TAB2XML Design Document

For version 0.4.0

2021 March 26

Contents

1 Front End Design

All TAB2XML front-end code is located in the `tab2xml.gui` *package.*

1.1 Front End Classes

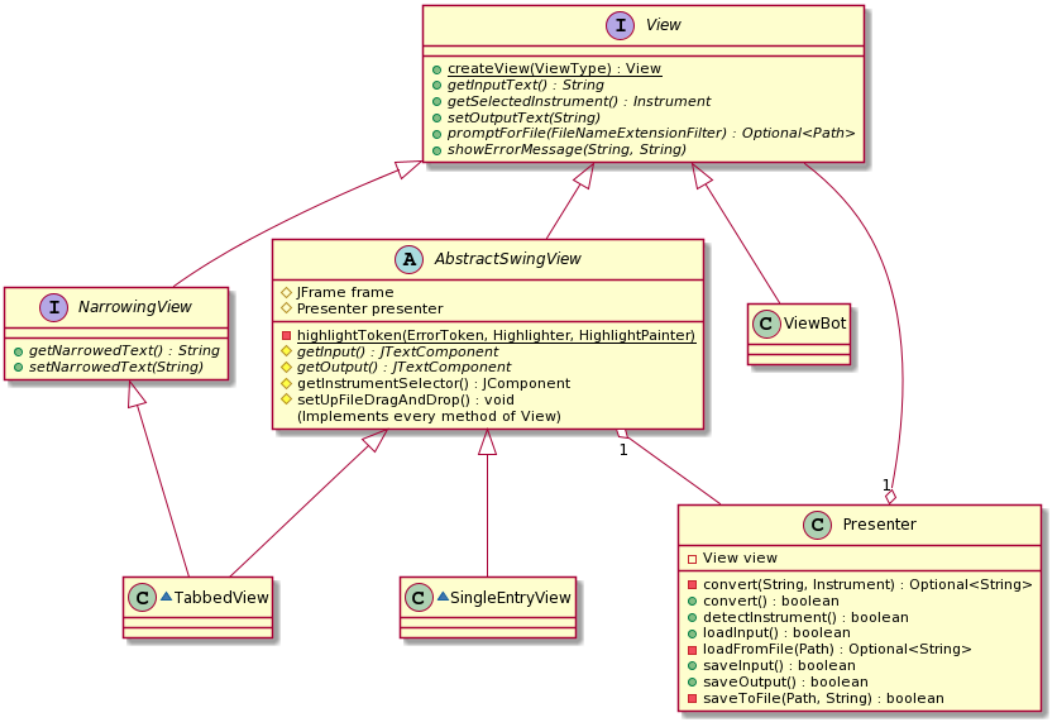

Figure 1: A class diagram for the frontend of TAB2XML.

The frontend of TAB2XML is designed using the MVP paradigm. It is divided into two main parts, the `View` and the `Presenter`.

The `View` is the part of the frontend that interacts with the user (the GUI). It is handled by the `View` interface; the GUIs for TAB2XML implement the View interface. In addition, all Views that represent a Swing GUI are subclasses of the skeletal implementation `AbsractSwingView`, which reduces the effort needed to make a View. Currently, there are four concrete classes implementing `View`: `TabbedView` (currently the one in use), `SingleEntryView`, `DoubleEntryView` and `ViewBot` (a mock view used for testing). The `NarrowingView` interface represents Views that additionally support TAB2XML's tab-narrowing functionality.

The `Presenter` is the part of the frontend that interacts with the backend code. It is a single class, not an interface that has multiple implementations. It implements behaviours such as converting a tab, loading from a file and detecting the instrument of the input tab. It uses the View interface's public methods to interact with the view. This means that the View's buttons can simply be linked to call the Presenter's methods, instead of having to implement the method in the View. All of the Presenter's methods return either a `boolean` or an `Optional` to describe whether they succeeded or not.

The rationale behind this design is to reduce the effort involved in creating a new GUI. If it extends `AbstractSwingView`, creating a new View is as simple as making a "mockup" Swing GUI and implementing two trivial methods. This makes it easy to work with multiple GUIs at once (allowing the customer to choose which they prefer). This design was especially important in the beginning of development, because I could prototype different GUI ideas with the customer using fully functional applications.

1.2 Converting Text Tabs

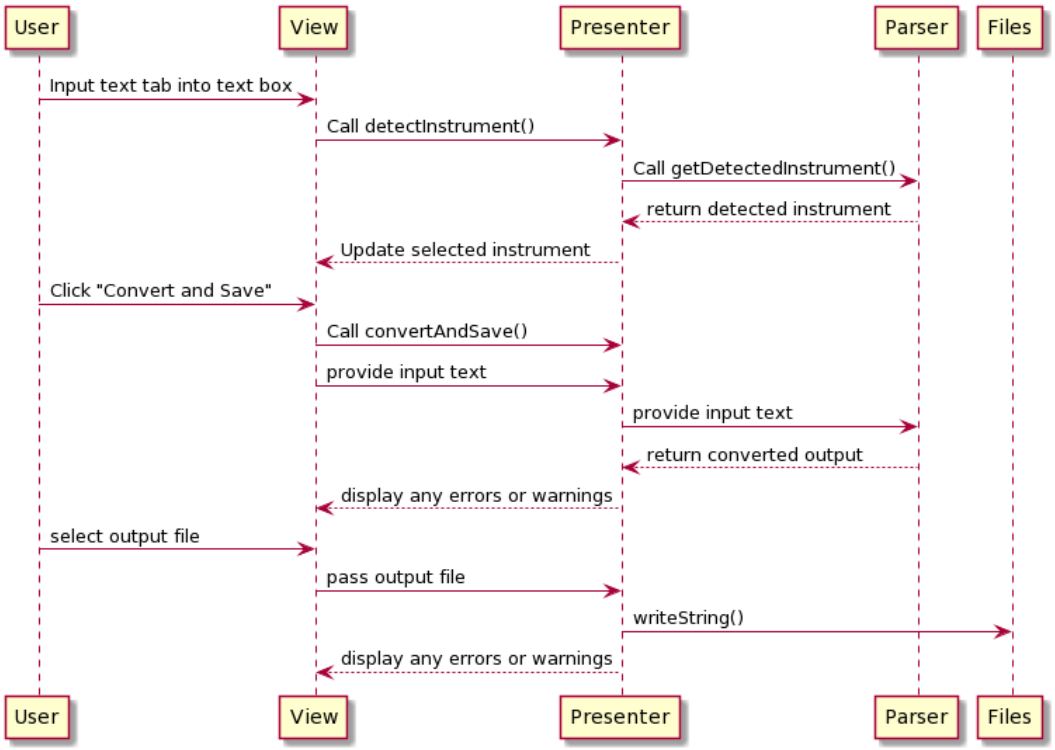

Figure 2: A sequence diagram for the "Convert and Save" operation

Here is how the "Convert and Save" operation works:

1. The user inputs the tab into the input text box (by typing, copy-and-pasting, the "Load from File" button or dragging and dropping a file).
2. The `View` calls the backend method `Parser.getDetectedInstrument(String)` with its text as input.
3. If it succeeded, the `View` sets its selected instrument to the detected instrument.
4. The user clicks the "Convert and Save" button.
5. The `View` calls the `Presenter`'s `convertAndSave()` method.

6. The `Presenter` calls the `View`'s `getInputText()` and `getSelectedInstrument()` methods to get the input tab and selected instrument.

7. The `Presenter` creates a new instance of `Parser` with the obtained input text and instrument. It then calls the `Parser`'s `parse()` method to convert the text tab.

8. The `Parser` returns the MusicXML, as well as any errors that occurred. Critical errors are thrown as Exceptions, noncritical errors are returned. This distinction exists so that critical errors stop the parsing, while noncritical errors do not stop it.

9. The `View` displays any errors or warnings to the user.

10. The `Presenter` calls the `View`'s `promptForFile` method to prompt the user for the desired destination file.

11. The `Presenter` calls `Files.writeString` to write the text tab to the selected file.

12. The `View` displays any errors that occurred during the file-saving operation.

1.3 Front End Maintenance

To create a new GUI, simply make your GUI handled by a class that extends `View`. It should also have a `Presenter` field instantiated using `new Presenter(this)`. You must implement all of the `View`'s methods, which is much easier if you extend `AbstractSwingView`. Then, you can call the presenter's methods within the GUI, and your GUI will be fully connected to the TAB2XML backend.

To modify the look of an existing View such as `TabbedView` (or add/remove components), simply modify its constructor (you may have to edit the other methods, if they are broken by the change). If you are adding a new feature that should exist in every Swing View, consider instead adding it to `AbstractSwingView`, as this will make it available for every View.