

Using MySQL in Plone

Scott Beardsley <scott@cse.ucdavis.edu>

CCSE at UC Davis

Plone Users Group of Davis

June 2007

Why?

- Legacy and Common Architectures
- Access data in non-python languages
- Easy ad-hoc reporting
- ZODB optimized for reads not writes
- Relational has a proven track record

Requirements

- Python MySQL
 - Python module
- Zope MySQL Database Adapter
 - aka ZMySQLDA
 - Zope product

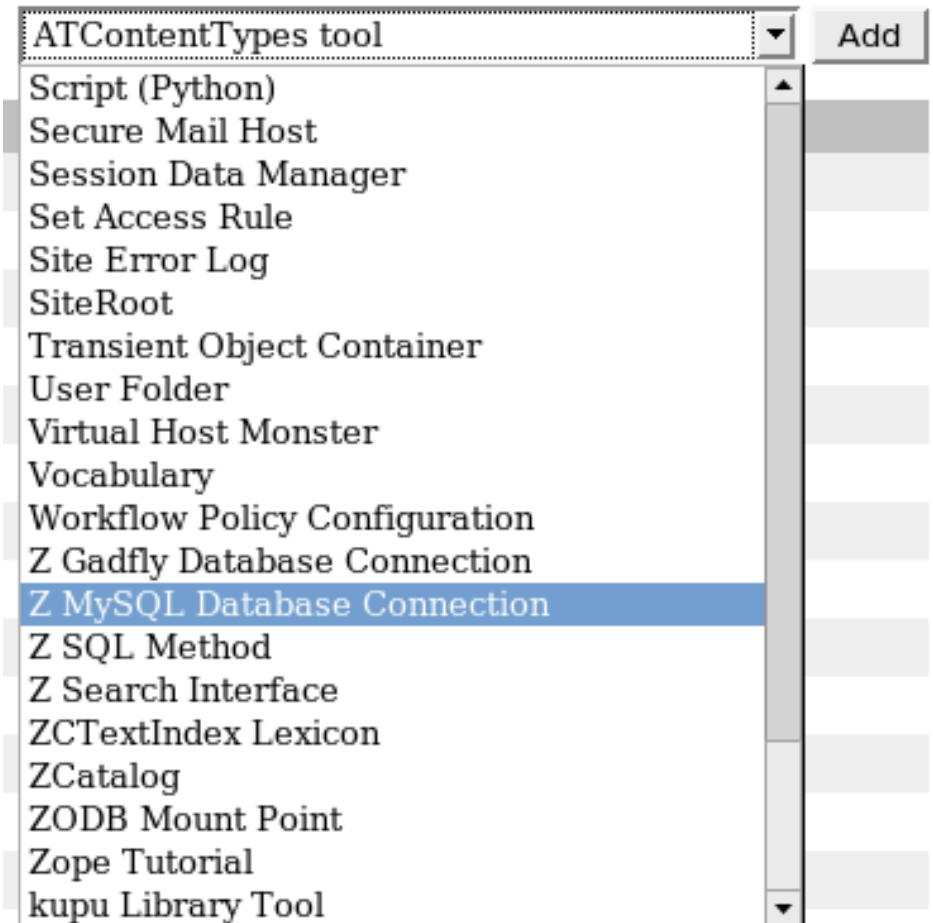
<http://sourceforge.net/projects/mysql-python>

Prepare MySQL

- Insure MySQL server is running
- Add a new user for zope
 - `mysql -p root -e "GRANT SELECT, INSERT, UPDATE, DELETE ON mydb.* TO 'zope'@'localhost' IDENTIFIED BY 'zope';"`
 - `mysql -p root -e "FLUSH PRIVILEGES;"`
- Test database access

Link Zope to MySQL

- Add a Z MySQL DB Connection from the ZMI
- Connect String
 - mydb zope zope
 - `[+/-]database[@host[:port]]`
`[user [password [unix_socket]]]`
- Security
 - Standard (ie powerful)
Plone security options
- Test the Connection



Z MySQL Methods

- Don't forget
 - Use Database Methods
 - Plone permission
 - Valid SQL
 - Limit results
 - No semi-colons
 - Beware of namespace problems
- Generalize w/ dtml
- Link results to a Python class

The screenshot shows the 'Z SQL Method' configuration window. At the top, there are tabs: Edit, Test, Advanced, Security, Undo, and Ownership. The 'Edit' tab is active. Below the tabs, the title is 'Z SQL Method at /people/copy_of_get_people' with a 'Help!' link. The 'Title' field contains 'Retrieves a list of people'. The 'Connection Id' dropdown is set to 'CSEDEV'. The 'Arguments' field contains 'id last active type entity'. The main text area contains a SQL query and a DTML snippet. The SQL query is: `SELECT p.*,CONCAT(l.building, ' ', l.room) as room,CONCAT(l2.building, ' ', l2.room) as room2 FROM people p JOIN affiliations AS a ON p.id=a.person_id JOIN locations AS l ON p.location_id=l.id JOIN buildings AS b ON l.building=b.id JOIN locations AS l2 ON p.location_id2=l2.id JOIN buildings AS b2 ON l2.building=b2.id`. The DTML snippet is: `<dtml-sqlgroup where>
<dtml-with REQUEST only>
<dtml-if id>
 p.id=<dtml-sqlvar id type=int>
</dtml-if>
</dtml-with>
<dtml-and>
<dtml-if last>
 p.last like <dtml-sqlvar last type=string>`. At the bottom, there are buttons: 'Save Changes', 'Change and Test', 'Taller', 'Shorter', 'Wider', and 'Narrower'.

Edit Test Advanced Security Undo Ownership

Z SQL Method at /people/copy_of_get_people Help!

Title Retrieves a list of people

Connection Id CSEDEV

Arguments id last active type entity

```
SELECT p.*,CONCAT(l.building, ' ', l.room) as room,CONCAT(l2.building, ' ', l2.room) as room2 FROM people p JOIN affiliations AS a ON p.id=a.person_id JOIN locations AS l ON p.location_id=l.id JOIN buildings AS b ON l.building=b.id JOIN locations AS l2 ON p.location_id2=l2.id JOIN buildings AS b2 ON l2.building=b2.id
<dtml-sqlgroup where>
  <dtml-with REQUEST only>
    <dtml-if id>
      p.id=<dtml-sqlvar id type=int>
    </dtml-if>
  </dtml-with>
  <dtml-and>
    <dtml-if last>
      p.last like <dtml-sqlvar last type=string>
```

Save Changes Change and Test

Taller Shorter Wider Narrower

Integration with ZPTs

- Call like any Python method



```
<tal:define tal:define="sname python:group[0];  
                lname python:group[1];  
                people python:context.get_people(type=sname,active='yes',entity=1);">  
<h3><a tal:attributes="name surname"></a><span tal:replace="lname"/></h3>  
<table width="100%" border="0" cellpadding="2" cellspacing="0">
```

- Loop through like any list of objects

```
<tr tal:repeat="person people">  
  <td><a tal:condition="person/website" tal:attributes="href person/website" tal:content="person/website"></a>  
  <td tal:condition="python:sname=='staff'" tal:content="person/title">Title</td>
```

Example Select

- Using Arguments
 - dtml-sqlgroup
 - dtml-if, dtml-elif, dtml-else
 - dtml-and, dtml-or
 - dtml-sqltest
- Types
 - int
 - float
 - string
 - nb (non-blank strings)


 Z SQL Method at /people/copy_of_get_people 

Title	Retrieves a list of people
Connection Id	CSEDEV <input type="button" value="v"/>
Arguments	id last active type entity

```
SELECT p.*,  
       CONCAT(l.building, ' ', l.room) as room,  
       CONCAT(l2.building, ' ', l2.room) as room2,  
       a.title as title,  
       b.longitude,  
       b.latitude  
FROM people AS p  
JOIN affiliations AS a ON p.id=a.person_id  
JOIN locations AS l ON p.location_id=l.id  
JOIN buildings AS b ON l.building=b.id  
JOIN locations AS l2 ON p.location_id2=l2.id  
JOIN buildings AS b2 ON l2.building=b2.id  
<dtml-sqlgroup where>  
  <dtml-sqltest id column="p.id" type="int" optional>  
  <dtml-and>  
  <dtml-sqltest last column="p.last" type="nb" optional>  
  <dtml-and>  
  <dtml-sqltest active column="a.active" type="nb" optional>  
  <dtml-and>  
  <dtml-sqltest entity column="a.entity_id" type="int" optional>  
  <dtml-and>  
  <dtml-sqltest type column="a.type" type="nb" optional>  
</dtml-sqlgroup>  
ORDER BY p.last
```


Example Insert

- Same as Select
- Call like any Python method
- Safe from SQL injection attacks
- MySQL may truncate values
- Enforces types
 - Invalid Integer Value

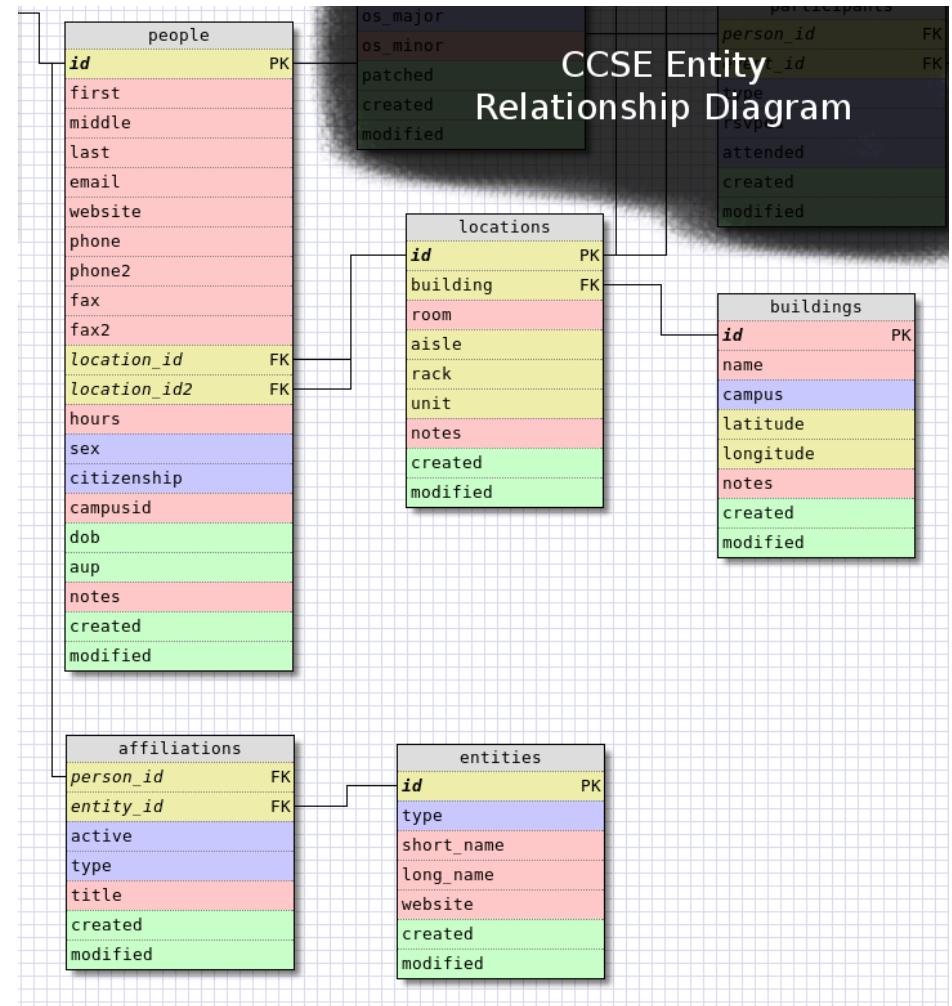
 Z SQL Method at /people/manager/add_person

Title	Add a person record
Connection Id	CSEDEV
Arguments	first middle last email website phone phone2 fax fax2 location_id location_id2 hours sex citizenship <u>campusid</u> <u>dob</u> <u>aup</u> notes

```
INSERT INTO people SET
  created      = now(),
  modified     = now(),
  first        = <dtml-sqlvar first type=string>,
  middle       = <dtml-sqlvar middle type=string>,
  last         = <dtml-sqlvar last type=string>,
  email        = <dtml-sqlvar email type=string>,
  website      = <dtml-sqlvar website type=string>,
  phone        = <dtml-sqlvar phone type=string>,
  phone2       = <dtml-sqlvar phone2 type=string>,
  fax          = <dtml-sqlvar fax type=string>,
  fax2         = <dtml-sqlvar fax2 type=string>,
  location_id  = <dtml-sqlvar location_id type=int>,
  location_id2 = <dtml-sqlvar location_id2 type=int>,
  hours        = <dtml-sqlvar hours type=string>,
  sex          = <dtml-sqlvar sex type=string>,
  citizenship  = <dtml-sqlvar citizenship type=string>,
  campusid    = <dtml-sqlvar campusid type=string>,
  dob         = <dtml-sqlvar dob type=string>,
  aup         = <dtml-sqlvar aup type=string>,
  notes        = <dtml-sqlvar notes type=string>
```

Use Case: A People Manager

- CCSE DB
 - Moderately complex relational database
 - 16+ mostly normalized tables
 - Integrates into mail server, dns, web logs, cluster management, etc.
 - MySQL replication
- Need a way to manage relationships between people and various of campus entities
- Adhoc reporting common
- Mmmm AJAX



Where to go now?

- Query results as Python objects
- Archetypes using SQLStorage
- PloneFormGen (Steve McMahon)
- Convert custom ZPT and scripts to a Plone product

References

- Python MySQL:
<http://sourceforge.net/projects/mysql-python>
- Zope Book:
http://www.plope.com/Books/2_7Edition/
- MySQL Documentation:
<http://dev.mysql.com/doc/refman/5.0/en/>
- Plone and MySQL Tutorial:
<http://plone.org/documentation/tutorial/plone-and-mysql/tutorial-all-pages>
- ZSQL Methods User Guide:
<http://www.zope.org/Documentation/Guides/ZSQL/2.1.1%20pdf/ZSQL.pdf>
- Archetypes using MySQL:
<http://plone.org/documentation/how-to/archetypes-using-mysql>
- PloneFormGen:
<http://plone.org/products/ploneformgen/documentation/tutorial/sql-crud>