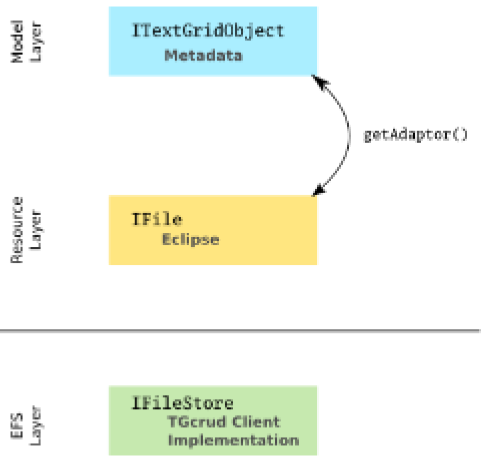
1 TextGridLab Data Model

2 Basic Architecture

TGcrud offers a rather different interface for saving information than Eclipse: Eclipse's storage system is based on a workspace containing named projects containing named files and folders, rather oriented on traditional file systems. TGcrud, however, maintains a set of objects described by a set of metadata (like authors, title etc.). Both can use URIs to identify files.

The plugins info.textgrid.lab.core.model and info.textgrid.lab.core.efs.tgcrud try to bridge the gap between both systems and to offer a convenient way for TextGridLab plugins to access both TextGridRep's objects and metadata.

2.1 Three implementation layers



We use three implementation layers to manage our objects:

- The model layer represents TextGridRep's objects using their metadata, in `TextGridObjects`. This layer is completely implemented by TextGrid, and it is the layer clients like object browsers or metadata editors usually talk with and which commands and menu entries on TextGrid objects are usually contributed to.
- The **resource layer** containing `IFiles`, `IProjects` and other `IResources` is the layer (content) editors usually see. It is the standard way of interacting with files in Eclipse, and it is completely implemented by Eclipse.
- The **EFS layer** contains the backend that performs the actual reading and writing. It is implemented by TextGrid extending an interface defined by Eclipse, and clients usually don't interact directly with this interface

2.2 Conversion

Conversion between the layers works by using the adaptor pattern as provided by Eclipse. I.E., if you have a TextGridObject and need an IFile,

```
TextGridObject textGridObject;
/* ... fill textGridObject ... */
IFile file = (IFile) textGridObject.getAdapter(IFile.class);
if (file != null)
  /* do something with file */
```

or vice versa

```
IFile file;
/* ... fill file ... */
TextGridObject textGridObject = (TextGridObject) file.getAdaptor(TextGridObject.class);
```

2.3 Tasks

Everything in this section is subject to change / refactoring. Please always check with the implementation, and please always see the javadocs (which may be more detailed, anyway).

2.3.1 Getting a TextGridObject

2.3.1.1 From a TextGrid URI

```
URI uri;
/* get the object's URI somehow into uri */
TextGridObject object = TextGridObject.getInstance(uri, true);
```

This creates a new TextGridObject from a textgrid uri, calling TGcrud#readMetadata if neccessary. If you set the last argument to `false`, the call to readMetadata may be delayed.

2.3.1.2 From a Metadata blob

You need to pass in the metadata in the form of an ObjectType of our core metadatsa as generated from JAXB.unmarshall().

```
final ObjectType metadata = JAXB.unmarshal(metaFile, ObjectType.class);
boolean complete;   // if false, element contains only fragmentary metadata
TextGridObject object = TextGridObject.getInstance(metadata, complete);
```

If you create a TextGridObject from an XML fragment, there will be no immediate network access. This is especially useful if you create a lot of TextGridObjects at once (think search) and do not want a web service call for every single one. With the last argument (complete), you determine whether the metadata fragment you pass in is complete, i.e. it contains all the metadata TextGrid knows about for this TextGridRep object. If you specify false here and access a field not present in the metadata fragment at a later point in time, the object is finally completed by a call to TGcrud's readMetadata operation.

2.3.2 Representing TextGridObjects

If your plugin somehow represents / contains a bunch of TextGrid objects (e.g. search results or object browser), your objects (the stuff you put in your StructuredSelections and/or that your ContentProvider provides) must adapt to TextGridObjects (and they should adapt to IFile, as well; once they have a TextGridObject tgo they do so by forwarding the adaption request to TextGridObject, i.e. `return tgo.getAdaptor(IFile.class)`).

Either your objects are PlatformObjects or IAdaptables and you provide an adaptor factory, or you simply put the TextGridObjects themselves in your selection/viewer.