SongBase Documentation

1 Introduction

SongBase is a database application for making music on your Microsoft Windows or Linux laptop or PC. Songs can have up to 15 instrumental voices and a percussion track. Instruments can be chosen from either the default soundbank of 128 instruments or other soundbanks of your choice.

Songs can be developed, tested and edited very quickly and easily by virtue of the database structure and the ***JFugue*** MusicString notation. Notes within a MusicString are specified by their name and octave or by their MIDI value and their durations are specified either by character code, or numerically. A scale of quarter notes can be written as:

C5q D5q E5q F5q G5q A5q B5q C6q

Sequences of MusicStrings can be assembled into Patterns, which can be nested to any level. Songs can be constructed from Patterns either timewise or by voice.

The *File/Template* menu item creates song templates by voice where each voice is a Pattern containing other Patterns and MusicStrings. Bars can be MusicStrings or Patterns. Pattern bars can be used to contain separate MusicStrings and/or Patterns.

All MusicStrings and Patterns within a song are given unique names which makes it easy to keep track of them within a composition. A MusicString can be shared between any of the Patterns in a song and it only needs to be edited once for the changes to be effective wherever that MusicString occurs within the song.

You can collaborate with your colleagues on a song by using a shared database. MySQL, Oracle and PostgreSQL are supported. You can also share your songs by exporting and importing to and from XML files.

SongBase will play back your music and also export to MIDI, MusicXML and WAV files for music processing and music publishing.

MP3 files can be produced by importing the MIDI files into ***SynthFont***, and optionally adding reverb using ***FreeverbToo***, then saving to MP3 using *Play to File*. Alternatively, the WAV files can be opened in ***Audacity***, then exported to MP3.

1.1 Background

SongBase is written in Java with a Swing forms user interface. It uses Apache Derby as the default database with an embedded JDBC driver. The music is generated by ***JFugue***, a Java API for music programming.

2 Installation

To install ***SongBase*** on Microsoft Windows you will first need to install the latest version of Java from ***Java.com***, or the OpenJDK Java Runtime from the Ubuntu Software Center if you are using Linux Ubuntu.

Then, from ***Gray10.com***, download and install the desktop shortcuts for setting up and running ***SongBase***.

The *Setup* shortcut is only required if you wish to create a database other than the default Apache Derby database.

If you are upgrading from version 2.0.3 or earlier of ***Song Builder*** then please note that the structure of the database has changed and you will need to follow the following steps for a successful upgrade:

1. Use *File/XML Export* to backup your songs to *.sbxml* files.

2. Delete the directory *C:\users\public\AppData\Donald G Gray\Databases\songbuilder* and/or any MySQL, Oracle and PostgreSQL databases.

3. Download and run the ***SongBase*** installer.

4. Run the *Setup* shortcut (optional).

5. Use *File/XML Import* to restore your songs.

You will also have to reload any soundbanks and re-associate these with your songs.

If you are using shared databases, you will need to re-input the connection parameters.

If you are upgrading from any version of ***Song Builder,*** then uninstall ***Song Builder*** from the Start Menu before installing ***SongBase***.

2.1 Soundbanks

SongBase uses the Gervill synthesizer to generate sound. Gervill can use the default soundbank or any soundbank with a *.sf2* or *.dls* extension.

Soundbanks can be kept on file or you can use *Soundbanks/Manage Soundbanks* to upload these to the database.

If you play a Pattern and specify the default soundbank, Gervill will search the locations:

- *<Java Runtime location>\lib\audio*.dls and *.sf2*
- *<Windows location>\System32\drivers\gm.dls*

If a soundbank is not found, Gervill will generate one using software.

*You can create your own soundbanks by recording instrument sounds in **Audacity** and importing them into the **Viena** SoundFont editor which is available from the **SynthFont** website.*

2.2 Backing up your Songs

Use *File/XML Export* to export a song to an XML file or *File/XML Import* to import a song from an XML file. The entire song structure is exported or imported with the exception of any soundbank reference.

2.3 Shared Database (optional)

To set up a shared database, use the configuration templates in:

Windows: *<Program Files location>\Donald G Gray\SongBase*
Linux: */opt/SongBase*

which you can edit, as Administrator, to suit the connection settings for your environment. These files can then be referenced by editing the corresponding parameter values in the *SongBase - Setup* desktop shortcut target.

Please ensure that your database is set up to allow direct access.

If are using PostgreSQL in a shared environment then ensure that the *pg_hba.conf* file in the PostgreSQL data directory of the database that is to be shared contains the line:

host all all 0.0.0.0/0 md5

If you are using Windows Firewall on a shared machine, then add a port exception for the shared database.

Use *Database/Database Connections* to configure a connection to the shared database and to connect to it.

If you are using a Database as a Service provider, then your hostname, port, database, username and password may be set up for you. In this case, use *Database/Database Connections* to create a new connection and enter the Connection Details. Please note that when entering the Connection Details, there should be no leading or trailing spaces in any of the fields as this will cause a communications failure. Then use *File/Create Tables* to create the database tables on the remote host.

3 Tutorial

SongBase opens with the ***Songs*** form for managing your songs. If you are using numeric note durations within a song, use the **/durations** column to specify whether these are expressed as decimal values of a whole note or as pulses, where 32 pulses represents a quarter note.

To ensure that the different voices in a song are synchronized, all note durations should equate to a whole number of pulses.

Numeric durations should be used for triplets. A quarter note divided into three triplets should be given durations of 11, 11 and 10 pulses, or 12, 10 and 10 pulses if you want to give more emphasis to the first triplet.

To edit a cell, either select the cell and type something into it or double-click on the cell. To copy data in a cell, highlight the text to be copied and press Ctrl-C. To paste, press Ctrl-V. This only works if the cell is in edit mode (yellow).

The **Strings** button opens a form for defining the ***JFugue*** MusicStrings in a selected song and the **Patterns** button opens a form for declaring the Patterns in the song.

The naming of MusicStrings is independant from the naming of Patterns. You may find it helpful to name MusicStrings in lower case and Patterns with an initial capital.

Use *File/Template* to give you a start when creating new songs.

Use *File/Tree View* to display the structure of Patterns and MusicStrings in a selected song (see below).

3.1 Strings

The ***Strings*** form is used to define the MusicStrings in the selected song. MusicString notation is described in the next section.

MusicStrings can be edited within this form or, if they are more than two lines long, the **Popup Editor** can be used to provide an editing window.

MusicStrings can be imported from other songs by using *File/Import Strings*. Libraries of MusicStrings can be built and used in this way.

3.2 Patterns

The ***Patterns*** form is used to declare the Patterns in the selected song. A Pattern is a container for other Patterns and MusicStrings.

The **Components** button opens a form for picking the Patterns and MusicStrings that are to be contained within a selected Pattern.

The **Play** button will play the selected Pattern.

The *File* options are *Export to MIDI, Export to MusicXML, Export to WAV* and *Clone.* If you want to create a Pattern that is very similar to an existing Pattern, use *Clone* and edit the clone.

3.2.1 Example of Cloning

Suppose you have a song which is 128 bars long and the first 12 bars are repeated with a different 12th bar on the repeat.

Use the template to create a song where the voices have 2 parts with 128 bars per part. Clone each Part 1 and call this the Repeat. Insert the Repeat after Part 1 in the voice patterns. Delete bar 13 from Part 1 and delete bar 12 from the Repeat. Bar 13 in the Repeat becomes the different 12th bar.

You can keep the existing bar numbers and since bar numbers are independant for each part you can start Part 2 at bar 13.

3.3 Components

The ***Components*** form is used to pick the Patterns and MusicStrings that are to be contained within the selected Pattern and to specify their position in the sequence in which they are to be played. Picking is done via a drop down list which is displayed when you click on a **Component** cell.

The **Insert** and **Renumber** buttons can be used to insert components into an existing sequence.

The **Drill Down** button allows you to drill down to the child components of a selected Pattern. Drilling down on a MusicString will provide a window for editing that MusicString. When selecting a component for **Drill Down** it is advisable to click on the **Position** cell otherwise the component pick list will be displayed.

If you would like to enter an anonymous MusicString into your Pattern, leave the **Component** cell blank and type the MusicString into the **Anonymous String** cell. This column can be made wider by dragging the boundary between the **Component** and **Anonymous String** headers.

3.4 Tree View

The tree view may be used to display the structure of Patterns and MusicStrings in a selected song.

Clicking on a MusicString will open a popup window for editing that MusicString. Use the **Refresh** button to propagate any changes made here throughout the tree.

Clicking on a Pattern will enable the **Play** button and also the *File* options: *Export to MIDI, Export to MusicXML, Export to WAV* and *Clone.*

3.5 MusicXML

If you export a Pattern to a MusicXML file, suitable for input to a music publishing system such as ***MuseScore***, then for best results:

- Tempo should be the first token in the Pattern and each different Voice token should appear only once in the Pattern and be followed by one each of Instrument and Key tokens.
- Combined harmony and melody should be represented by separate voices. Notes connected to chords using the _ character will not be exported to the MusicXML file.
- Bar lines should be used.
- If numeric note durations are used they should match exactly to durations within the range whole note going down by halves to 128th note (dotted or undotted, but not a dotted 128th note).

4 MusicStrings

4.1 Notes

A C Major scale of quarter notes, starting at middle C, can be written as:

```
C5q  D5q  E5q  F5q  G5q  A5q  B5q  C6q
```

or as:

```
C5/0.25  D5/0.25  E5/0.25  F5/0.25  G5/0.25  A5/0.25  B5/0.25  C6/0.25
```

or, if you have **/durations** set to pulses, as:

```
C5/32  D5/32  E5/32  F5/32  G5/32  A5/32  B5/32  C6/32
```

In addition to the note letters, A to G, you can use R for a rest. Sharps, flats and naturals can be added by placing the character #, b or n immediately after the note letter so B-flat above middle C is writen as Bb5.

Please note that, if you are transcribing music, accidentals in SongBase apply only to the immediately following note and not to the end of the bar as in conventional music notation.

MusicStrings can optionally be split into bars (or measures) by using the vertical bar character (|):

```
C5q  D5q  E5q  F5q | G5q  A5q  B5q  C6q |
```

As an alternative to note letters, MIDI values, enclosed in square brackets, may be used:

```
[60]q  [62]q  [64]q  [65]q | [67]q  [69]q  [71]q  [72]q |
```

4.1.1 Durations

The duration characters are:

w	whole note
h	half note
q	quarter note
i	eighth note
s	sixteenth note
t	thirty-second note
x	sixty-fourth note
o	one-twenty-eighth note

Dotted duration can be achieved by putting the period character (.) immediately after the duration character.

4.1.2 Chords

Chords are formed by adding the constituent notes together. A C Major chord can be written as:

```
C5q+E5q+G5q
```

4.1.3 Ties

Two or more notes of the same pitch can be tied together by using the hyphen character (-). Place the hyphen immediately after the duration of the note at the start of the tie and immediately before the duration of the note at the end of the tie. Notes in the middle of the tie have hyphens placed immediately before and after the note duration:

```
C5q  D5q  E5q  F5q- | F5-w- |  F5-q  G5q  A5q  B5q |
```

4.2 Tempo

A tempo of 120 beats per minute can be expressed as:

```
T120
```

or as:

```
T[allegro]
```

Note the use of a pre-defined numeric constant within the square brackets. More tempo constants are available from *Insert/Tempo*.

4.3 Constants

Constants are defined using the $ character followed by the constant name.

4.3.1 Numeric Constants

Numeric constants can be used anywhere that a number would appear in a MusicString. In addition to the pre-defined constants available from the *Insert* menu, you can define your own constants. For example, a scale with non-standard MIDI values could be defined as:

```
$A1=70
$G=68
$F=67
$E=65
$D=63
$C=62
$B=60
$A=58
```

and played as:

```
[A]q [B]q [C]q [D]q [E]q [F]q [G]q [A1]q
```

4.3.2 String Constants

Suppose you wanted to use the following arpeggio several times in your music:

```
F3i A3i C4i F4i C4i A3i
```

You could define a string constant as:

```
$arpeggioFoctave3=F3i~A3i~C4i~F4i~C4i~A3i
```

Then, in your music, you could refer to it as:

```
{arpeggioFoctave3}
```

Note the use of curly brackets for string constants.

4.4 Voices

Voices are specified by the V character followed by a number from 0 to 15. Note that V9 is the percussion voice and has its own set of instruments.

4.5 Key Signatures

Key signatures are specified by the K character followed by a note letter (or a note letter followed by # or b) followed by maj or min to indicate a major or minor scale.

4.6 Instruments

Instruments are specified by the I character followed by a number from 0 to 127. You can use *Insert/Instrument* to pick one of the pre-defined values.

4.7 MIDI Controller

MIDI controller events can be specified by the X character followed by the controller number followed by the equals sign (=) followed by a value. You can use *Insert/Controller* to pick one of the pre-defined controllers.

4.8 Pitch Wheel

A change of pitch can be specified by the & character followed by a number from 0 to 16383. This affects all following notes:

&0	lowers the pitch by a full tone;
&8192	returns the pitch to no change;
&16383	raises the pitch by a full tone.

5 Linux

To install on Linux Ubuntu you will first need to use the Ubuntu Software Center to install the OpenJDK Java Runtime.

If you have problems installing Java, you may first need to run *sudo apt-get update* from the command prompt.

Then, download the appropriate package for ***SongBase*** and install using the Ubuntu Software Center.

The programs will be installed in */usr/share/applications* and are named:

Setup SongBase
SongBase

If you have a preference for Microsoft Windows fonts, run *sudo apt-get install msttcorefonts* from the command prompt.

Since no soundbanks come with this version of Java, Gervill will generate one automatically. You may wish to search the Internet for SoundFont files to get better quality. You can also follow the SoundFont links on the ***SynthFont*** website.

A quick way to get a SoundFont is to first run *sudo apt-get install timidity lame* from the command prompt, then from the Ubuntu Software Center, find timidity and select the *Fluid (R3) General MIDI SoundFont (GM)* Add-on. This will put a SoundFont in */usr/share/sounds/sf2*. To use this, select

"Soundbank from File" when playing a Pattern.

To convert your songs to MP3, first *Export to MIDI*, then from the command prompt, run:

timidity <song.mid> -c <config.cfg> -Ow -o - | lame - <song.mp3>

You will first need to create a *<config.cfg>* file with the following entry:

soundfont <soundfontfile.sf2>

Please note that timidity will remove any silence from the beginning of the song. If you want to restore some silence, use the following commands:

timidity <song.mid> -c <config.cfg> -Ow -o <song.wav>
sox <song.wav> <song_padded.wav> pad <seconds>
lame <song_padded.wav> <song.mp3>

To apply reverb, open the *.wav* file in ***Audacity*** version 2.0.4 or later and use the *Reverb* effect, then *File/Export* to MP3.

Alternatively, you can *Export to WAV* from ***SongBase***, missing out some of the above steps.

5.1 Ubuntu PPA

To start installing and using software from a Personal Package Archive, you first need to tell Ubuntu where to find the PPA.

Important: The contents of Personal Package Archives are not checked or monitored. You install software from them at your own risk.

If you're using the most recent version of Ubuntu (or any version from Ubuntu 9.10 onwards), you can add a PPA to your system with a single line in your terminal.

Step 1: Open a terminal and enter:

sudo add-apt-repository ppa:t-9nfo-b/songbase

Your system will now fetch the PPA's key. This enables your Ubuntu system to verify that the packages in the PPA have not been interfered with since they were built.

Step 2: Now, as a one-off, you should tell your system to pull down the latest list of software from each archive it knows about, including the PPA you just added:

sudo apt-get update

6 XML Specification

Here is the XML specification of the *.sbxml* files used in *File/XML Export* and *File/XML Import*.

```
<?xml version="1.0"?>

<songs>

  <song>

    <song_name>Song Name</song_name>
    <numeric_duration_type>decimal or pulses</numeric_duration_type>

    <components>

      <component>

        <component_type>pattern or string</component_type>
        <component_name>Pattern or String Name</component_name>
        <string_value>a JFugue MusicString</string_value>

      </component>

    </components>

    <pattern_components>

      <pattern_component>

        <pattern_name>Pattern Name</pattern_name>
        <component_position>an integer value</component_position>
        <component_type>pattern or string</component_type>
        <component_name>Pattern or String Name</component_name>
        <anonymous_string>a JFugue MusicString</anonymous_string>

      </pattern_component>

    </pattern_components>

  </song>

</songs>
```