BelleChat IRC client for Symbian - app showcase

BelleChat is an IRC (Internet Relay Chat) client for Symbian Anna and Belle smartphones written in Qt C++ and QML, and using the Qt Quick Components for Symbian and the Communi library written by J-P Nurmi et al. This article explains the main elements of its design.

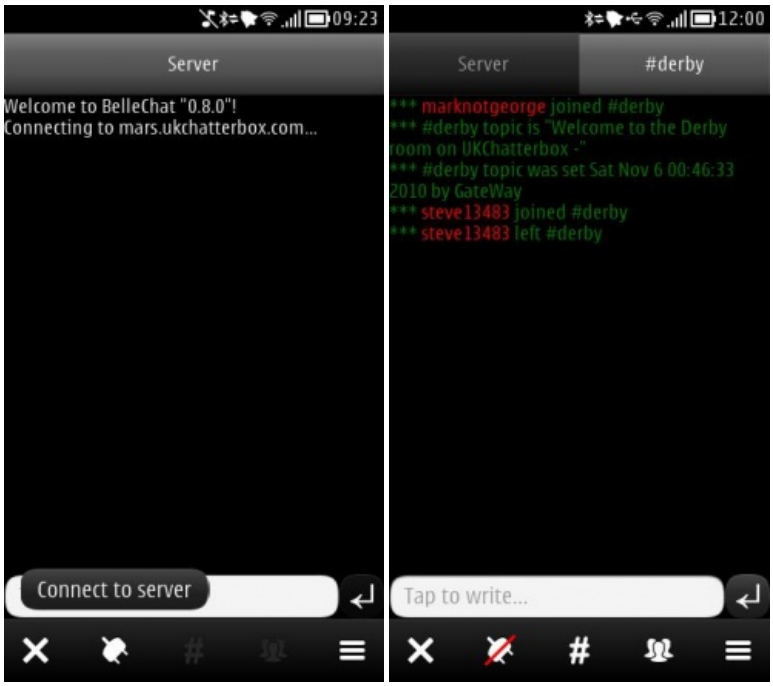
Introduction

BelleChat is an IRC (Internet Relay Chat) client for Symbian Anna and Belle smartphones. It offers the following features:

- Conforms to the Nokia Belle style guide.
- Supports the half-screen virtual keyboard.
- Correctly renders and outputs mIRC colour codes.
- User can join more than one channel at once.

BelleChat is written using QML and the Qt Quick Components for Symbian for the UI, with a C++ backend based on the Communi library written by J-P Nurmi et al. It is available for purchase in the Nokia Store.

Architecture



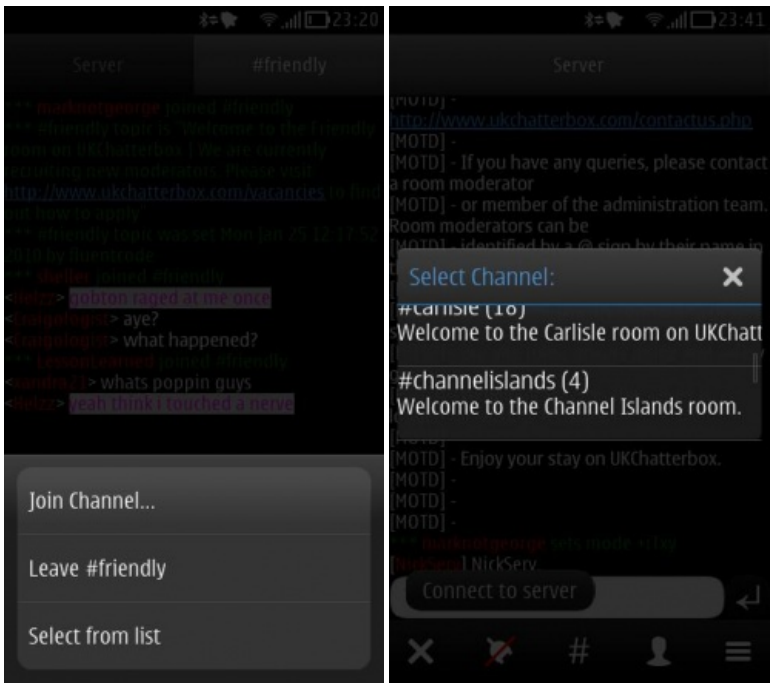
BelleChat is based around a main screen, which is a `PageStackWindow` with a Toolbar and a `TabBar`. The `TabBar` is used to show output from the IRC server on one Tab, with a Tab for each of the IRC channels joined. There is also a `TextField` to enter the user's input, and a Button which mimics pressing the Return or Enter key, on the virtual keypad. Pressing Return or Enter tells BelleChat there's a line of user input to be processed, so a button mimicking Return (by calling the same code that is called when Return is presses) was felt to be useful for when the virtual keyboard is hidden.

Each Tab is based round a `ListView`, which uses a ListModel, and a Text as a delegate. As an output string comes in from the server, it is processed by the C++ backend, which (among other things) parses the mIRC formatting, converting it to HTML. The string is then appended to the List model for that Tab. Using HTML allows the ListView to easily display the mIRC formatting, and also allows clickable links.

The ToolBar consists of 5 buttons, from left to right:

- **Exit**
- **Connect** or **Disconnect** from the IRC server.
- **Channels**, which is disabled when BelleChat is disconnected from an IRC Server. This opens the **Channels Menu**.
- **Users**, which is disabled unless the active Tab is a Channel tab. This opens the **Users Page**.
- More, which opens the **More Menu**. This has two MenuItems: **About BelleChat...** which shows an About Box (a `QueryDialog`), and **Settings**, which opens the **Settings Page**.

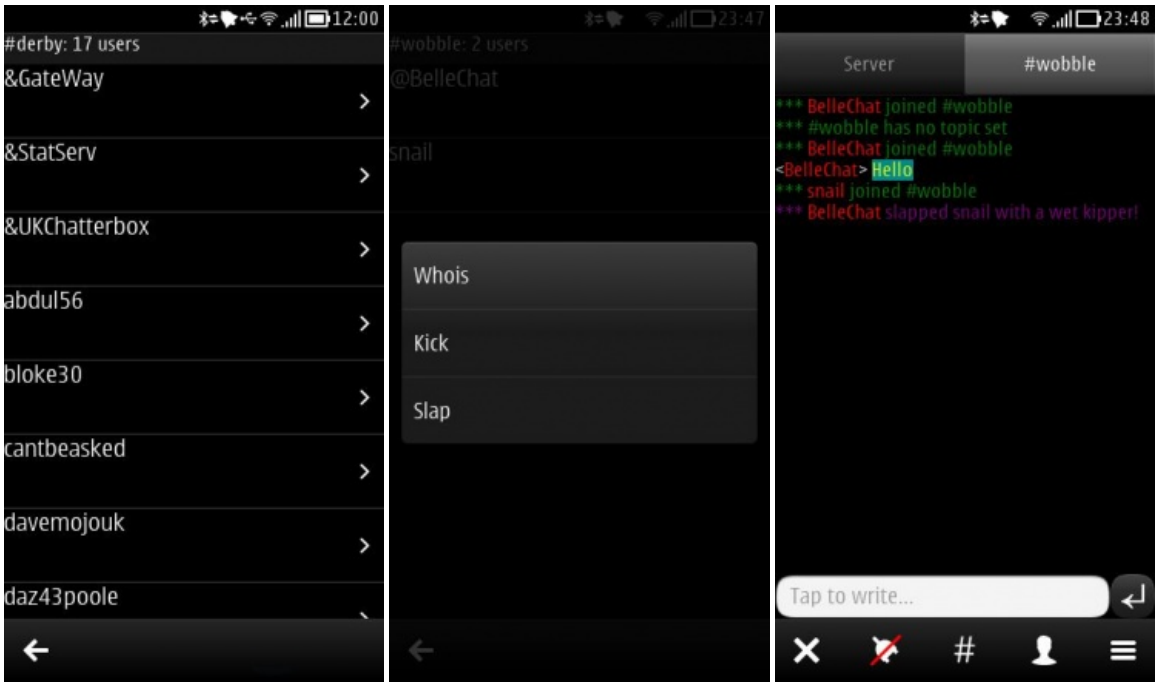
Channels Menu



This has 3 MenuItems:

- **JoinChannel...** This allows the user to type in an IRC channel name. This uses a `TextPickerDialog`, which is a Component I created myself. This is a `TextField` encapsulated in a `CommonDialog`.
- **Leave #<channel>** Only shown when selected Tab is a Channel tab. Allows user to leave the channel.
- **Select from list** This shows a list of all channels and their topics on the IRC server in a `SelectionDialog`. If there are no channels on a server, the user is asked (via a QueryDialog) if they wish to type in a channel name - in IRC, joining a non-existent channel creates it. If the user selects "Yes", a TextPickerDialog opens as above.

Users Page



This is a `Page` with a `ToolBar` (which has only a Back button) and a ListView. The model for the `ListView` is created by the back end, and is a list of the channel's users. The delegate for the `ListView` is a Column of 2 ListItemTexts, one for the user's IRC nickname, and one for the user's IRC realname (only shown if extended user information is available).

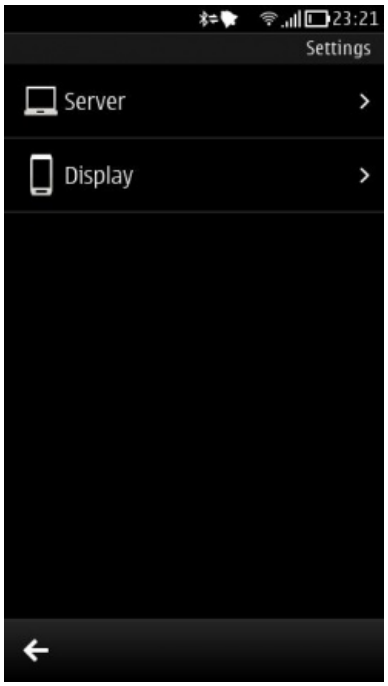
The user can click the delegate to open a **UserDetails** page. This shows the user's WHOIS information, in a Page which consists of a ToolBar with a single Back button, a ListHeading, and a Column of ListItemTexts.

Each delegate also has a `ContextMenu`, which has 3 MenuItems:

- **Whois**, which shows the extended user data as above.
- **Kick**, which sends an IRC KICK command to the server.

- **Slap**, which causes the string *User_1 slaps User_2 with a wet kipper!*. It's a bit of silly fun...

Settings Page

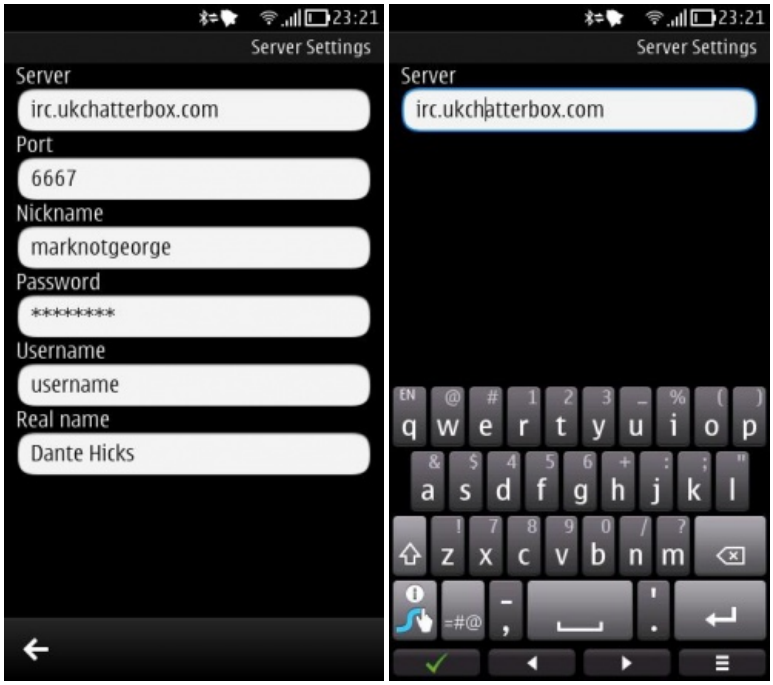


This is a `Page` with a `ToolBar` (with single Back button), and a `ListView` wth two items:

- **Server**, which opens the **Server Settings Page**
- **Display**, which opens the **Display Settings Page**

One of the problems to be overcome when writing the program was the lack of a persistent settings API in QML. To work round this, I created a C++ class which wraps around a QSettings object, and is exposed to QML via a setContextProperty() call.

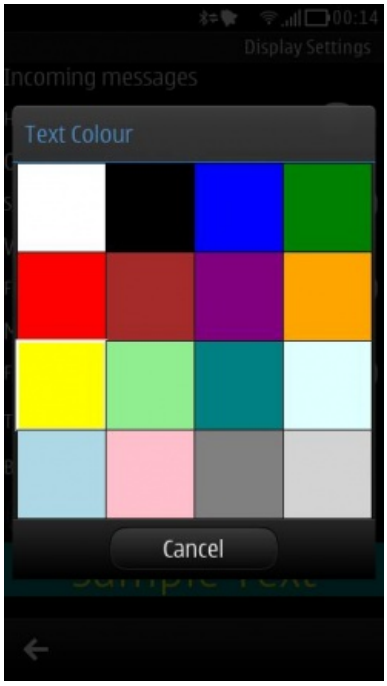
Server Settings Page

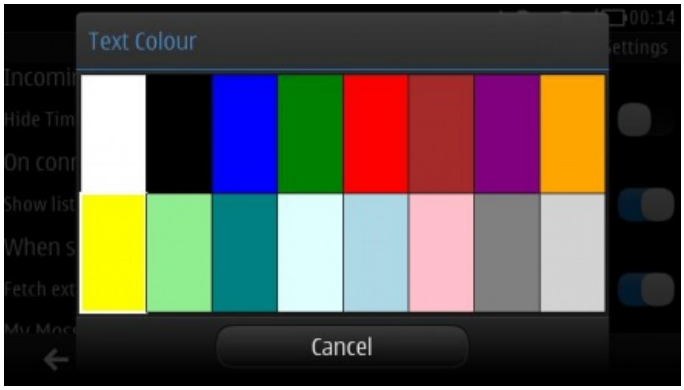


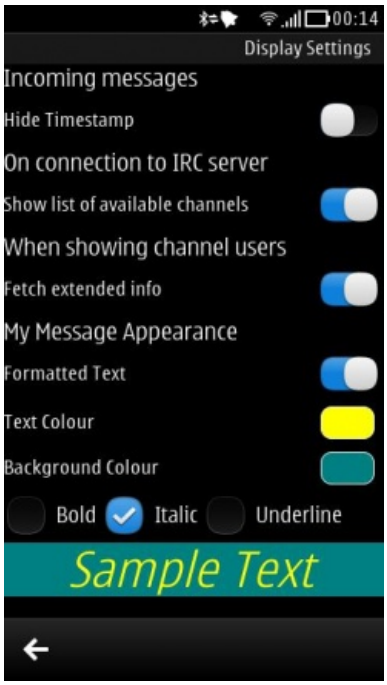
This is a `Flickable` Column of Labels and TextFields, one for each of the servers settings: hostname, port number, nickname, username, password & realname. When the half-screen virtual keyboard is open, the code makes all `TextFields` (and their corresponding Label), except the one which has the input focus, to `visible: false`.

The Toolbar has a single Back button, which saves the settings if they have changed (by means of a boolean variable set to true by any `TextField` which emits its `TextChanged` signal). If BelleChat is connected to the server, the user is warned (by means of a QueryDialog) that settings changes will not apply until the next time he connects.

Display Settings Page







This page concerns itself with settings relating to the display of output: whether or not to display timestamps, whether or not to show the Channel list on connection and whether or not to format the user's input. It does this via two components I wrote myself: the `ColourPicker` and the `LabelledSwitch`.

The `LabelledSwitch` addresses the lack of such a component. It consists of a Switch, with a `ListItemText`, both in a Row. the `ListItemText` has two available strings, depending on whether the Switch is checked or unchecked.

The `ColourPicker` consists of a `Rectangle` with a `MouseArea`. When the `MouseArea` is clicked, a `CommonDialog` opens with a Grid of the 16 mIRC colours. This is created with a Repeater. It correctly handles landscape and portrait screens (showing an 8x2 or a 4x4 grid respectively) and highlights the picked colour. The `Rectangle` also shows the picked colour.

Availability

BelleChat is now available in the Nokia Store. The source code is available at my GitHub repository. Please note, the code is in active development, and may change!

Note: This is an entry in the Symbian Qt Quick Components Competition 2012Q1