The Virtual Leaf

Friday 11 June 2010

Issues:

1. Every class that has an association or composition eventually will need its own custom view page template, e.g. sed_view. For the moment I have set the default view for these classes to dev_view (a copy of base_view).
2. An association in UML is cast as a ReferenceField in the resulting class schema. Open the association and add the widget:label tag to the class that is being associated.
3. The default description text for a class should be a informative description of the class, not the instance, derived from the SED-ML manual.
4. Apply 'searchable' tags to the appropriate class attributes.
5. I choose 'isTidyHtmlWithCleanup' as the MathML validator. Will this work?
6. What should be added to our implementation of SED-ML to make it more than merely a mirror image of the SED-ML XML format?

    a. The ability to attach files at various points, e.g. parameter files.
    b. Additional output types, e.g. animations.
    c. Custom page templates
    d. Custom search form
    e. Custom catalog indexes
    f. Custom portlet(s)
    g. Custom viewlet(s)

Friday 24 September 2010

To Do:

1. Publish our mercurial repositories (Chris/Martin). This may mesh with deploying Plone.
2. Create a vleaf mailing list (Henk).

Found an article on SBRML - a markup language for associating systems biology data with models. At a glance the schema seems bigger than SED-ML's, but SBRML appears to be SBML specific.

Monday 27 September 2010

Spoke with Chris about publishing the Mercurial repositories and deploying Plone. Both can be accomplished in their own project space on a server in the DMZ. Chris said he would supply me with login credentials for both accounts. I will install Plone and Chris (or Maarten) will help me configure Apache/nginx and Varnish. Chris will install the required Mercurial packages; after which I can clone the repositories to their new home.

Tuesday 5 October 20010

tutorial0.pro

Added simplugin.h to list of header files. NB: this is the only tutorial in which simplugin.h appears.

auxingrowthplugin.h

Declared, and defined, a virtual function named `DefaultLeafML()` which merely returns a QString naming a LeafML filename sans path.

VirtualLeaf.cpp

Moved `Cell::SetMagnification()` and `Cell::SetOffset()` from `main()` to `MainBase::Init()`.

canvas.h

Declare `exportCellData()`.

canvas.cpp

Add an 'Export cell areas' to the file dropdown menu which invokes - surprise - `Main::exportCellData()`:

```
void Main::exportCellData(void) {
  QFile file("areas.csv");
  if ( file.open( IO_WriteOnly ) ) {
    QTextStream stream( &file );
    mesh.CSVExportCellData(stream);
    mesh.CSVExportMeshData(stream);
    file.close();
  }
}
```

mesh.h

Include `<QTextStream>`

Set the boundary_poloygon pointer to zero in the class constructor, and delete it, if it exists, in the class destructor.

Declare `Compactness()`, `CSVExportCellData()` and `CSVExportMeshData()`:

```
double Compactness(double *res_compactness=0, double *res_area=0, double *res_cell_area=0);
void CSVExportCellData(QTextStream &csv_stream) const;
void CSVExportMeshData(QTextStream &csv_stream);
```

mesh.cpp

In `mesh::clear()`, delete the boundary_polygon only if the pointer hasn't been assigned:

```
if (boundary_polygon) {
   delete boundary_polygon;
   boundary_polygon=0;
 }
```

Ditto for `mesh::clean()`.

Define the code for `Compactness()`, `CSVExportCellData()` and `CSVExportMeshData()`.

modelcatalogue.cpp

In `InstallModel()`, find and load the default LeafML file.

simplugin.h

Declare `DefaultLeafML()`:

```
// Default LeafML-file to be read after model startup
virtual QString DefaultLeafML(void);
```

simplugin.cpp

Define `DefaultLeafML()`. Returns an empty `QString`:

```
QString SimPluginInterface::DefaultLeafML(void) { return QString(); }
```

xmlwrite.cpp

In `Mesh::XMLReadCells()` - Delete the boundary_ploygon only if its pointer has been assigned.