Paxtools, a Java API for BioPAX
http://bit.ly/harmony2011paxtools

Paxtools: Features

- A complete and consistent implementation of BioPAX specification:
  - POJO Beans
  - Getters/Setters
- Handles non-OO OWL constructs
  - Subproperties, reverse properties, anonymous classes

Paxtools Feature:cntd

- 2 Levels of validation:
  - Fail-fast syntactic validation
  - “Post” validation: Complex rules and best practices
- Seamless handling of different BioPAX levels
- Efficient traversal and editing via reflection
- Converting to and from different formats
  - PSI-MI
  - GSEA, SIF, SBGN

Paxtools Feature:cntd

- Advanced Graph Queries
- Modular and lightweight structure
- A platform for development of BioPAX software infrastructure

Getting Started

- Wiki: http://sourceforge.net/apps/mediawiki/biopax/index.php?title=Paxtools
- Project documentation: http://biopax.sourceforge.net/paxtools
- Java Apidocs: http://biopax.sourceforge.net/paxtools/apidocs/index.html

Obtaining Paxtools

- Release bundles – “Fat” jars (with dependencies)
- http://sourceforge.net/projects/biopax/files/paxtools/
- Individual modules (compiled JARs and POMs) can be found in the BioPAX maven repositories: http://biopax.sf.net/m2repo/snapshots or http://biopax.sf.net/m2repo/releases

Developing Paxtools

Checkout" (read-only) source using the Mercurial client:

hg clone http://biopax.hg.sourceforge.net:8000/hgroot/biopax/paxtools

If you get stuck or have questions, look for support in the Paxtools support mailing list: https://lists.sourceforge.net/lists/listinfo/biopax-paxtools.

For general questions related to BioPAX, use BioPAX discuss mailing list: http://groups.google.com/group/biopax-discuss.

Module Structure

- Core
- Jena IO
- Query
- PathwayCommons client
- PSI-MI converter
- Sif, GSEA, SBGN converter

My first model

```
Level3Factory factory = (Level3Factory)
BioPAXLevel.L3.getDefaultFactory();


Model model = factory.createModel();


Protein protein1 = model.addNew(Protein.class,
"http://biopax.org/tutorial/protein1");
```

Choosing a good ID

- BioPAX uses RDF-IDs to uniquely identify resources.
- IDs sould be valid URIs and globally unique.
- Current best practice is to use the URL of your project or institution as the base for generating IDs.

Adding fields

```
protein1.setStandardName("standard test");

protein1.setDisplayName("test");

BiochemicalReaction rxn1 =
model.addNew(BiochemicalReaction.class,
"http://biopax.org/tutorial/rxn1" );
```

- rxn1.addLeft(protein1);

Reading & Writing

JENA

```
BioPAXIOHandler handler = new JenaIOHandler();
```

or for level 3:

```
BioPAXIOHandler handler = new JenaIOHandler(BioPAXLevel.L3);
```

Simple

```
BioPAXIOHandler handler = new SimpleIOHandler();
```

Reading & Writing

```
Model model =
handler.convertFromOWL(inputStreamFromFile);
```

```
handler.convertToOWL(model, outputStream );
```

Exporting to SIF

```
    SimpleInteractionConverter converter = new
SimpleInteractionConverter(
        new org.biopax.paxtools.io.sif.level3.ControlRule(),
        new org.biopax.paxtools.io.sif.level3.ParticipatesRule(),
        new org.biopax.paxtools.io.sif.level3.ComponentRule(),
        new org.biopax.paxtools.io.sif.level3.ControlsTogetherRule());

converter.writeInteractionsInSIF(model,out);

converter.writeInteractionsInSIFNX(
    m,out,out,true,handler.getEditorMap(),"name","xref");
```

Exporting to SIF

```
    SimpleInteractionConverter converter = new
SimpleInteractionConverter(
        new org.biopax.paxtools.io.sif.level3.ControlRule(),
        new org.biopax.paxtools.io.sif.level3.ParticipatesRule(),
        new org.biopax.paxtools.io.sif.level3.ComponentRule(),
        new org.biopax.paxtools.io.sif.level3.ControlsTogetherRule());

converter.writeInteractionsInSIF(model,out);

converter.writeInteractionsInSIFNX(
    m,out,out,true,handler.getEditorMap(),"name","xref");
```

Using Property Editors

```
public static String[][] listProperties(BioPAXElement bpe)
    {
    // In order to use properties you first need to get an EditorMap
    EditorMap editorMap = new SimpleEditorMap(BioPAXLevel.L3);

    // And then get all the editors for our biopax element
    Set<PropertyEditor> editors = editorMap.getEditorsOf(bpe);

    // Let's prepare a table to return values
    String value[][] = new String[2][editors.size()];

    int row =0;
```

Using Property Editors -cntd

```
// For each property
    for (PropertyEditor editor : editors)
    {
            // First column is the name of the property, e.g. “Name”
        value[0][row]= editor.getProperty();
        // Second column is the value e.g. “p53”, note that the value is       // sometimes a Set and we are using the Set.toString() to display      // the contents of multiple cardinality properties
            value[1][row] = editor.getValueFromBean(bpe).toString();
        // increase the row index
        row++;
    }
    // We are done!
    return value;
}
```

Traversing, Cloning and Merging

```
BioPAXIOHandler jena = new JenaIOHandler();

Model model1 = jena.convertFromOWL(“pathway.owl”);

Model model2 =
model1.getLevel().getDefaultFactory().createModel();

model2.add( model1.getById(“The_reaction_id”) );

jena.convertToOWL(model2, file2OutputStream);
```

Traversing - cntd

```
public void visit(BioPAXElement domain, Object range,
 Model model, PropertyEditor editor)
   {
   // We are only interested in the BioPAXElements since
      // primitive fields are always copied by value
   if(range != null && range instanceof BioPAXElement)
      {
         BioPAXElement bpe = (BioPAXElement) range;

      if(!targetModel.contains(bpe))
      {
         targetModel.add(bpe);
            traverser.traverse(bpe, model);
      }
   }
   }
```

Traversing-cntd

```
public Model excise(Model sourceModel, String... ids)
{
// Create a new model that will contain the element(s) of interest
 this.targetModel =   editorMap.getLevel().getDefaultFactory().createModel();

for(String id: ids)
{
        // Get the BioPAX element
    BioPAXElement bpe = sourceModel.getByID(id);
    // Add it to the model
        targetModel.add(bpe);
    // Add the elements that bpe is dependent on
        traverser.traverse(bpe, sourceModel);
}

return targetModel;
}
}
```

Using Property Editors -cntd

```
// For each property
    for (PropertyEditor editor : editors)
    {
            // First column is the name of the property, e.g. “Name”
        value[0][row]= editor.getProperty();
        // Second column is the value e.g. “p53”, note that the value is       // sometimes a Set and we are using the Set.toString() to display      // the contents of multiple cardinality properties
            value[1][row] = editor.getValueFromBean(bpe).toString();
        // increase the row index
        row++;
    }
    // We are done!
    return value;
}
```

Queries

```
Set<BioPAXElement> sourceSet = new HashSet<BioPAXElement>();
    // Add the related source PhysicalEntity (or children) objects to the source set
    sourceSet.add(...
    int limit = 2;
    boolean upstream = true;
    boolean downstream = true;

    Set<BioPAXElement> result =
    QueryExecuter.runNeighborhood(sourceSet, model, limit, upstream, downstream);
```

Queries

```
Set<BioPAXElement> sourceSet = new HashSet<BioPAXElement>();
    // Add the related source PhysicalEntity (or children) objects to the source set
    sourceSet.add(...
    int limit = 2;
    boolean upstream = true;
    boolean downstream = true;

    Set<BioPAXElement> result =
    QueryExecuter.runNeighborhood(sourceSet, model, limit, upstream, downstream);
```

Queries

- Neighborhood
- GOI
- POI
- CommonStream
- CommonStream with POI

Accessing from Console

--merge <file1> <file2> <output>

--to-sif <file1> <output>

--to-sifnx <file1> <outEdges> <outNodes> prop1,prop2,etc...

--validate <path> <out> [xml|html]

--integrate <file1> <file2> <output> (experimental)

--to-level3 <file1> <output>

--psimi-to <level> <file1> <output>

--to-GSEA <file1> <output> <database> <crossSpeciesCheck>" converts level 1 or 2 or 3 to GSEA output. Searches database for participant id or uses biopax rdf id if database is NONE. Cross species check ensures participant protein is from same species as pathway (set to true or false).

--fetch <file1> <id1,id2,..> <output> extracts a sub-model (id1,id2, etc. - new “root” elements)

--get-neighbors <file1> <id1,id2,..> <output> nearest neighborhood graph query

Get Involved

- Use Paxtools and report issues and feature requests. See : http://biopax.sourceforge.net/paxtools/issue-tracking.html
- Respond to questions by other users at biopax-paxtools
- Try your hand at fixing bugs and implementing requests
- Improve this documentation.
- Spread the word. Let other people know about BioPAX and Paxtools.