Inhoud

[1 Nota’s en Eigendommen 1](#_Toc41598502)

[2 Media 1](#_Toc41598503)

[2.1 Data model 1](#_Toc41598504)

[2.2 Media Database Application 2](#_Toc41598505)

[2.2.1 goedegep.media.mediadb.albuminfo 4](#_Toc41598506)

[3 Rolodex 6](#_Toc41598507)

[4 Unit Converter 6](#_Toc41598508)

[5 Vacations 6](#_Toc41598509)

[6 PCTools 6](#_Toc41598510)

[7 External Projects 6](#_Toc41598511)

[7.1 Gluon Maps 6](#_Toc41598512)

# TEMP Dialogs

## Swing

OptionDialog

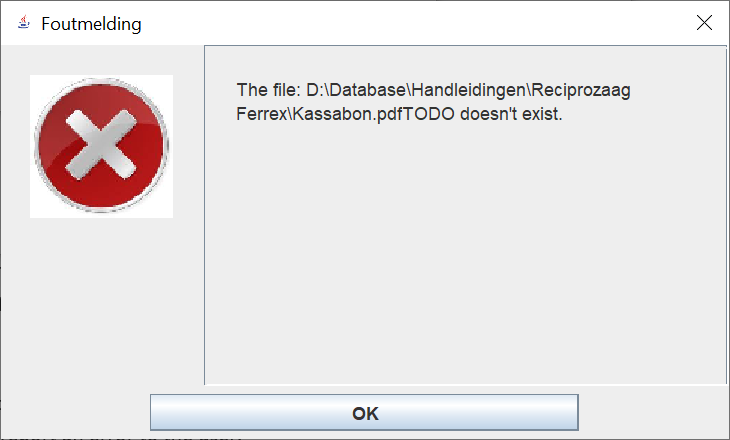
AppDialog

javax.swing.JDialog

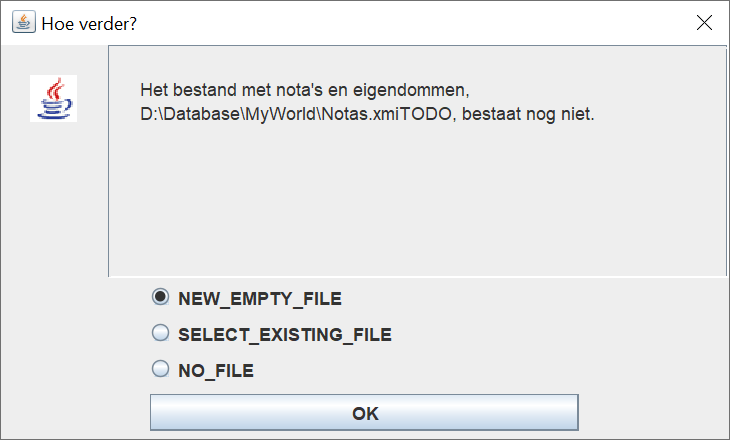
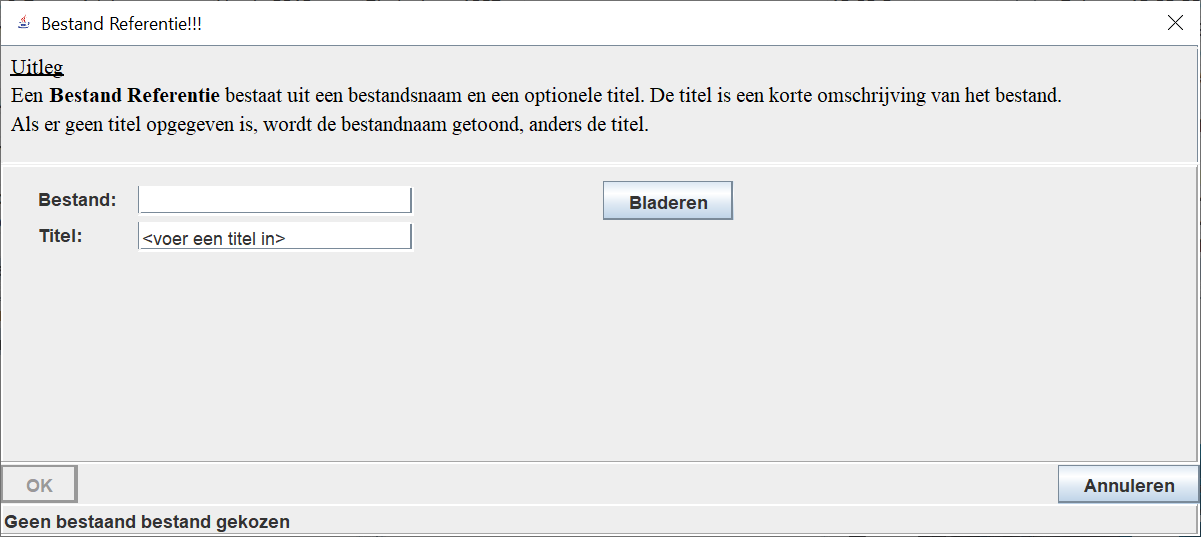
MessageDialogType

MessageDialog

My Swing implementation provides:

* AppDialog  
  Abstract common parent. Contains no real functionality.
* MessageDialogType  
  Specifies the type of MessageDialog. Value: ERROR, WARNING.
* MessageDialog  
  This is an information dialog. Type only specifies the icon (ERROR, WARNING), the icon can also be set to anything else.  
  



* OptionDialog  
  The user is offered a number of options via a number of radio buttons.  
  There’s a single ‘OK’ button.  
    
  
* BestandReferentieDialog  
  This is a dialog to enter a BestandReferentie.  
  

## My Current JavaFx solution

## JavaFx standard available dialogs

## What do I need?

**General layout:**

* Title in title bar (parameter)
* Icon in title bar (customization)
* Icon to indicate type of dialog:
  + Error dialog: X
  + Option dialog :?

**Dialog types:**

* Error dialog (information only)  
  Use case: report an error to the user.
  + Icon to indicate type of dialog: X
  + text to explain the error
  + OK button
* Warning dialog (information only)  
  Use case: warn the user about something.
  + Icon to indicate type of dialog: !
  + text with the warning
  + OK button
* Option dialog  
  Use case: user has to select one of several options, before the application can continue.
  + Icon: ?
  + text to explain the option selection
  + radio buttons to select option
  + OK button
  + Do I need a cancel button? (not for now)
* FileReference dialog  
  Use case: create a new FileReference.
  + No icon
  + text to explain the FileReference
  + fields to select a file
  + fields to specify a title
  + OK button – FileReference will be created
  + Cancel button – No action.

## Solution

* Stay as close as possible to the JavaFx dialog API.
* Error dialog -> Alert, type is ERROR
* Warning dialog -> Alert, type is WARNING
* Option dialog -> ChoiceDialog (combo box instead of radio buttons).
* FileReference dialog: to be created.

# Nota’s en Eigendommen

This component has sub components, so it uses a component launcher, class NotasEnEigendommenMenuWindow.

# Media

## Data model

Albums/tracks

Track are stored as a separate list.  
An Album has discs.

Track things to handle:

* commercial disc
  + references to the tracks needed, for a track listing of the disc.  
    Covered by a list of TrackReferences.
  + if I have the complete album, information about where (can be per track)  
    Needed to play the tracks.  
    TrackReferences -> Tracks -> MyTrackInfo -> iHaveOn
  + if I have tracks as part of my own compilation, information about which tracks on which compilation (has to be part of MyInfo)  
    TrackReference.myInfo. *compilationTrackReference*
  + if I have one or more tracks as part of a Collection, information about which tracks in which Collection (has to be part of MyInfo).   
    Is part of the MyInfo per track.
* MyCompilation album  
  By definition, I have all tracks.
  + references to the tracks needed, for a track listing of the disc  
    The trackReferences
  + information about where (can be per track)  
    Needed to play the tracks. Covered by IHaveOn/Collection of the track
  + per track a reference to the track of the original album (has to be part of MyInfo)  
    covered by *originalAlbumTrackReference*
* Separate tracks, stored in Collections
  + References to the tracks needed, for a track listing (per Collection)  
    These are all tracks in MediaDb.track, which have collection set to a specific collection.
  + References to the track needed to be able to play them  
    Find in complete track list, or collection.
  + A possible reference to the album it came from  
    Covered by referredBy

For a track of a disc the following applies:

* commercial disc  
  Each track is a TrackReference to a Track.
* my own compilation disc  
  Each MyTrackInfo.TrackReference

Updated data model:

## Media Database Application

Hi

AlbumsTable

albumsTable

MediaDbWindow

TracksTable

**goedegep.media.mediadb.app.guifx**

**goedegep.media.model.mediadb**

**goedegep.appgenfx**

tracksTable

**goedegep.util**

AlbumDiscLocationInfo

**goedegep.media.mediadb.app**

**goedegep.media.mediadb.albuminfo**

Description per package:

* goedegep.media.mediadb.app.guifx  
  The GUI of the application.
* goedegep.media.mediadb.app  
  The functionality of the application (as far as not covered by the other packages).
* goedegep.media.mediadb.albuminfo  
  This package has all the knowledge of the structure and file name conventions of the AlbumInfo files. Information read from the AlbumInfo files is stored in the MediaDb data model, and a MediaDb data model can be written to AlbumInfo files.
* goedegep.media.model.mediadb  
  The data model of the media database.  
  TODO: to be renamed to goedegep.media.mediadb.model.

### goedegep.media.mediadb.albuminfo

This package has all the knowledge of the structure and file name conventions of the AlbumInfo files. Information read from the AlbumInfo files is stored in the MediaDb data model, and a MediaDb data model can be written to AlbumInfo files.

AlbumInfoHandler

albumInfoHandler

AlbumInfoFilesReader

AlbumInfoErrorInfo

**goedegep.media.mediadb.albuminfo**

**goedegep.media.model.mediadb**

**goedegep.util**

AlbumInfoError

errorCode

errors

* **AlbumInfoFilesReader**  
  This class reads 'AlbumInfo' files and stores the information in a **MediaDb** structure. It also writes the information from a **MediaDb** structure to 'AlbumInfo' files.  
  This class has the knowledge of the filename conventions. For reading it scans the folder with 'AlbumInfo' files. For each file it calls an **AlbumInfoHandler** to parse the file.
* **AlbumInfoHandler**This class reads 'AlbumInfo' XML files and stores the information in a **MediaDb** model.  
  The schema for these files is defined in the xsd file "D:\Database\Muziek\Albums\AlbumInfo.xsd".  
  The parser can generate errors of the following types: **AlbumInfoErrorInfo**, **goedegep.util.ParseException**.

# Rolodex

This component has sub components, so it uses a component launcher, class RolodexMenuWindow.

# Unit Converter

This component is a single application, where the main class is UnitConverterWindow.

# Vacations

This component is a single application, where the main class is VacationsWindow.

# PCTools

This component has sub components, so it uses a component launcher, class PCToolsMenuWindow.  
The sub components are:

* FileFinder
* FilesControlled

# External Projects

## Gluon Maps

This project provides a map view.

MapPoint

MapView

javafx.scene.layout.Region

BaseMap

baseMap

MapLayer

layers

javafx.scene.Parent

javafx.scene.Group

**JavaFx**

**com.gluonhq.maps**

* **com.gluonhq.maps**
  + **MapPoint**  
    This class provides a point on a map, which are just the latitude/longitude coordinates.
  + **MapView**This is the top UI element of the map component.  
    This class is mainly a wrapper around a **BaseMap**.
  + **MapLayer**A MapLayer can be added on top of a BaseMap (which provides the map tiles).  
    This class only provides the framework for map layer. Any specific map layer shall extend this class and override the methods: initialize() and layoutLayer().
* **com.gluonhq.impl.maps**
  + **BaseMap**  
    The BaseMap provides the underlying map tiles. On top of this, additional layers can be rendered.