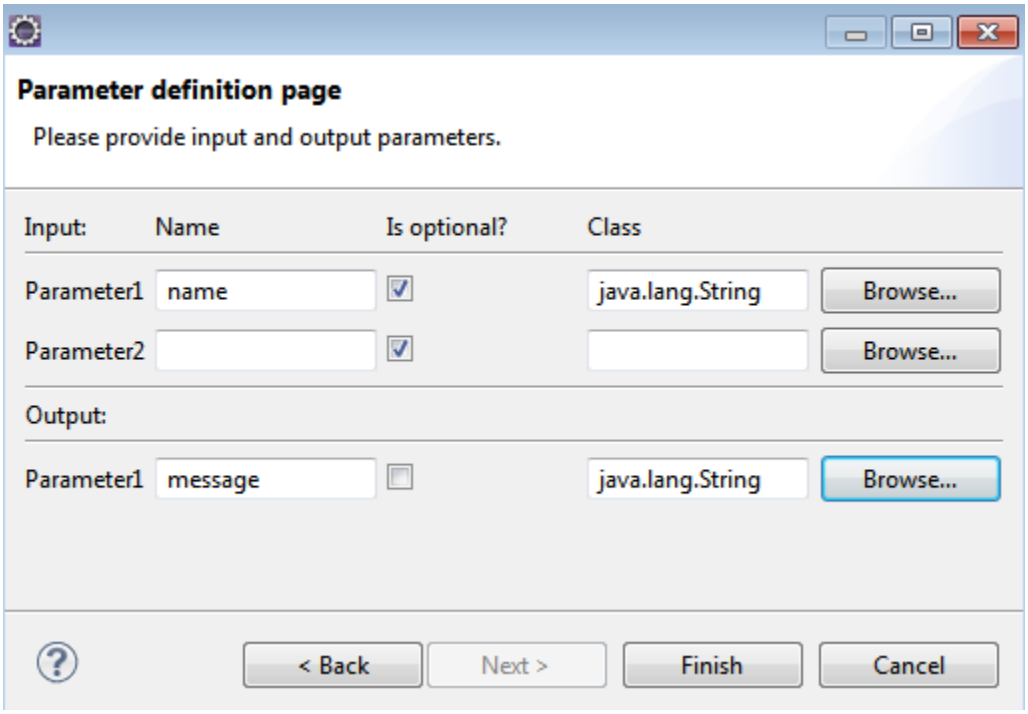


CUSTOM BLOCK

11) Select package “org.neuro4j.example”-> Right-Click->New->Other->Neiro4j->”New Custom Block Wizard”



12) Select package – enter name “HelloBlock.java” -> click “Next”;
13) Create 1 optional input parameter “name” with type “java.lang.String” and mandatory output parameter “message” with type “java.lang.String”;



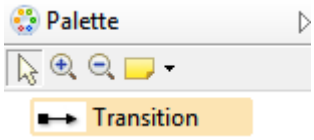
14) Click “Finish”;
15) New java file has been created;

```
@ParameterDefinitionList(input={
                @ParameterDefinition(name=IN_NAME, isOptional=true, type= "java.lang.String")},
        output={
                @ParameterDefinition(name=OUT_MESSAGE, isOptional=false, type= "java.lang.String")})
public class HelloBlock extends CustomBlock {

    static final String IN_NAME = "name";

    static final String OUT_MESSAGE = "message";


    @Override
    public int execute(LogicContext ctx) throws FlowExecutionException {

        Object name = ctx.get(IN_NAME);

        //TODO: put your code here

        if (/*error != */false)
        {
            return ERROR;
        }

        ctx.put(OUT_MESSAGE, null);

        return NEXT;
    }

    @Override
    protected void init() throws FlowInitializationException{
        super.init();
    }

}
```

16) Update method *execute*:

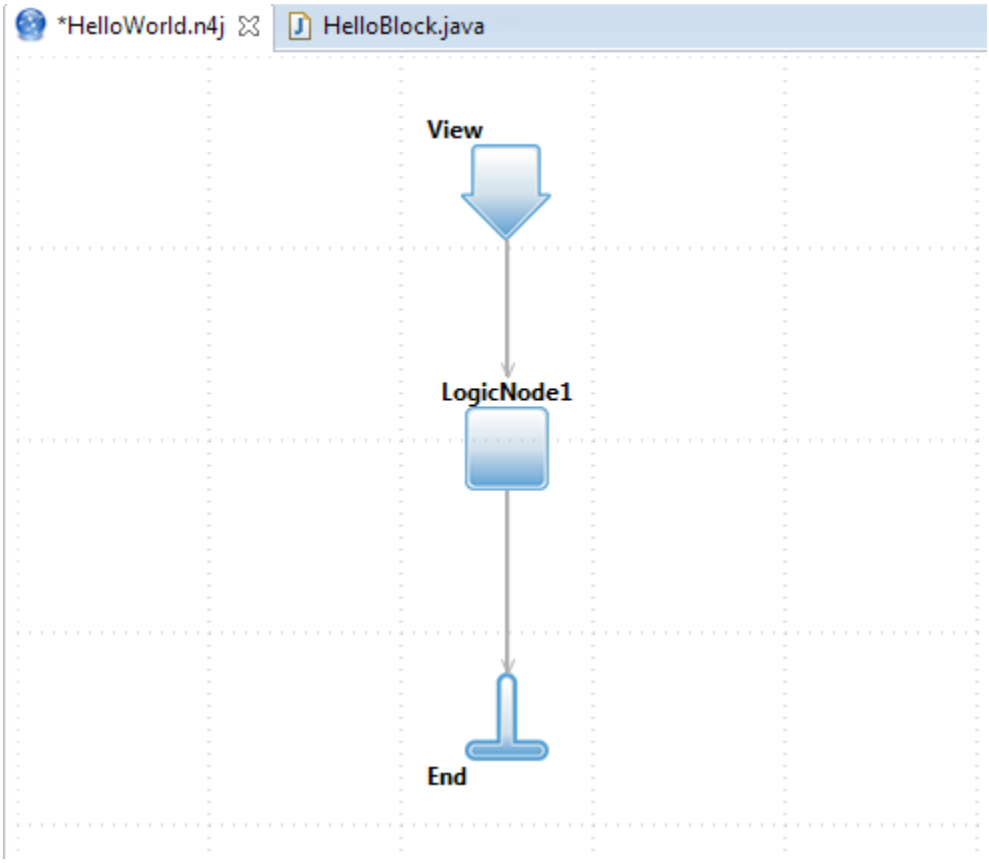
```
@Override
public int execute(LogicContext ctx)   throws FlowExecutionException {

        Object name = ctx.get(IN_NAME);

        String myMessage = "Hello ";

        if(name != null)
        {
                myMessage += name;
        }

        ctx.put(OUT_MESSAGE, myMessage);


return NEXT;

 }
```

17) Open Diagram and create “Custom Node”

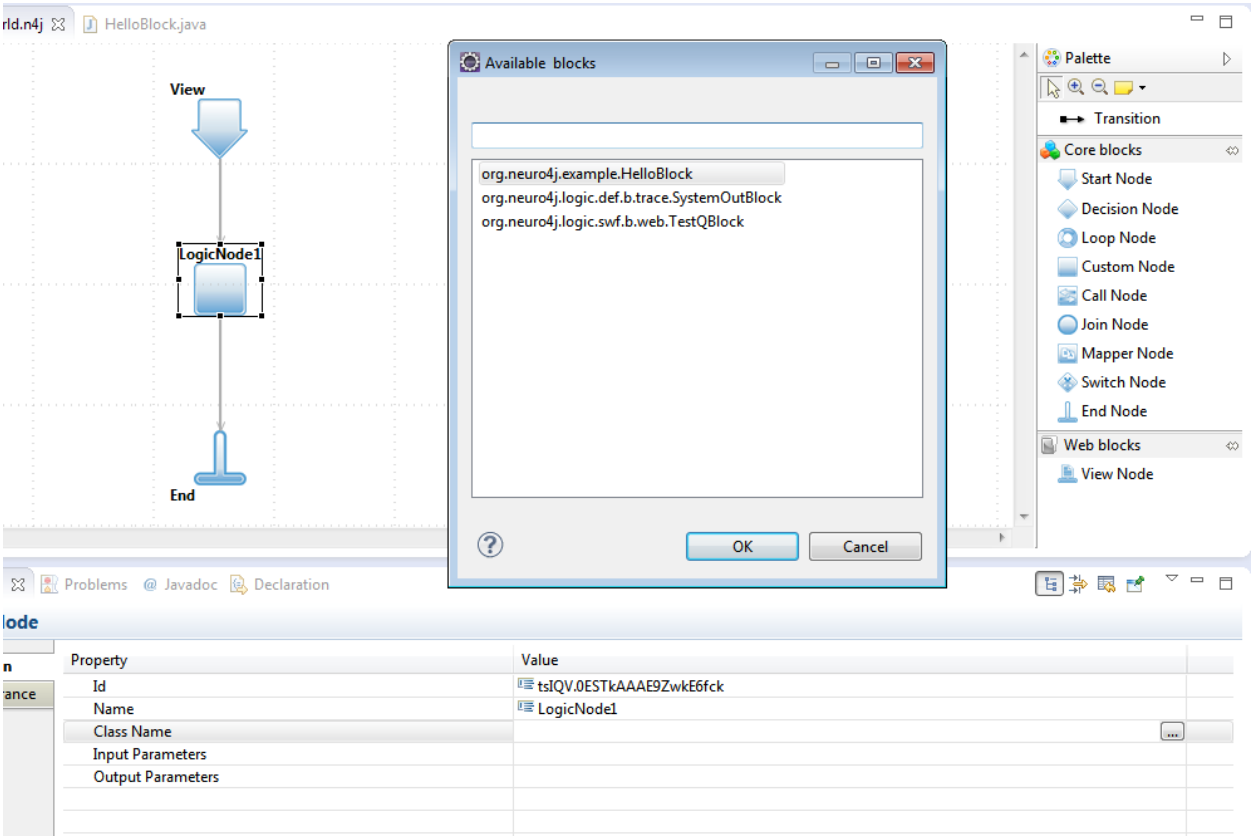


18) Click on “Transition”

19) Make connection between StartNode and CustomNode; CustomNode and EndNode;



20) Select on CustomNode and in Properties-View select “Class Name”



21) Select “HelloBlock” and click “Ok”;

RUN FLOW

22) Create java class with main-method;
23) Put add following code;

```
public static void main(String[] args)
{

    Map<String, Object> params = new HashMap<String, Object>();

    // name of input parameter
    params.put("name", "World");

    LogicContext logicContext = null;
    try {

        logicContext = SimpleWorkflowEngine.run("org.neuro4j.example.HelloWorld-View", params);

    // name of output parameter
    String message = (String)logicContext.get("message");
    System.out.println(message);

    } catch (FlowExecutionException e) {
```

```
            e.printStackTrace();
        }
    }
```

24) Run class with main method – Message “Hello World” should be printed in Console;