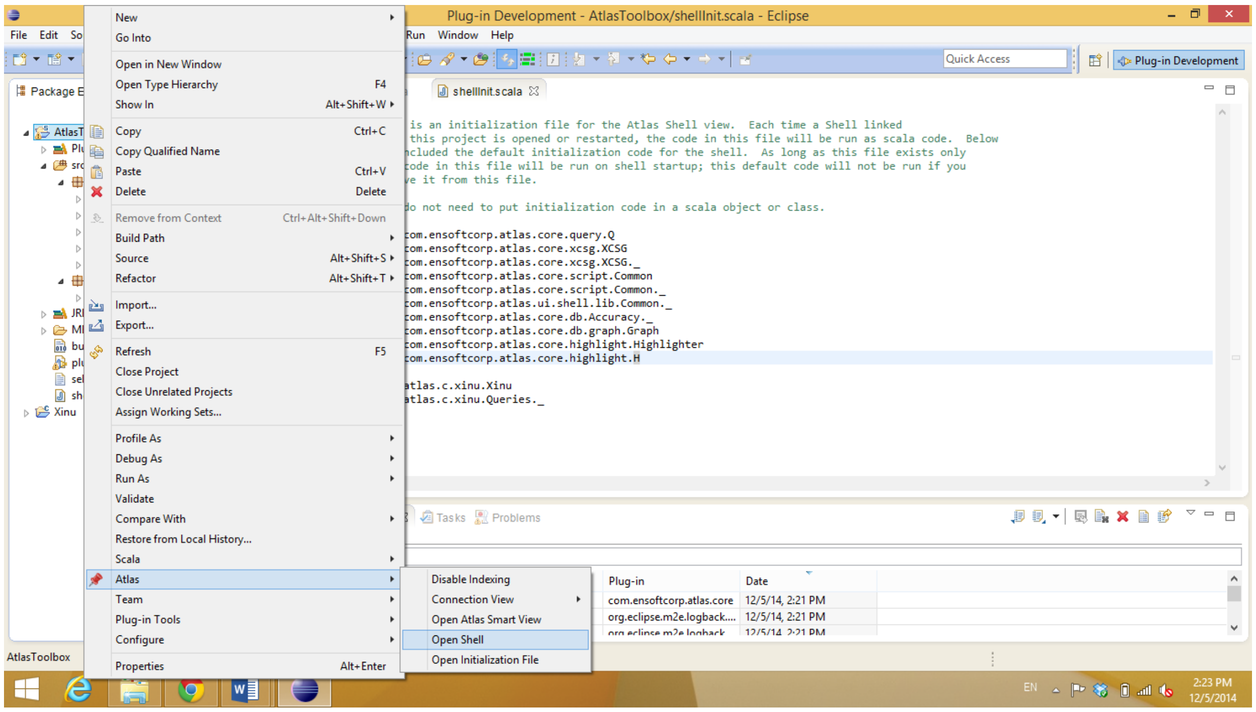
Queries.java - User Manual

How to begin writing queries?

- Right Click on (Atlas Toolbox) workspace -> Atlas -> Open Shell



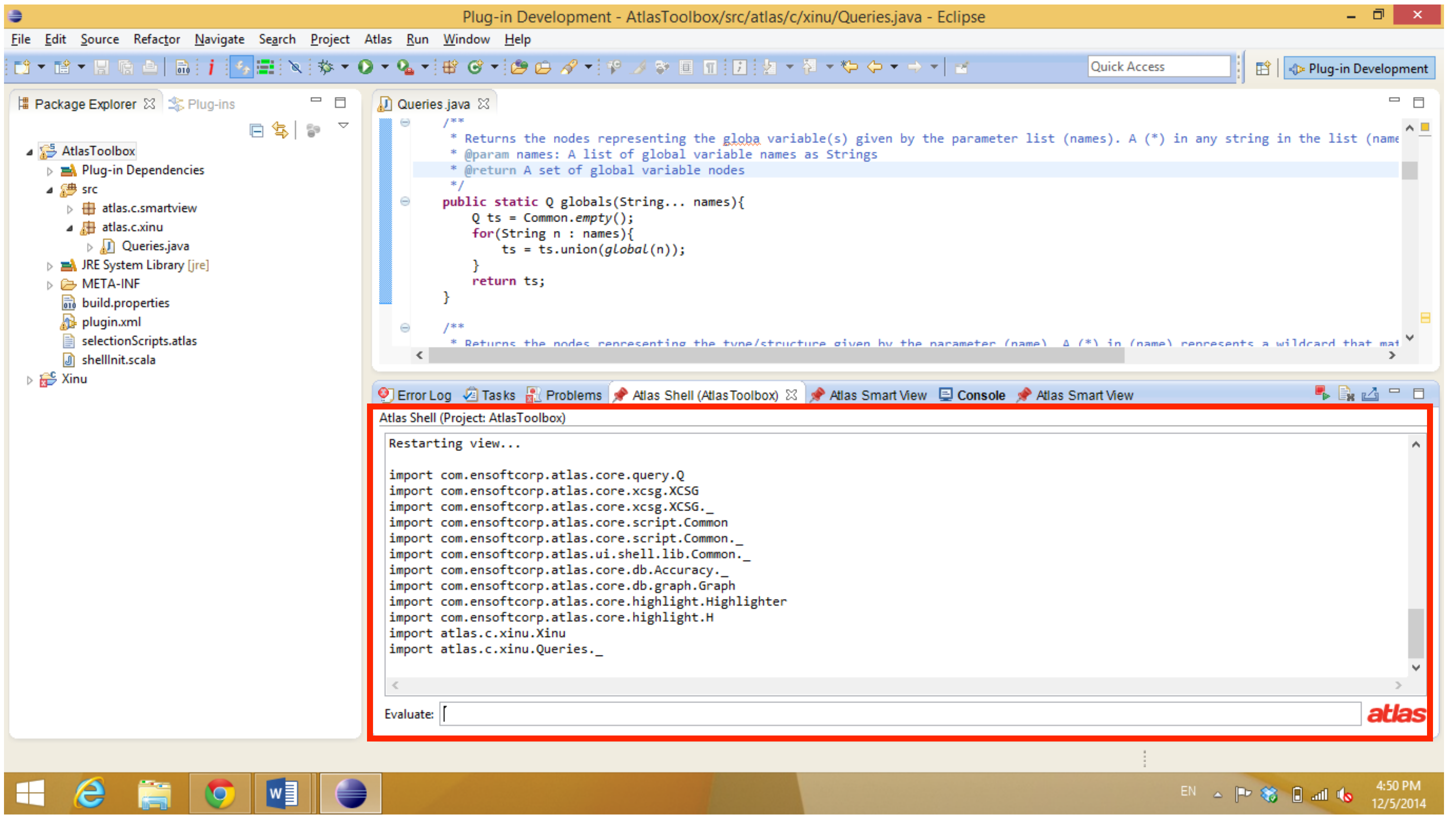
```
is an initialization file for the Atlas Shell view.  Each time a Shell linked
 this project is opened or restarted, the code in this file will be run as scala code.  Below
ncluded the default initialization code for the shell.  As long as this file exists only
code in this file will be run on shell startup; this default code will not be run if you
ve it from this file.

do not need to put initialization code in a scala object or class.

com.ensoftcorp.atlas.core.query.Q
com.ensoftcorp.atlas.core.xcsg.XCSG
com.ensoftcorp.atlas.core.xcsg.XCSG._
com.ensoftcorp.atlas.core.script.Common
com.ensoftcorp.atlas.core.script.Common._
com.ensoftcorp.atlas.ui.shell.lib.Common._
com.ensoftcorp.atlas.core.db.Accuracy._
com.ensoftcorp.atlas.core.db.graph.Graph
com.ensoftcorp.atlas.core.highlight.Highlighter
com.ensoftcorp.atlas.core.highlight.H

atlas.c.xinu.Xinu
atlas.c.xinu.Queries._
```

- Now, you can write your queries in the shell box highlighted below:



```
/**
 * Returns the nodes representing the globa variable(s) given by the parameter list (names). A (*) in any string in the list (name
 * @param names: A list of global variable names as Strings
 * @return A set of global variable nodes
 */
public static Q globals(String... names){
    Q ts = Common.empty();
    for(String n : names){
        ts = ts.union(global(n));
    }
    return ts;
}

/**
 * Returns the nodes representing the type/structure given by the parameter (name). A (*) in (name) represents a wildcard that mat
```

Atlas Shell (Project: AtlasToolbox)

```
Restarting view...

import com.ensoftcorp.atlas.core.query.Q
import com.ensoftcorp.atlas.core.xcsg.XCSG
import com.ensoftcorp.atlas.core.xcsg.XCSG._
import com.ensoftcorp.atlas.core.script.Common
import com.ensoftcorp.atlas.core.script.Common._
import com.ensoftcorp.atlas.ui.shell.lib.Common._
import com.ensoftcorp.atlas.core.db.Accuracy._
import com.ensoftcorp.atlas.core.db.graph.Graph
import com.ensoftcorp.atlas.core.highlight.Highlighter
import com.ensoftcorp.atlas.core.highlight.H
import atlas.c.xinu.Xinu
import atlas.c.xinu.Queries._
```

Evaluate:

Queries:

Function	`functions`
Parameters	`Parameters (functionNames): A list of function names as Strings`
Description	Returns the set of functions where their names matches the any of the names given (`functionNames`) list. A (*) in (`functionNames`) represents a wildcard that matches any string.

Example (1)	*Return the functions named "dswrite" and "dsread"*
	`var funcs = functions("dswrite", "dsread") ↵`
	`show(funcs) ↵`

Example (2)	*Return all functions where their names start with/match "ds*" or "dg*"*
	`var funcs = functions("ds*", "dg*") ↵`
	`show(funcs) ↵`

Function	`globals`
Parameters	`Parameter (names): A list of global variable names as Strings`
Description	Returns the nodes representing the global variables given by the parameter (`name`). A (*) in (`name`) represents a wildcard that matches any string.

Example (1)	*Return the global variable named "devtab"*
	`var globalVar = global("devtab") ↵`
	`show(globalVar) ↵`

Example (2)	*Return all global variables where their names start with/match "dv*"*
	`var globalVars = global("dv*") ↵`
	`show(globalVars) ↵`

Function	`types`
Parameters	`Parameter (names): A list of type names as Strings`
Description	Returns the nodes representing the types/structures given by the parameter list (`names`). A (*) in any string in the list (`names`) represents a wildcard that matches any string.

Example (1)	*Return all the types/structures named "dreq", "epacket"*
	`var ts = types("dreq", "epacket") ↵`
	`show(ts) ↵`

Example (2)	*Return all the type/structures where their names start with/match "d*", or "e*"* `show(types("d*", "e*")) ↵`

Function	`ref`
Parameters	`Parameter (object): the set of global variables and/or types`
Description	Returns the set of functions referencing (read/write) the given global variables and/or types (structures) given in parameter (`object`).

Example	*Return all functions referencing the structures/types "dreq", "epacket"*
	`var ts = types("dreq", "epacket") ↵`
	`var refFuncs = ref(ts) ↵`
	`show(refFuncs) ↵`

Function	`cfg`
Parameters	`Parameter (funcName): function name as a String`
Description	Returns the control-flow graph (CFG) of the given function name in (`funcName`)

Example	*Returns the CFG of function "dswrite"*
	`var dswriteCFG = cfg("dswrite") ↵`
	`show(dswriteCFG) ↵`

Function	`cfg`
Parameters	`Parameter (funcs): function node(s)`
Description	Returns the control-flow graph (CFG) of the given function node in (`funcs`)
Example (1)	*Returns the CFG of function "dswrite"* `var dswrite = function("dswrite")` ↵ `var dswriteCFG = cfg(dswrite)` ↵ `show(dswriteCFG)` ↵
Example (2)	*Get the CFG of the function node selected from a previously produced graph* `var result= cfg(selected)` ↵ `show(result)` ↵

Function	`cg`
Parameters	`Parameter (funcs): function node(s)`
Description	Returns the call graph of the given function(s) in (`funcs`)
Example (1)	*Returns the Call Graph of function "dswrite"* `var dswrite = function("dswrite")` ↵ `var callgraph = cg(dswrite)` ↵ `show(callgraph)` ↵
Example (2)	*Get the call graph of a function node selected from a previously produced graph* `var result= cg(selected)` ↵ `show(result)` ↵

Function	`rcg`
Parameters	`Parameter (funcs): function node(s)`
Description	Returns the reverse-call graph of the given function(s) in (`funcs`)
Example	*Returns the Reverse-Call Graph of function "freebuf"* `var func = function("freebuf")` ↵ `var x = rcg(func)` ↵ `show(x)` ↵
Example (2)	*Get the reverse-call graph of a function node selected from a previously produced graph* `var result= rcg(selected)` ↵ `show(result)` ↵

Function	`call`
Parameters	`Parameter (funcs): function node(s)`
Description	Returns the direct callers of the given function (s) in (`funcs`)
Example	*Returns the direct callers of function "getbuf" and "freebuf"* `var funcs = functions("freebuf", "getbuf")` ↵ `var callers = call(funcs)` ↵ `show(callers)` ↵

Function	`calledby`
Parameters	`Parameter (funcs): function node(s)`
Description	Returns the set of functions directly called by the given function (s) in (`funcs`)
Example	*Returns the functions directly called by "dswrite"* `var func = function("dswrite")` ↵ `var callees = calledby(func)` ↵ `show(callees)` ↵

Function	`induce`
Parameters	`Parameter (funcs): function node(s)`
Description	Induces a call edges between the given set of function(s) in (`funcs`)

Example	*Induce the call edges (if any) between functions “dswrite” and “dskenq”*
	`var funcs = functions(“dswrite”, “dskenq”) ↵`
	`var result = induce(funcs) ↵`
	`show(result) ↵`

Function	`inducecfg`
Parameters	`Parameter (nodes): control-flow node(s)`
Description	Induces the control-flow edges between the given control-flow blocks in (`nodes`)

Example	*Induce the control-flow edges (if any) between selected blocks*
	`show(cfg(function(“dswrite”))) ↵`
	Select a subset of control-flow blocks in the produced graph, the selection can be accessed through the variable (`selected`) *as follows:*
	`var result = inducecfg(selected) ↵`
	`show(result) ↵`

Function	`graph`

Parameters	`Parameter (roots): function node(s)`
	`Parameter (leaves): function node(s)`

Description	Returns the call graph between the functions in (`roots`) and functions in (`leaves`)

Example	*Return the call graph between “dswrite” as a root and “freebuf” as a leaf*
	`var callgraph = graph(function(“dswrite”), function(“freebuf”)) ↵`
	`show(callgraph) ↵`

Function	`mpg`

Parameters	`Parameter (e1Functions): L function node(s)`
	`Parameter (e2Functions): U function node(s)`
	`Parameter (object): type/structure`

Description	Returns the matching-pair graph (MPG) that reference the structure given in (`object`) w.r.t L functions given in (`e1Functions`) and U functions given in (`e2Functions`)

Example	*Get the MPG for type/structure “dreq” and L function “getbuf” and U function “freebuf”*
	`var result = mpg(function(“getbuf”), function(“freebuf”), type(“dreq”)) ↵`
	`show(result) ↵`

Function	`dfg`

Parameters	`Parameter (functionContext): caller of functionSource`
	`Parameter (functionSource): return value to use as origin in data-flow graph`
	`Parameter (functionSink): stopping point for data-flow graph`

Description	Returns the data-flow graph (inter-procedural forward slice) starting at the return value of Function “`functionSource`” in the calling function “functionContext”, stopping at Function “functionSink”.

Example	*Return the inter-procedural data-flow graph for the pointer allocated by “getbuf“ in function “dswrite”*
	`var result = dfg( “dswrite”, “getbuf”, “freebuf”) ↵`
	`show(result) ↵`

Function	`projectdfg`

Parameters	`Parameter (dfg): A previously computed inter-procedural data-flow graph`
	`Parameter (func): function node(s)`

Description	Returns the portion of data-flow graph within the given function (`func`). (aka intra-procedural data-flow graph)
Example	*Return the portion of DFG in function "dskenq" that is part of DFG for the allocated pointer by "getbuf" in function "dswrite"* `var result = dfg(function("dswrite"), function("getbuf"))` ↵ `var projection = projectdfg(result, function("dskenq"))` ↵ `show(projection)` ↵

Function	`forwardSlice`

Parameters	`Parameter (projdfg): Intra-procedural data-flow graph`
	`Parameter (node): the data-flow node where the forward slice starts`

Description	Returns the intra-procedural forward slice starting at the given (node) from the given data-flow graph in (`projdfg`)
Example	`var result = dfg(function("dswrite"), function("getbuf"))` ↵ `var projection = projectdfg(result, function("dskenq"))` ↵ `show(projection)` ↵ *Select the data-flow node of interest from the produced graph, the selection can be accessed through the variable* (`selected`) *as follows:* `var forwardSlice = forwardSlice(selected)` ↵ `show(forwardSlice)` ↵