Patchwork: reducing your patch workload

Jeremy Kerr
<jk@ozlabs.org>
Patchwork basics

- Patches parsed from mailing list
- Maintainer updates state
  - Depending on patch review
- Patch flow is tracked
Patch flow

new → under review

accepted
rejected
“Todo list” for maintainers
Patch flow
Status updates for contributors
Slightly-hidden features
pwclient
<table>
<thead>
<tr>
<th>ID</th>
<th>State</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>44602</td>
<td>Rejected</td>
<td>Configure WSGI to pass HTTP authorization</td>
</tr>
<tr>
<td>44603</td>
<td>New</td>
<td>De-hyphenate Git commands</td>
</tr>
<tr>
<td>44605</td>
<td>Accepted</td>
<td>Add a simple sample Git post-receive hook</td>
</tr>
<tr>
<td>44606</td>
<td>Accepted</td>
<td>Decode patch from UTF-8 while parsing</td>
</tr>
</tbody>
</table>
$ pwclient list -n 4

<table>
<thead>
<tr>
<th>ID</th>
<th>State</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>44602</td>
<td>Rejected</td>
<td>Configure WSGI to pass HTTP authorization</td>
</tr>
<tr>
<td>44603</td>
<td>New</td>
<td>De-hyphenate Git commands</td>
</tr>
<tr>
<td>44605</td>
<td>Accepted</td>
<td>Add a simple sample Git post-receive hook</td>
</tr>
<tr>
<td>44606</td>
<td>Accepted</td>
<td>Decode patch from UTF-8 while parsing</td>
</tr>
</tbody>
</table>

$ pwclient get 44603
Saved patch to De-hyphenate-Git-commands.patch
$ pwclient list -n 4
ID    State      Name
--    -----      ----
44602 Rejected   Configure WSGI to pass HTTP authorization
44603 New        De-hyphenate Git commands
44605 Accepted   Add a simple sample Git post-receive hook
44606 Accepted   Decode patch from UTF-8 while parsing

$ pwclient get 44603
Saved patch to De-hyphenate-Git-commands.patch

$ git am De-hyphenate-Git-comands.patch
$ pwclient list -n 4
<table>
<thead>
<tr>
<th>ID</th>
<th>State</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>44602</td>
<td>Rejected</td>
<td>Configure WSGI to pass HTTP authorization</td>
</tr>
<tr>
<td>44603</td>
<td>New</td>
<td>De-hyphenate Git commands</td>
</tr>
<tr>
<td>44605</td>
<td>Accepted</td>
<td>Add a simple sample Git post-receive hook</td>
</tr>
<tr>
<td>44606</td>
<td>Accepted</td>
<td>Decode patch from UTF-8 while parsing</td>
</tr>
</tbody>
</table>

$ pwclient get 44603
Saved patch to De-hyphenate-Git-commands.patch

$ git am De-hyphenate-Git-commands.patch

$ pwclient update -s Accepted 44603
Patch hashes

• `pwparsrer --hash < some.patch`
  • Generates a hash of `some.patch`
• `pwclient update -h <hash> ...`
  • Updates patchwork patch with hash `<hash>`
# Patchwork “catchup” script

git rev-list rev1..rev2 | 
while read commit 
do 
  h=$(git show $commit | pwparserr --hash)
  pwclient update -s Accepted -h $h 
done
X-Patchwork-Hint: ignore
New stuff
Hello,

The following patch (submitted by you) has been updated in patchwork:

* models: Fix HeaderParser import on python 2.4
  - [http://patchwork.ozlabs.org/patch/69913/](http://patchwork.ozlabs.org/patch/69913/)
  was: Under Review
  now: Accepted

This email is a notification only - you do not need to respond.

Happy patchworking.

--

This is an automated mail sent by the patchwork system at patchwork.ozlabs.org. To stop receiving these notifications, edit your mail settings at:

[http://patchwork.ozlabs.org/mail/](http://patchwork.ozlabs.org/mail/)
xmlrpc: do slice before patch_to_dict

Currently, we map patch_to_dict before we slice the results (to only return max_count patches). This means that we have to retrieve all patches, then throw away most of the results of the map.

This change does the slice on the patches before the map, letting django do a LIMIT-ed query instead.

Signed-off-by: Jeremy Kerr <jk@ozlabs.org>

--- a/apps/patchwork/views/xmlrpc.py
+++ b/apps/patchwork/views/xmlrpc.py
@@ -328,7 +328,7 @@ def patch_list(filter={}):
     patches = Patch.objects.filter(**dfilter)

         if max_count > 0:
-            return map(patch_to_dict, patches)[:max_count]
+            return map(patch_to_dict, patches[:max_count])
     else:
         return map(patch_to_dict, patches)
From: Patch Author <author@example.com>
X-Patchwork-State: RFC
X-Patchwork-State: rejected
Feature requests
Invalid encoding handling
Current encoding handling

- Parser reads encoding from mail headers
- Aborts on invalid byte sequences
A charset unification patch

• Removes a line in iso-8895-1
• Adds a line in utf-8
Patches as binary data

- Invalid encodings are flagged, not dropped
Patches as binary data

• How do we display patches?
• How do we indicate encoding errors?
Notification suppression
Multiple patch targets
What would help reduce your backlog?
Development
git://git.ozlabs.org/home/jk/git/patchwork
$ createdb patchwork
$ ./manage.py syncdb
$ ./manage.py runserver
$ ./manage.py test patchwork
Thank you.