Appion - Feature #345

automatic database conversion on release update

04/12/2010 03:36 PM - Anchi Cheng

| Status:     | Closed                  | Start date: | 04/12/2010 |
| Priority:   | High                    | Due date:   |            |
| Assignee:   | Amber Herold            | % Done:     | 0%         |
| Category:   | Installation            | Estimated time: | 0.00 hour |
| Target version: | Appion/Leginon Future Version | Spent time: | 0.00 hour |

Description

When database need updating, a conversion script is needed. Here is some of our past communications:

From Anchi

I think we should make an effort of creating conversion script, testing on testing databases as we did with the user login effort, before the change is made to the real database and make sure that the conversion script is in svn to give Scott a chance to update easier. I know it might sound much for one person but we will likely do this again in the future, so developing a good strategy will be worth while.

Anchi

From Eric

That's right! We might also want to think about not to dynamic create tables and field in the DB. Therefore all the database tables creation will be in default XML files. So we can write db upgrade scripts base on XML version number. We can also create a folder to contain all the default xml file and upgrade scripts. Right now the xml files are in couple different places.

Thanks.

Eric

Related issues:

- Related to Appion - Feature #344: Create database conversion script
  - Closed 04/12/2010
- Related to Appion - Bug #495: Create a new wizard path for people doing upgrade.
  - Closed 05/17/2010
- Related to Appion - Bug #328: Major Appion database schema change
  - Closed 04/07/2010
- Related to Appion - Bug #1148: DEF_timestamp is NULL
  - Closed 01/18/2011
- Blocks Appion - Bug #333: Make all tables compatible with sinedon
  - New 04/08/2010

Associated revisions

Revision b990d423 - 12/13/2010 09:36 PM - Anchi Cheng
First iteration of schema update base class that look up and update the database revision record refs #345

Revision 6262dd1b - 12/15/2010 08:41 PM - Anchi Cheng
get version and branch from svn info if available refs #345

Revision c17ae0c7 - 12/15/2010 09:20 PM - Anchi Cheng
more types of needUpdate results refs #345

Revision b6ebb7d3 - 01/20/2011 07:11 PM - Anchi Cheng
get release revision from xml file refs #345
make semi-automated schema update work with releases, too refs #345

add revision log in database for branch 2.0 refs #345

I do not agree that we should go away from dynamically creating tables and fields from sinedon. It will slow down the development because the proper formatting xml or mysql files from scratch is not the strength of most of our developers. For new installation, Neil's "create tables" script should be a good start for our current initialization problem. xml file is still needed since not every table in every database we deal with are defined in sinedon format, nor they belong in there. For example, stuff in projectdb and viewer stuff. If it is desirable to combine the two, the appiondata.py should be the source of information, and an automatic script creates xml entry that can be appended to the default.xml

For old installation and for our conversion, the xml file would be better created through a look up of difference in appiondata.py. This way, new changes can be added easily.

Conversely, we could design the PHP to parse the python database file and the python could automatically migrate the data.

The problem is that we are defining the database schema in two incompatible places.

Why shouldn't the projectdb and viewer stuff conform to the sinedon standard. It annoys the crap out me that it does not. Ugh!!!

Is this happening?

r15139 contains a few idea that hopefully will work:

1. schemabase.py is the base class that allows subclasses written without worrying about checking previous upgrade and update the database revision.
2. schema-r14891.py is an example subclass that replaced the original schema-r14891.py
3. schema_update.py will be the script we uniformly have our developers to run to find out what schema update they need to run still to be compatible to their check out. Eventually we can make it run a series of schema-rxxxxx.py.

4. updatelib.py contains functions that may be needed both by SchemaUpdate() base class and schema_update.py
   schema_revisions list at the beginning of the file determines the ordered sequence in which schema_update.py to run and which schema update need to go before the one that is being considered.

My vision is that any of us who want to update database for new features that can cause an incompatibility with the older database in the ways that it breaks a general query or existing myamibase page to write one of these schema-rxxxxx.py. He/She also need to add the schema revision number in updatelib.py to the list of schema_revision. This way the others can run schema_update.py when they noticed a possible schema change.

Gabe, Scott, and Neil:

I especially need your input and to try this out so that all of us can develop without breaking each other's sandbox or real database.

#11 - 12/13/2010 07:24 PM - Anchi Cheng
r15142 has another example used to add disabled group of users.

#12 - 12/14/2010 07:01 AM - Scott Stagg
I love the idea of a script that automatically mends any problems associated with the database updates, but I have many questions and comments.

1) How does one know when to run schema_update.py? Do you run it every time you svn update or do you wait to see if something is broken?

2) Once one runs schema_update.py, does it make myami incompatible with previous versions?

3) If 2 is the case, am I correct that it is not possible to have a stable version and a developer version?

4) I would love to try this out, but I am worried about breaking things. It is very important to me to have a working version at all times. I'm sure the same is true at Scripps. How do y'all do database tests without interrupting the normal workflow?

5) I'm afraid that your answer to 4 is that you have database clones that you do testing on. I would love to do something like that, but it is very difficult for me to get stuff like that set up.

#13 - 12/14/2010 11:27 AM - Anchi Cheng
Scott,

1. Yes, you should run schema_update.py every time if you want to know if there is an update schema for you. schema_update.py only lists what scripts need to be run in order, it does not run them automatically. The point of the script is so that we don't have to announce a new schema-rxxxxx.py is available every time to each other.

2. Thus far we have not had any change of schema that prevents previous myami versions to be used on the updated database. We will keep it this way as long as possible. The schema update handles in general the ability of using new feature and to mend bugs.

The two examples that use the schemabase.py are simpler cases. schema-r14891.py is a bug fix for those install from scratch of earlier myami. schema-r15069.py is to add new rows of data so that new feature can be used. Neither of these types of database schema update will affect the usage of the older myami.

The worse case will be when a rename of field or table is proposed. For example, Jim wanted to rename some fields to better organize certain Leginon settings. Instead of doing an renaming of the field in the database, which will break the old version, he added new fields and uses only the new fields in future Leginon. This way, off-site developer like you can continue to use your sandbox once schema-rxxxxx.py is run which copy/translate the old fields to the new if needed. the general users can continue to use the stable myami off the same database. We will have to create a method to force this kind of update (currently, you can not do the same schema update twice) so that when everyone is ready to move on, the schema-rxxxxx.py can be performed again. In the particular case of Leginon node settings, if you run older, stable version after running the new version and changed settings, there might be a problem since the new version will not fill in the old fields. Therefore, if you are using updated sandbox, you'd better stay on it in terms of a unique Leginon user in this particular case. Would this be acceptable to you?

On Appion side, we also have to worry about the old version of myamiweb can not read the results of new feature, although I am not as concerned about it as long as it is not a replacement.

3. trunk is the developer version. Branch is the stable. These schema update hopefully will keep both usable at the same time even though not by the same person from Leginon's point of view because of node settings. I don't think Appion side would be a problem in this parallel universe.

4. Jim and I still develop Leginon on the working database and both of us stay on the trunk. I do switch to the branches to merge in debug and check that it works there. Have not had problem yet. If there is serious change that likely breaks the older query like a real renaming (which we would like to avoid until maybe next major release (3.0). We are doing minor releases since 2.0), a copied database will be created for testing. I also have my minimal test database that I can recreate easily for testing schema update script.

5. If you know what you are adding as a developer, you are unlikely to cause disaster even on your working database by adding features. If it is still a concern to you and prevent you from development, we can help you create the minimal database that ties to your trunk sandbox, too, for initial testing of your development.

#14 - 12/14/2010 11:55 AM - Scott Stagg
If I understand correctly, the only time I could get in trouble is if a user tried the trunk version of Leginon after a schema update, used it for a while, then tried to go back to the stable version. That seems fine with me. As long as I can have stable and developer versions of myami existing on the same database, I am happy. I will try out the schema_update.py script today or tomorrow.

#15 - 12/14/2010 04:18 PM - Anchi Cheng
That is what I can see as the only problem thus far. When you are done with this wave of update, you should only see one change: You will get a new user group called Disabled. If you assign someone in that group, he/she will not be able to login any more on the current version of myamiweb. With older version, it behaves like Guests.

#16 - 12/15/2010 02:55 PM - Scott Stagg
I get this error on running schema_update.py:

```python
connected to db 'projectdata' on server 'zeus'
failed to get an integer revision
Traceback (most recent call last):
  File "/schema_update.py", line 12, in <module>
    check_out_revision = updatelib.getCheckOutRevision()
  File "/home/sstagg/myami_dev/dbschema/updatelib.py", line 18, in getCheckOutRevision
    return false
NameError: global name 'false' is not defined
```

#17 - 12/15/2010 06:24 PM - Anchi Cheng
r15174 added a missed modification in leginon/version.py to check check out version and branch
r15177 correct false to False and add more types of needUpdate cases

Sorry, Scott, for the bugs. It should work now.

#18 - 12/15/2010 07:10 PM - Scott Stagg
Now I get this error:
connected to db 'projectdata' on server 'zeus'
failed to get an integer revision
Traceback (most recent call last):
File "/schema_update.py", line 13, in <module>
schema_revisions = updatelib.getUpdateRevisionSequence()
File "/home/sstagg/myami_dev/dbschema/updatelib.py", line 7, in getUpdateRevisionSequence
if version.getSVNBranch() == 'trunk':
  AttributeError: 'module' object has no attribute 'getSVNBranch'

#19 - 12/15/2010 07:56 PM - Scott Stagg
Whoops, sorry, I hadn't updated my whole sandbox. After I updated, it worked with this output:

```sql
CUSTOM: SELECT value FROM `install` WHERE `key`='revision'
Already Up to Date for schema r12857
CUSTOM: SELECT value FROM `install` WHERE `key`='revision'
Already Up to Date for schema r13713
CUSTOM: SELECT value FROM `install` WHERE `key`='revision'
Already Up to Date for schema r14077
CUSTOM: SELECT value FROM `install` WHERE `key`='revision'
CUSTOM: SELECT value FROM `install` WHERE `key`='revision'
You must successfully run schema-r14891.py first
------Copy the following lines to command terminal to run -------
python schema-r14891.py
python schema-r15069.py
Exception TypeError: TypeError("'NoneType' object is not callable",) in <bound method Popen.__del__ of <subprocess.Popen object at 0x13eef10>> ignored
```

There was some error at the end, but it didn't seem to effect how the program ran.

#20 - 12/15/2010 07:58 PM - Scott Stagg
The second update executed but had this error:

```bash
sstagg@zeus:~/myami_dev/dbschema$ python schema-r15069.py
connected to db 'projectdata' on server 'zeus'
connected to db 'dbemdata' on server 'zeus'
CUSTOM: SELECT value FROM `install` WHERE `key`='revision'
```
Upgrading dbemdata

---

Successful Update

---

CUSTOM: SELECT value FROM `install` WHERE `key`='revision'

---

Upgrading dbemdata

---

CUSTOM: SELECT * FROM `install` WHERE `key`='revision'

UPDATE: UPDATE `install` SET `value`='15069' WHERE install.key = 'revision';

Exception TypeError: TypeError("'NoneType' object is not callable",) in <bound method Popen.__del__ of <subprocess.Popen object at 0x1519350>> ignored

#21 - 12/16/2010 12:57 PM - Anchi Cheng

Scott,

Could you try to run version.py in leginon folder to see if you get the same error? I can't reproduce it here and version.py is the only place I can tell that opens a subprocess. The output is just a couple of svn information.

#22 - 12/16/2010 12:59 PM - Scott Stagg

That works, and I get the following:

```
sstagg@zeus:~/myami_dev/leginon$ python version.py
15183
trunk
/home/sstagg/myami_dev/leginon
```

#23 - 12/16/2010 01:03 PM - Anchi Cheng

I am lost, then. Jim?

#24 - 01/20/2011 06:00 PM - Anchi Cheng

r15299 make it possible to check and update on releases, too. Also can save package revision as the last script run.

This will be merged into branch 2.0 and branch 2.1 so that we can patch bug #1148 on damaged databases

#25 - 02/02/2011 02:11 PM - Amber Herold

I tested this feature, such that one update script needed to be run. After running the script I reran schema_update.py and it correctly told me to run the version update script. The version update worked. However, it is not clear why we have 2 database entries or the revision now. Under Project->install I have both "revision" and "Revision". Both of these fields were updated to the current revision of my installation.

#26 - 02/02/2011 06:24 PM - Anchi Cheng

- Status changed from In Code Review to Assigned
- Assignee changed from Jim Pulokas to Eric Hou

Eric,

In install/newDBsetup.php, revision is entered with key 'Revision', while in /myamiweb/xml/projectDefaultValues.xml, which is used in Wizard, revision is entered with key 'revision'. Unaware of your format in newDBsetup.php, I have been following what was found from running Wizard, which is why after update script is run, there are two in the database. it will need to be consistent. Maybe another schema_update script need to be written?

#27 - 02/04/2011 12:59 PM - Eric Hou

- Priority changed from Normal to High

#28 - 05/12/2011 01:53 PM - Eric Hou

- Assignee changed from Eric Hou to Amber Herold

#29 - 08/13/2014 05:36 PM - Amber Herold

- Status changed from Assigned to Closed

fixed in r18201