1. Self Introduction

1.1 Basic information

- Full name: Wei Xin
- Email:weixindlut@gmail.com or weixindlut@qq.com
- IRC:weixin(@irc://freenode.net #mixxx)
- Skype:weixindlut
- GTalk:weixindlut@gmail.com
- Telephone:+86-15842623380
- Address: Room A0823, Innovation Park Building, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Liaoning, China.
- Languages: Chinese (Native language), English (Fluent), Japanese (Fluent)
- Resume: My English CV
- Github: https://github.com/weixindut
- Major/ specialty at school: VANET /BSN, Embedded System Design

1.2 Programming Skills and Experience

Projects:

- 2012.11 - 2013.2 Chemistry Intelligent Calculate Website, Python,Django

  Build a website using Django, and design an API Engine which is a RESTful API with Piston. Google protobuf as the RPC protocol.

  From this project, I gained a lot of experience on Django, Piston, Google protobuf and Frontend technologies.

  The project is in: https://github.com/chemistryTools/ChemToolsWebService
- 2012-now Intelligent Intersections Management Simulator , C++,JAVA.

  Combine network simulator NS3 and traffic simulator VISSIM through middleware and Socket technologies, build microscopic vehicle model in VISSIM referencing AIM, design algorithm of coordination and collaboration between vehicles and intersections.

  From this project, I enhanced my C++ techniques, though I have had a lot of courses on C++.

  The project is in:https://github.com/weixindut/eAIM
- 2011 Visualization Monitoring Platform of Wireless Sensor Network, Python

TinyOS nesC.Implement runtime control of wireless sensor nodes through EPRC and XML technologies,build graphical interface with PyQt and multi-thread.

  From this project, I got a lot of Qt design experience.

  The project is in:https://github.com/weixindut/WSNTools
- 2009-2011 Embedded works, embedded C.

  Renesas micro-car,wireless extensible LCD,car timer and NetCard module of embedded development board. Mainly be responsible for hardware modular design,circuit design,program design and optimizatio.

  These projects deepened my understanding of computer system from hardware to software.

Technologies:

I have a strong knowledge of Python and C++ which are my favourite lanuages. I have five years' experience in Linux system operation and development and I'm familiar with shell, awk, sed, regular expression and Linux commands. Ubuntu is my favorite Linux distribution. In web development, I am skillful at django 1.4 and I have also studied HTML,JQuery and CSS by myself.

2. General Questions from mixxx

2.1 Top Project of Choice in mixxx gsoc 2013

I have chosen the second project idea "plug and play MIDI Mode/Community MIDI Mappings" in mixxx gsoc 2013.The reasons are:

- The MIDI controller manager is the core of our Mixxx. I think this project is the most suitable one which can make me understand Mixxx soon.
- I love hack so much that I choose the one which I think is one of the most challenging projects in this summer.
- I have a lot of experiences on embedded system development, so I have so much passion on DJ devices.
- I am familiar with C++, Qt, Django, and I think I have ability to complete this project.

2.2 How much have you used Mixxx and for what?

I have used Mixxx for songs integration and created a playlist taken over by Auto DJ.

2.3 What kinds of music do you like?

country, pop, absolute, modern blues

2.4 Why you are the best person for this project？

- I am so interested in Mixxx and mixxx team.
- I have a good knowledge of the relevant techniques which gives me abilities of finishing the project.
- I have read some revelant source code, and based on which I have done a detail design for this project including UI and implementation methods.
- I have some experiences on open source.
- I am a girl loving music, programing and hacking.

2.5 What I have done in mixxx ?

- I have compile and run Mixxx, and read some source code.
- I have installed some other popular DJ softwares and made a comparison with mixxx.
- I discuss the problem that what virtual MIDI devices can work with Mixxx through mailing list and IRC.
- I have read all resources of Mixxx, include wiki, lanunchpad, forums and user maunal.
- I'm trying to fix some bugs.

2.6 How much time can I expect to have for this project?

I will work full time 40h/week during GSoC 2013. My time schedule will be organized as follows:

Table 1: Time Schedule（UTC+8:00）

Day	Time	Task
Monday ~ Friday	8:00 - 10:00	design and write code
	14:30 - 17:30	write code, test and fix bugs
	18:30 - 19:30	mail problems and daily summary and plan
Saturday	10:00 - 12:00	solve technical problems
	14:00 - 17:00	solve technical problems
	19:00 - 22:00	solve technical problems
Sunday	14:00 - 18:00	Blog-post, write technology document, weekly summary and plan, a mail on the mailing list about what I have done in the week.

3. Proposal in detail

3.1 Abstract

Mixxx, a popular free cross-platform DJ mixing software, currently has some features need to design and develop and some bugs need to fix. My proposal is to perfect the preset mapping workflow, as well as develop an api server for users to get presets from mixxx.org or other websites. Expect these two major tasks, I will try to implement software update manager and enhance Language options. Then, a series of tests in different platforms shall be performed. Finally,Some translation tasks,technical documents and manual shall be developed.

Deliverables:

1. Analyze Mixxx code and dive into MIDI controller
2. Automaticly load preset file
    - 2.1 Enhance preset search algorithm
    - 2.2 Adjust the workflow for mixxx initialization
    - 2.3 Modify controller dialog for covert preview
    - 2.4 Popup Mapping Preset Manager
3. Mapping Preset Manager
    - 3.1 search bar for preset files
    - 3.2 local or cloud content for searching result

3.3 Intelligent recommendation system for mapping preset files
3.4 Mapping Cover UI
3.5 Submenu for Mapping Preset Manager, it can download, load more details etc.
3.6 notification board
3.7 users rate and comment on preset files

4. Rest API Engine
   4.1 build a api server on django and piston
   4.2 design protobuf for api response and request formats
   4.3 api/search
   4.4 api/upload
   4.5 api/details
   4.6 api/checkversion
   4.7 api/rate
   4.8 api/download
   4.9 database scheme design
5. Fix bugs:
   5.1 bug/723028
   5.2 bug/1166016
   5.3 bug/909392
   5.4 bug/734704
   5.5 bug/1152930
   5.6 bug/887328, bug/1004989, bug/673237
   5.7 bug/1100882
   5.8 bug/991938
6. Other tasks
   6.1 Software update manager
   6.2 Language Options Enhanced
7. Test
   7.1 write UNIT test scripts
   7.2 test mixxx in different platforms
8. Write technical documents
9. Translation tasks
   9.1 develop a Chinese UI version
   9.2 develop a Japanese UI version
   9.3 translate user manual into Chinese

3.2 Proposal Design

3.2.1 Current version of Mixxx code analysis

By reading the source code, I have a knowledge of the function-call procedure on mapping file preset.I drew a picture like Fig.1:

1. During the initialization, Mixxx tries to load the preset files from local library. If success, the preset will be saved in a data structure; if unsuccess, the data structure keep empty.

2. Apply the preset saved in the data structure.
3. When users open the controller configuration widget, the applypreset() function will be recalled.
4. Since the return results of the two loadPreset() functions in green are not handled, it leads to a problem that the mapping files loaded automatically or manually may don't work and the user may confuse the reason. The reason may be the wrong selections by users or some wrong configurations in the downloaded preset files, but who knows? so I give the following features with the wish of making Mixxx more friendly to users.

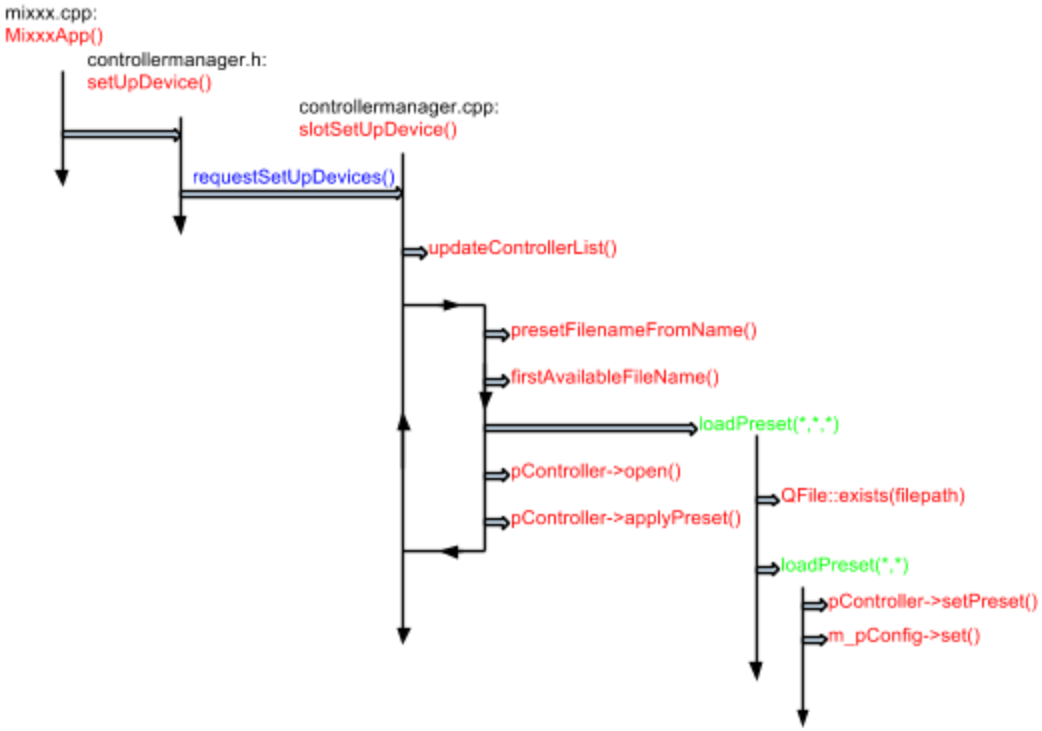


Fig.1 Current controller preset file load procedure

3.2.2 Automatic loading preset file [New Feature]

1. Use Case

When a controller is plugged in and the user needs to do some configurations automatically or manually.Current version Mixxx 1.11.0 provide a semiautomatic way, i.e. if there are mapping files in local library, Mixxx will load it during the initialization, if not, Mixxx will do nothing and users have to select a right preset mapping file from a drop-down list or even build mapping files by themselves.This is a little hard to some initial users, so I design a new small feature it will be a user-friendly design if Mixxx can automitically search local library backend and list the match results for user.

2. Core functions
    - Once a local mapping file isn't searched during the initialization, Mixxx will give a notice message box to user, such as "Sorry,Mixxx cann't select a perfect mapping for you from local library, you can go to Preferences->Controller widget

to get a right one by yourself through our ‘Mapping Preset Manager’”.

- If Mixxx load a preset for the devices at the beginning, the result will be showed on ‘dlgprefcontrollerdlg.ui’ just like the current version. But I do a little change to ‘dlgprefcontrollerdlg.ui’, see Fig.2.
  - I add a device picture. This is very easy to implement, but I think it will help some user a lot. Through this intuitive image, User will easily know whether a right map has be done comparing with their devices at their fist sight of the ‘dlgprefcontrollerdlg.ui’. Of course, if no file can be loaded, there will display nothing.
  - I change the drop-down list to a button with the name “Mapping Preset Manager”. When the button is pushed, another dialog like Fig.3 will be showed.

3. Design and Impletation
   - **UI Mockups**

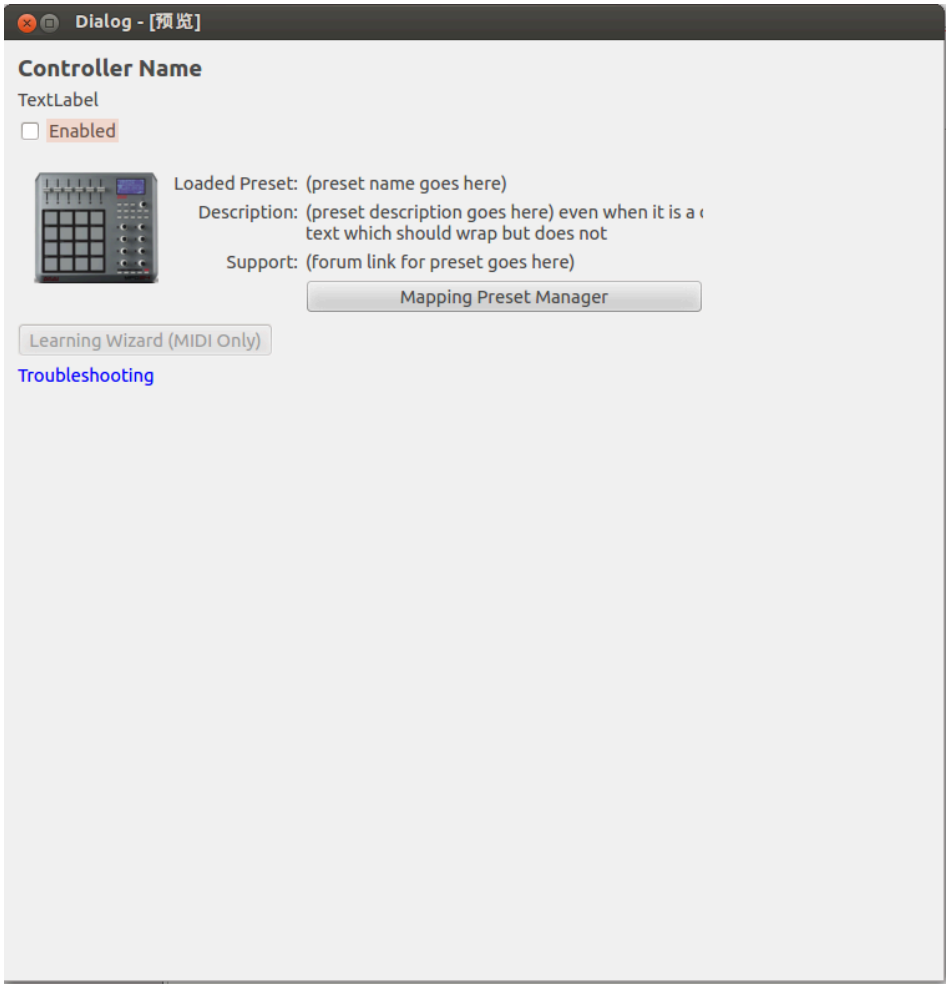


Fig.2 new dlgprefcontrollerdlg.ui

   - **Notice message box**

Based on the current code, a piece of program can be added to handle the return result of 'loadPreset(*,*,*)' in 'controllermanager.cpp'. When a false return, a notice message box will be showed for a friendly guidence to controller configuration dialog.

- **device picture**

  Here I will keep a table for every device to map its unique picture. This will be mainly stated in section 3.2.2.

- **Mapping Preset Manager button**

  _____ The drop-down list will be replaced by this button so that I can introduce my new feature into the Mixxx, as little as possible to change the original code logic.

3.2.3 Search controller mappings via API Engine with Django [New Feature]

1. **Use Case**
   - Mixxx can not do an automical map or an perfect map from local library at the beginning, so users need to choose one by themselves.
   - Mapping files needed by user may exist in local or website, we'd better provide an API for user to query both local and cloud depository.
   - Most users are not very sure which one to choose, so we'd better make an intelligent recommendation for users, especially for novice users, to reduce the sense of fear of using mixxx at the beginning. At the same time, it is also very convenience.
   - A mapping file has been downloaded into local library, but maybe it has some bugs or is not that perfect. And there is a new update version on the mixxx forum or somewhere, so when users try to apply a local preset in the case of network connected, we'd better to advice users to update to a new version of preset.
2. **Core functions**

   Here I think we can create a new feature called **Mapping Files Manager**, with the following functions:
   - A search bar, through which user can get some likely right mapping files coing form local and cloud.
   - A local tab and a cloud tab, which can list the searching results seperately
   - Intelligent recommendation system, which can do an intelligent sort for users searching results and show the results with the most possibility at the top.
   - friendly and cool devices UI display, called mapping cover, which can help users locate the mapping file item quickly.
   - displaying the source(mixxx or mixxx forum or other websites), grade level, certification or not, and the authors.
3. **Design and Impletation in Client Side**
   - **UI Mockups**

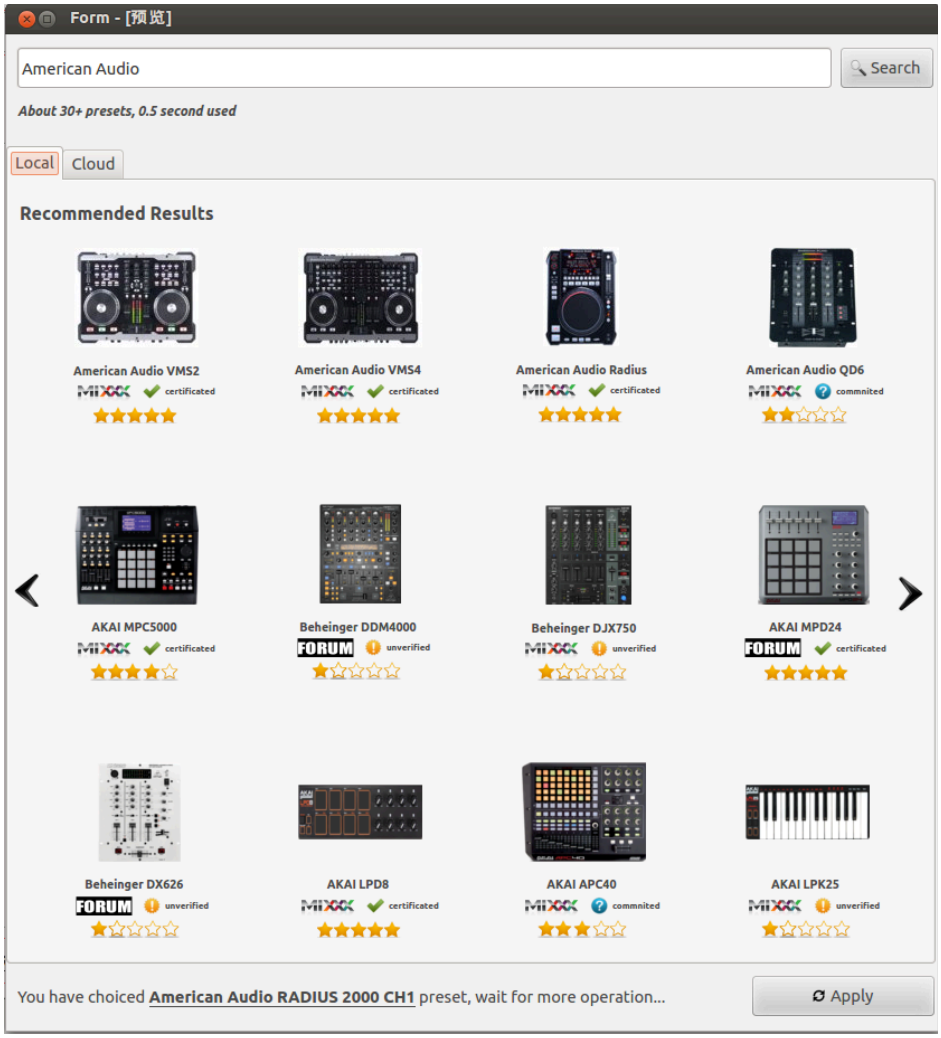


Fig.3 Mapping Preset Manager

- **Search bar**

When users input some querys in search box, such as controller product specification or manufacturers, and click search button, it will search in local database and cloud database via API Engine.

For the search query, we can support a simliar search on device specifications and manufacturers. A simple and fesible implementation method is to keep controllers information tables at local and cloud, and a automatical LIKE match in different fields can be used when users do a search.

For the server side, the corresponding api will receive the query strings and return a set of results consisting of the cover of mapping files, author, link url and grade level.

In default, when users open this dialog, the search box will be filled with device infomation which is detected by our mixxx software, and list some results recommended by mixxx first. It can improve the searching experience.

- **Tap switch between Local and Cloud**

The search results will be divied into two parts: local and cloud. The local panel will show the local mapping files existed in our PC. The cloud panel will list some presets coming from mixxx official, mixxx forum and other websites if the network is accessible . When without a network situation or our api engine server is broken down, we can just show the “network unreachable” message on the cloud panel.

The apperance of cloud panel is the same as the local panel. The only difference is the processing logic when push the appy button down on each panel.

- when users push the apply button on local panel, mixxx will check hiddenly if there are newer versions for the mapping files that user has chosen. If a new version existed in cloud, an suggestion message box will pop out to ask user whether download the newest version. If user choose no, then apply the preset directly, otherwise, do a download and replace the local old version preset, then apply it.
- When users push the apply button on cloud panel, it means that the chosen preset will be downloaded first and then do a map with the user’s device.

The reason why we make them seperately is that user can distinguish which presets have been downloaded into local PC and which ones are not. The most important reason is that the implementation logic is more clear.

- **Search results pannels**

The results sets got from api/search interface wil be listed here in order. In order to provide a more intuitive and friendly appearence, I design a scrolled area to wrap the search results (See Fig.3) instead of the original mappling presets drop-down list.

- At the top of the panel, the result statistics info label, including results number, total time etc. will be shown.
- Every result item is not just a text any more, but with a mapping-cover, grade level information, source information and certification information.
- Authors, URLs, brief description and other information will be display when mouse hovers on the item surface.
- Left Click: choose one preset.
- Double click: apply it directly.
- Right click: a popup menu which contains “More Information”, “Download”, “Reload” and “upload” items will appear.

Mixxx does an intellignet recommedation according some algorithm for user. The items which match the query best ,own the highest grade level, and has been certificated may be listed at the top location. Of course, the sort algorithm need a more detail and professional design.

- **Notification board and apply button**

At the bottom of the widget, I add a notification board which will show some useful message to help users know the status of Mapping Files Manager, network accessiblity, download progress or operation notifications.

Apply button will be enabled after the use choice one preset item, I have stated in section “Tap switch between Local and Cloud ” that its processing logic may be a little

different on local panel and cloud panel.

- **Local data-model**

_____Local data-model is so simple that three tables are enough. Local Models only update information from server. When some newer presets are shown in API Engine, database will add these new records without replacing the existing records.

i. MappingPresetObject

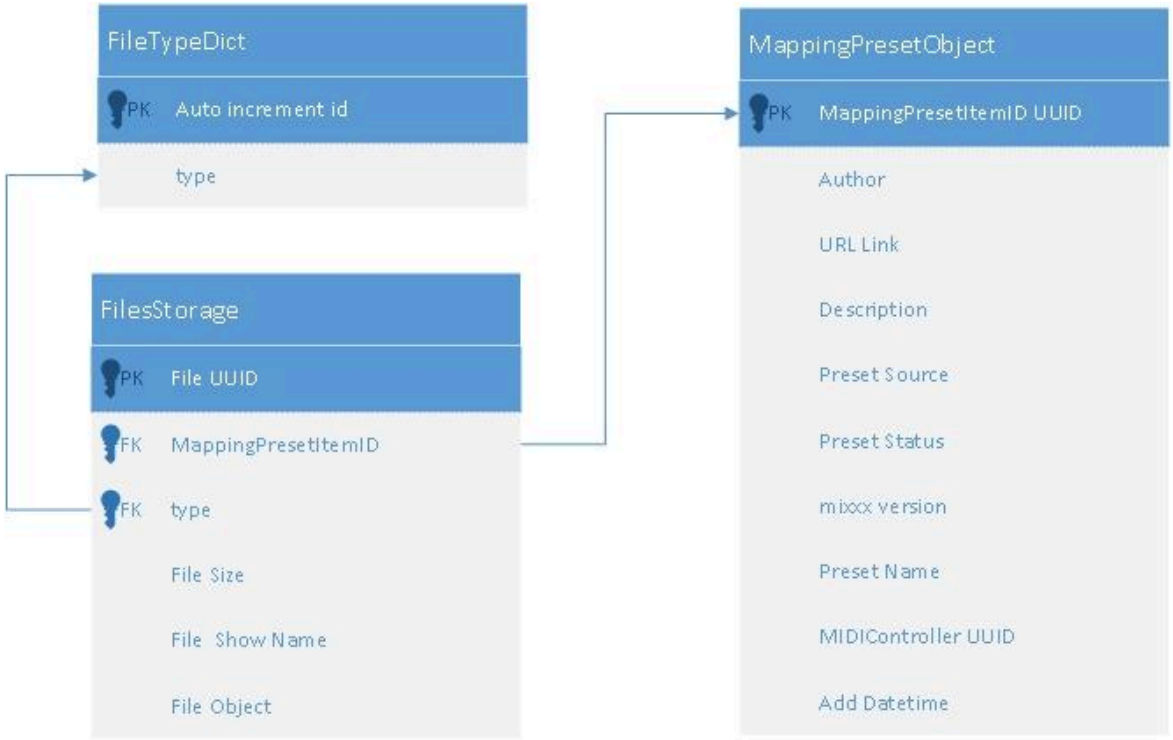
* Description: In this table, many fields can get from server side directly. Every update query will send “MappingPresetItemID” to API Engine.

ii. FilesStroage

* Description: MappingPresetObject keeps an one-to-many relationship with this object, because an preset may include many files with diffirent types. File Object field keeps the file path.

iii. FileTypeDict

* Description:the type of some resources used for presets,the values of type include XML,JS,JPG(used for device cover) etc.



4. **Design and Impletation in Server Side**

- **Principles and RestFul API**

In order to achieve API Engine Server, we use REST API with Django, Piston and Google Protobuf.

- Django: A famous full-stack framework in python.
- Piston: A RESTFul API App for django. It makes REST API very simple and tidy.
- Google Protobuf: A high performance RPC with cross-language support. I

found our mixxx org has already used protobuf.

I have already used these techniques in my previous projects, and they together worked fine.

- **API Details**
  i. **api/search**
     * HTTP Methods: GET
     * Description: Return a set of mapping preset results according to the search query.
     * Response Formats: mapping-cover picture, controller name, link url, author, grade level, whether is certificated or not, mixxx official or forum, the uuid of the mapping preset(It is unique id.)
     * Request Formats: search query string
  ii. **api/upload**
     * HTTP Methods: POST
     * Description: When a user makes a mapping preset file by our learning wizard or manually, he can upload it to our server. Administrator will check the quality of the mapping preset and republish it.
     * Response Formats: True or False (upload state)
     * Request Formats: controller name, manufacturers, version, year, mapping file(.xml, .js and picture), description, author
  iii. **api/details**
     * HTTP Methods: GET
     * Description: When a user clicks “More Information” item in popup menu, it will get more details on this preset from our server.
     * Response Formats: description, year, version, comments for user groups etc.
     * Request Formats: the uuid of mapping preset
  iv. **api/checkversion**
     * HTTP Methods: GET
     * Description: check one mapping preset version. If there is a newer for the current software version, the api will return True. If not, return False.
     * Response Formats: True or False.(To decide whether to update it.)
     * Request Formats: the uuid of mapping preset, mixxx software version
  v. **api/rate**
     * HTTP Methods: POST
     * Description: When a user has used a mapping preset for a period of time, he can make an objective evaluation on it as a feedback to our api engine. This let user to feel our care for their ongoing experience.
     * Response Formats: True or False. (Whether or not successful)
     * Request Formats: the uuid of mapping preset, mixxx software version, grade level(1~5, like playlist rate), comments
  vi. **api/download**
     * HTTP Methods: GET

* Description: When a newer version or cloud version is chosen, the manager will download it by the uuid of mapping preset.

* Response Formats: a set of mapping preset files(*.xml, *.js)

* Request Formats: the uuid of mapping preset

- **Server-side data model**
  i. **MappingPresetObject**

* Author: The ID of the user who builds this preset.

* URL Link: A webpage of mapping preset , which includes exhaustive information.

* Description:

* Preset Source: from mixxx, forums or others

* Preset Status: whether has been certified

* mixxx version: this preset will adapt the specific mixxx version. The same MIDI Controller maybe have different preset for different mixxx version, a historical issue.

* Preset name: show name

* MIDIController: the controller device ID.It is a foreign key.

* Add Datetime: used for presets update, it can decide whether to update preset or not. Comparing Add Datetime value with that of query from client side, if client value is not older than sever side values, update progress is not needed, otherwise, do an update.

  ii. **FilesStorage**

This model will take charge of the storage of mapping preset files , which include skins, XML, JS or other additional files.

* File UUID: user uuid, it is a primary key for every file.

* MappingPresetItemID: It is a foreign key which is the mapping preset object. it is One to Many relation, every mapping preset object is connected to many related files.

* type: it is a foreign key, which will storage the type of this file, not the postfix, it is the type of resources.

* File Size:

* File Show Name:

* File Object:It is a file object , in django this field is a FileField which includes file storage path, verbose name etc. The file itself will be stored in file server.

  iii. **PresetComments**

This model will store all comments for preset files.The rating of one specific mapping preset will be calculated through this table. One mapping preset object will map many comments.

iv. **UserInformation**

_____Some information for the preset author. We will introduce the account information from mixxx forum. Current data model doesn't include the part of user authority.

v. **MIDIController & MIDICompanyDict**

_____* The two models show the midi devices information. Controller UUID explains the id of every specifical device.

* Controller Name field and Company field are the composite unique.

vi. **MixxxUpdate**

_____This model shows the software update information. The client can update its software by this model. By the "download url" field,Software Update Manager can make an update.

vii. **MappingPresetSourceDict**

_____A dict which can show the source of mapping preset, such as mixxx official, mixxx forum or others.

viii. **CertificatedOperationDict**

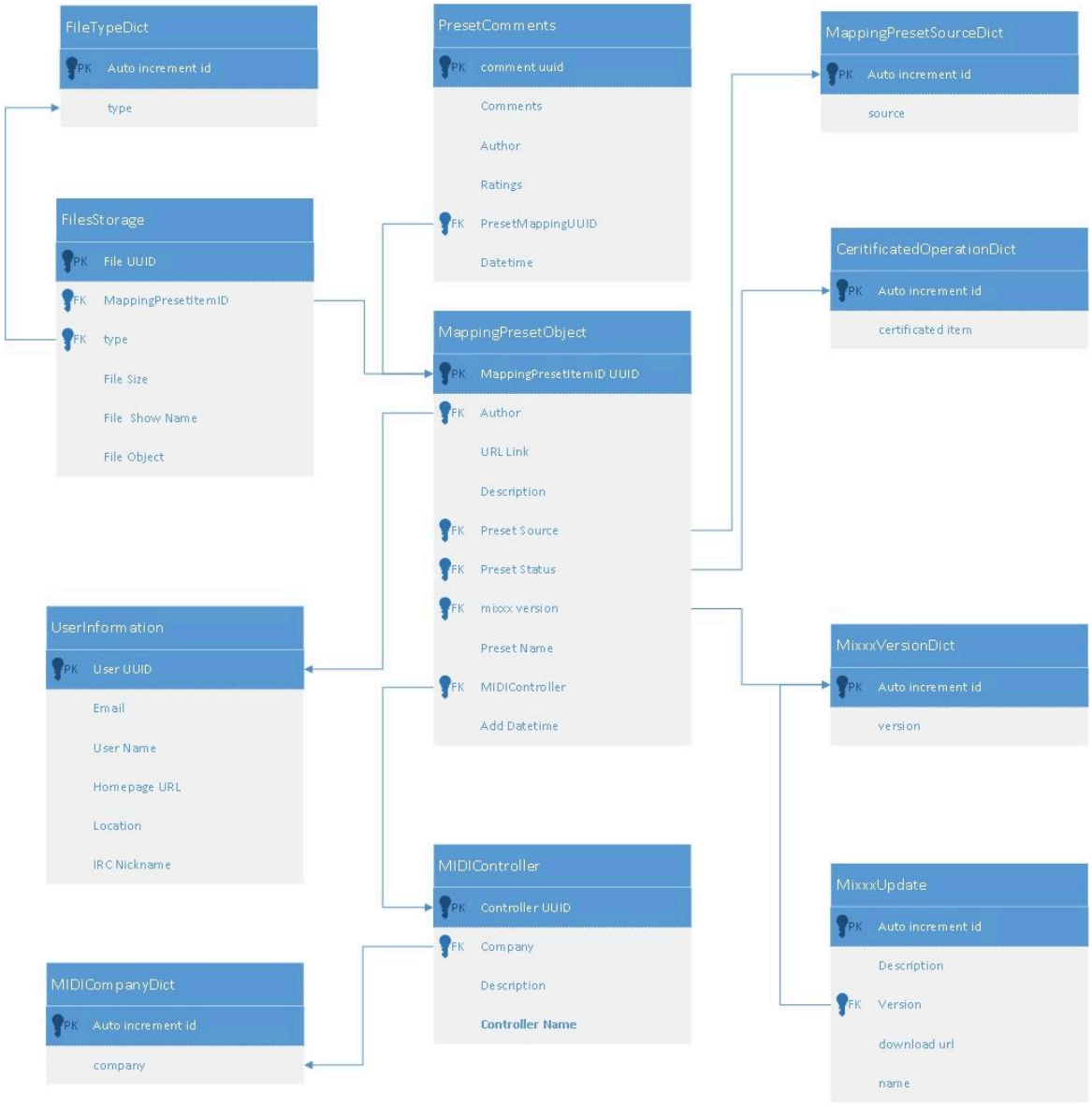
_____A dict telling whether has certified.

ix. **FileTypeDict**

_____A dict show the type of preset files, such as skins, js, xml, configure file .

x. **MixxxVersionDict**

_____A dict of the mixxx version, such as 1.10.0, 1.10.1 etc.



3.2.4 Fix bugs [Enhancement, Fix bugs]

I will fix some bugs from launchpad, some bugs are related to the MIDI controller and some are only need to be fixed.:

1. https://bugs.launchpad.net/mixxx/+bug/723028
   Export is very useful for MIDI controller bindings, I will export the binings as XML file.
2. https://bugs.launchpad.net/mixxx/+bug/1166016
   Adjust the data structure of controller script hash table.
3. https://bugs.launchpad.net/mixxx/+bug/909392
   Enhance controller mapping JS parse.
4. https://bugs.launchpad.net/mixxx/+bug/734704
   Enhance MIDI script engine outputs.
5. https://bugs.launchpad.net/mixxx/+bug/1152930

Easy bug , but it is necessary.

6. https://bugs.launchpad.net/mixxx/+bug/887328
   https://bugs.launchpad.net/mixxx/+bug/1004989
   https://bugs.launchpad.net/mixxx/+bug/673237
   They are all the missing song issues.
7. https://bugs.launchpad.net/mixxx/+bug/1100882
   Color the deck, it is a cool feature.
8. https://bugs.launchpad.net/mixxx/+bug/991938
   Add total number, it is so easy but nobody fixes it.

3.2.5 Other necessary tasks[Enhancement, new features]

1. **Software update manager**

Software update manager is necessary for mixxx. Mixxx must be able to check and download plugins or upgrade-packages(tarball files) from our API Server Engine automactically or manually. Also it can replcae the older files automatically.

2. **Language Options Enhanced**

When users install Mixxx, they can choose the default language,for example English, Chinese and Spanish, Portuguese etc. They also can change the software language in Preferences submenu -> language options.

3.3 Test in different platforms

3.3.1 Unit Tests

Use QT unit test tool to test new features, and write some scripts for it.

3.3.2 Different platform tests

I will test the new features of mixxx software in different platforms , such as Windows 7 32bit and Windows7 64bit, Ubuntu 12.04, 13.04, 13.10.

3.4 Write technical documents

3.5 Translation Chinese and Japanese

3.5.1 develop a Chinese UI version

Chinese is my native lanuage, I can translate a professional Chinese UI version for Mixxx, it will affect many protential users.

3.5.2 develop a Japanese UI version

I also master the Japanese and I have obtained Japanese Language Proficiency Test, Level 2 certification.

3.5.3 translate user manual into chinese

Besides Chinese UI, I will also translate our user manual into Chinese, mixxx may be introducted into China successfully!

4. Timeline

Each task is addressed to each week of Google Summer of Code 2013, Starting from

May 8 to September 27. In Table 2, each task deadline is discriminated.

Table 2: Google Summer of Code 2013 deadlines

Deadline	Tasks
May 28	1.Setup the development environment 2.Read documents, dive into mixxx source code
June 2	Talk with mentors the changes or adaptations
9	Analyze Mixxx code and dive into MIDI controller
16	1.Enhance preset search algorithm 2.Adjust the workflow for mixxx initialization 3.Modify controller dialog for covert preview 4.Popup Mapping Preset Manager
23	1.search bar for preset files 2.local or cloud content for searching result
30	1.Intelligent recommendation system for mapping preset files 2.Mapping Cover UI
July 7	1.Submenu for Mapping Preset Manager, it can download, load more details etc. 2.notification board 3.users rate and comment on preset files
14	1.build a api server on django and piston 2.design protobuf for api response and request formats
17	1.api/search 2.api/upload 3.api/details
21	1.api/checkversion 2.api/rate 3.api/download 4.database scheme design
28	Language Options Enhanced
August 4	1.Write Mid-term evaluations report 2.Software update manager

11	1.Solve the inscrutable technical problems Part1 2.Fix bugs
18	1.Solve the inscrutable technical problems Part2 2.Fix bugs
25	1.write UNIT test scripts 2.Fix bugs
September 1	1.test mixxx in different platforms 2.Fix bugs
8	Documentation: New features and developer (English and Chinese)
15	1.Documentation: user manual (English and Chinese) 2.Languages: make a Chinese and Japanese UI version .
27	Write Final evaluating

5. Additional Information

1. How do you plan to interact and create a closer relation with the project's community?

In this summer of code, I will be always available at IRC which I will check at least once per day so any message will reach me within 24-hours. I also have email configured on my mobile phone, so I can check emails within a few minutes unless I'm dozing off. Every weekend, I'll write both end-user and technology documents. I will submit a blog-post or a mail on the mailing list about what I have done in the week and what I want to do in next week. If I have some urgent issues and have to leave for a while , I will inform my mentors in advance.

2. After Gsoc 2013, what I will to do?

As a volunteer , I want to be a true contributor in Mixxx and code on after GSOC . I will try my best to discuss the project problems in the mailing list, IRC , forums and launchpad. I will always pay close attention to the new features which are submitted in the launchpad.

No software is perfect but pretty error prone, so maintenance is necessary. I really want to contribute to open source development. I have looked through all the project pages especially the ideas pages and I agree with the development ideas of the Mixxx Open Source Organization. I want to work with Mixxx Organization and I hope to become a member of the Mixxx Project!

3. Why we move code into Github?

As everyone knows, Github is the most social code organization, and many famous open

source project are all in it! And GIT is the most powerful distributed version control system. I have seen some emails in our mixxx maillist which are focused on this task. I am interested in convert bzr into git repository. Maybe in my part time , I can make some efforts on this.