Any Java class that implements the BioModule interface can be added to a BioLockJ pipeline.

The BioLockJ v1.0 implementation is currently focused on metagenomics analysis, but the generalized application framework is not limited to this domain. Users can implement new BioModules to automate a wide variety of bioinformatics and report analytics. The BioModule interface was designed so that users can develop new modules on their own.

Beginners

See the BioModule hello world tutorial.

Coding your module

To create a new BioModule, simply extend one of the abstract Java superclasses, code it's abstract methods, and add it to your pipeline with #BioModule tag your Config file:
1. BioModuleImpl: Extend if a more specific interface does not apply 1. ScriptModuleImpl: Extend if your module generates and executes bash scripts 1. JavaModuleImpl: Extend if your module only runs Java code 1. ClassifierModuleImpl: Extend to support a new classifier program 1. ParserModuleImpl: Extend to parse output of a new classifier program 1. R_Module: Extend if your module generates and executes R scripts

To support a new classifier, create 3 modules that implement the following interfaces:

1. ClassifierModule: Implement to generate bash scripts needed to call classifier program
2. ParserModule: Implement to parse classifier output, configure as classifier post-requisite
3. OtuNode: Classifier specific implementation holds OTU information for 1 sequence

BioModuleImpl is the top-level superclass for all modules.

Method	Description
checkDependencies()	**Must override.** Called before executeTask() to identify [[Configuration]] errors and perform runtime validations.
executeTask()	**Must override.** Executes core module logic.
cleanUp()	Called after executeTask() to run cleanup operations, update Config properties, etc.
getInputFiles()	Return previous module output.
getModuleDir()	Return module root directory.
getOutputDir()	Return module output directory.
getPostRequisiteModules()	Returns a list of BioModules to run after the current module.
getPreRequisiteModules()	Returns a list of BioModules to run before the current module.