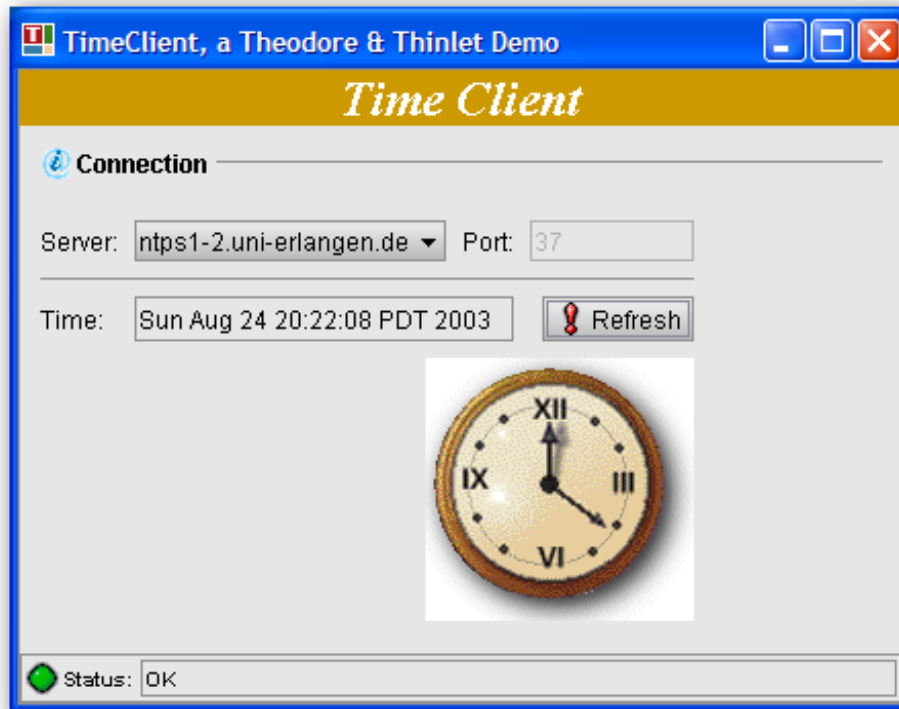


Theodore – Best Practice:

TimeClient Sample application

Project description:

Thinlet based Java application displaying the current time acquired by an Internet Time server.



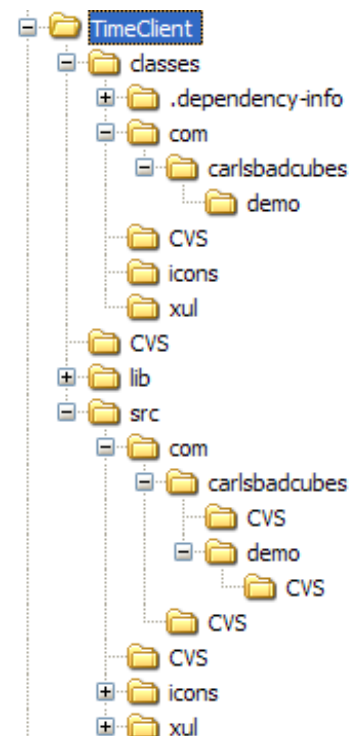
Project Setup

On a Windows box, the C:\work\TimeClient directory serves as the home you the project during the development process.

The *TimeClient* project's directories are setup best separate Java source form Java class files. The classes are stored in TimeClient/classes while the source files are stored in the TimeClient/src sub tree.

Resources

The required `thinlet.jar` like other optional jars a project might require may be put best into the TimeClient/lib directory. Icons and images go into the Thinlet/src/icons directory and the Thinlet GUI descriptor XML files are being stored in the TimeClient/src/xul directory.



Built Process

No matter if ANT or an IDE is used to support the built process, all icons and XML files need to be copied from the source into the classes tree during the built process.

Business Logic

Since this document is all about how to use Theodore when developing a Thinlet based Java application but less concerned with the actual functionality of the application, let's get the business of the sample application behind us. The `com.carlsbadcubes.demo` package has a `TimeRequester` class, providing static methods to communicate with a few Time Servers, which have been already defined in the class (Source and JavaDoc can be found attached to this document.)

Theodore Project Setup

Now with the biz logic available, let's start to work on the GUI. The project's major Java class, sub-classing `Thinlet`, is `com.carlsbadcubes.demo.TimeClient`.

```
package com.carlsbadcubes.demo;

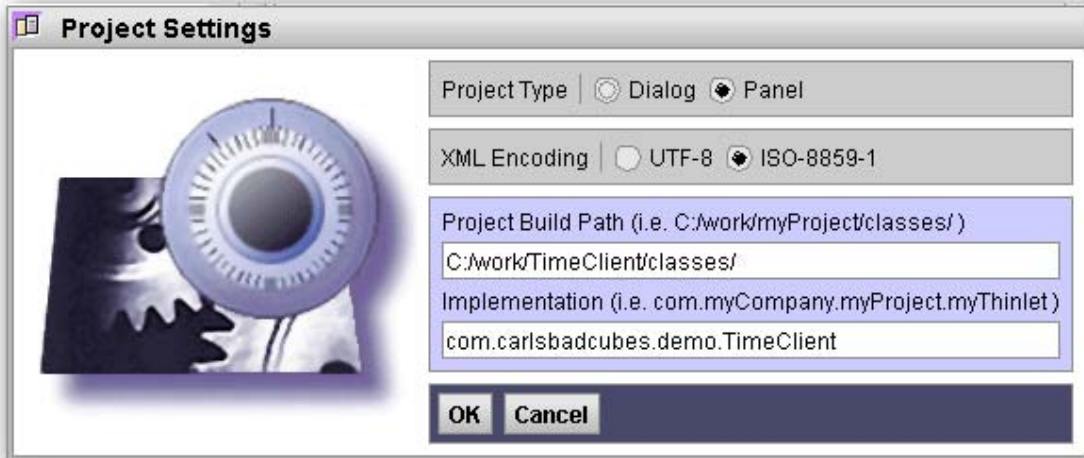
import thinlet.FrameLauncher;
import thinlet.Thinlet;

import java.awt.*;
import java.io.IOException;

/**
 * Simple Thinlet based TimeClient application displays current Internet Time,
 * learned from various servers.
 */
public class TimeClient extends Thinlet {

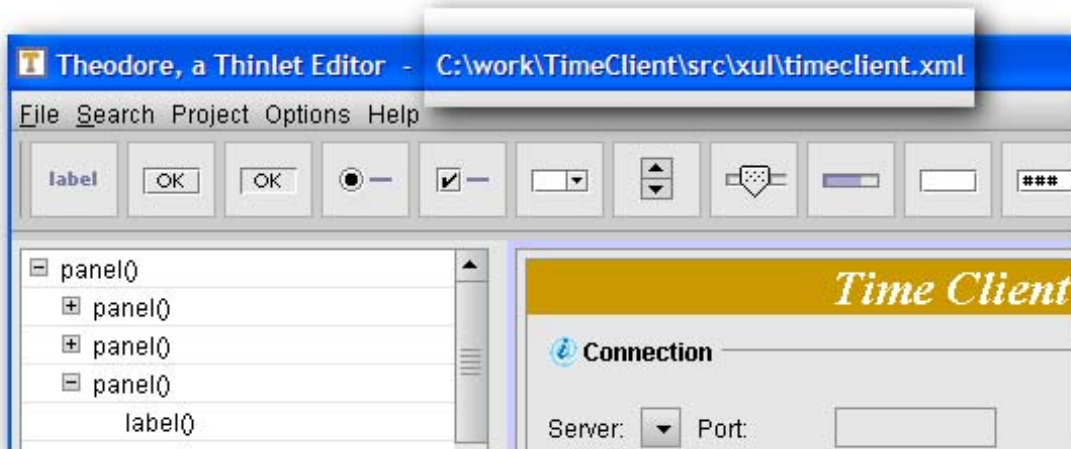
    /** <code>String</code> XUL for the Time Client GUI. */
    private static final String XUL = "/xul/timeclient.xml";
```

After launching Theodore, the Project settings are entered like this:



Theodore works rather on the class files (compiled byte code) than on the Java source. Therefore, the Project Build Path is set to `C:/work/TimeClient/classes` – since Theodore is capable of introspecting classes, i.e. to find public methods qualifying as event handlers, the implementation class is set to `com.carlsbadcubes.demo.TimeClient`.

With those settings in place, the GUI can be designed in Theodore and stored in the source tree, i.e. `C:/work/TimeClient\xul\timeclient.xml`. It's important to not confuse the locations. If the XML-descriptor, available in the classes' tree, were edited, changes would be lost since the next build process would overwrite this file with the one available in the source tree.



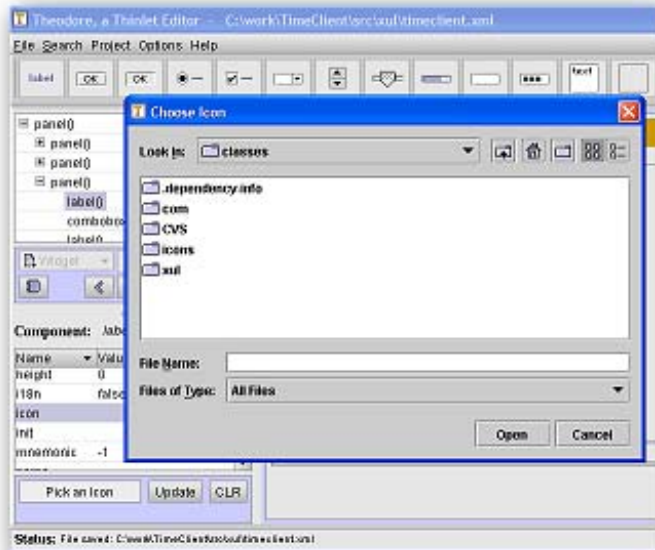
With all these settings made, it's now very easy to set icons and link event handlers.

Icons

Decorating a label with an icon is done by first selecting the label in tree view and secondly the icon attribute in the attribute table.

Clicking the “Pick an Icon” button brings up a file browse dialog, which allows selecting a file.

Finally, clicking the update button will insert the file name into the XML descriptor and also update the preview and display the icon.



Event handlers

Linking event handlers to widgets works through introspection of the class file specified in the project settings dialog.

After an active widget, like a button for instance, is selected in the tree-view and the action attribute is selected in the attribute table, the edit area is rendered as an editable combo box, which allows to select any available method, fitting the valid event handler signature (public void methodName(..)).

Since Theodore works with the class files, a modified java file needs to be compiled before Theodore picks up new or modified methods.

