Redmine - Feature #33784
Updating Mercurial helper to work with Python3

2020-07-29 16:26 - Harald Klimach

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	SCM	**Estimated time:**	0.00 hour
Target version:	Candidate for next major release		
Resolution:			

Description

As Mercurial is transitioning to Python 3, I updated my Mercurial version now to make use of Python 3 and ran into trouble with the RedmineHelper. Mainly due the string handling, as the Mercurial API requires bytes objects instead of strings.

Here is a patch, that changes the helper to work with Mercurial on Python 3, though it is not compatible with Python2 then anymore. There probably are better ways to implement this, but it works for me in this form.

```
Index: lib/redmine/scm/adapters/mercurial/redminehelper.py
===================================================================
--- lib/redmine/scm/adapters/mercurial/redminehelper.py    (revision 19937)
+++ lib/redmine/scm/adapters/mercurial/redminehelper.py    (working copy)
@@ -45,14 +45,14 @@
       </repository>
     </rhmanifest>
 """
-import re, time, cgi, urllib
+import re, time, html, urllib
 from mercurial import cmdutil, commands, node, error, hg, registrar

 cmdtable = {}
 command = registrar.command(cmdtable) if hasattr(registrar, 'command') else cmdutil.command(cmdtable)

-_x = cgi.escape
-_u = lambda s: cgi.escape(urllib.quote(s))
+_x = lambda s: html.escape(s.decode('utf-8')).encode('utf-8')
+_u = lambda s: html.escape(urllib.parse.quote(s)).encode('utf-8')

 def _changectx(repo, rev):
     if isinstance(rev, str):
@@ -70,10 +70,10 @@
         except TypeError:  # Mercurial < 1.1
             return repo.changelog.count() - 1
     tipctx = _changectx(repo, tiprev())
-    ui.write('<tip revision="%d" node="%s"/>\n'
+    ui.write(b'<tip revision="%d" node="%s"/>\n'
              % (tipctx.rev(), _x(node.hex(tipctx.node()))))

-_SPECIAL_TAGS = ('tip',)
+_SPECIAL_TAGS = (b'tip',)

 def _tags(ui, repo):
     # see mercurial/commands.py:tags
```

```
@@ -84,7 +84,7 @@
             r = repo.changelog.rev(n)
         except error.LookupError:
             continue
-        ui.write('<tag revision="%d" node="%s" name="%s"/>\n'
+        ui.write(b'<tag revision="%d" node="%s" name="%s"/>\n'
                  % (r, _x(node.hex(n)), _x(t)))

 def _branches(ui, repo):
@@ -104,66 +104,67 @@
             return repo.branchheads(branch)
     def lookup(rev, n):
         try:
-            return repo.lookup(rev)
+            return repo.lookup(str(rev).encode('utf-8'))
         except RuntimeError:
             return n
     for t, n, r in sorted(iterbranches(), key=lambda e: e[2], reverse=True):
         if lookup(r, n) in branchheads(t):
-            ui.write('<branch revision="%d" node="%s" name="%s"/>\n'
+            ui.write(b'<branch revision="%d" node="%s" name="%s"/>\n'
                      % (r, _x(node.hex(n)), _x(t)))

 def _manifest(ui, repo, path, rev):
     ctx = _changectx(repo, rev)
-    ui.write('<manifest revision="%d" path="%s">\n'
+    ui.write(b'<manifest revision="%d" path="%s">\n'
              % (ctx.rev(), _u(path)))

     known = set()
-    pathprefix = (path.rstrip('/') + '/').lstrip('/')
+    pathprefix = (path.decode('utf-8').rstrip('/') + '/').lstrip('/')
     for f, n in sorted(ctx.manifest().iteritems(), key=lambda e: e[0]):
-        if not f.startswith(pathprefix):
+        fstr = f.decode('utf-8')
+        if not fstr.startswith(pathprefix):
             continue
-        name = re.sub(r'/.*', '/', f[len(pathprefix):])
+        name = re.sub(r'/.*', '/', fstr[len(pathprefix):])
         if name in known:
             continue
         known.add(name)

         if name.endswith('/'):
-            ui.write('<dir name="%s"/>\n'
-                     % _x(urllib.quote(name[:-1])))
+            ui.write(b'<dir name="%s"/>\n'
+                     % _x(urllib.parse.quote(name[:-1]).encode('utf-8')))
         else:
             fctx = repo.filectx(f, fileid=n)
             tm, tzoffset = fctx.date()
-            ui.write('<file name="%s" revision="%d" node="%s" '
-                     'time="%d" size="%d"/>\n'
```

```
+            ui.write(b'<file name="%s" revision="%d" node="%s" '
+                     b'time="%d" size="%d"/>\n'
                     % (_u(name), fctx.rev(), _x(node.hex(fctx.node())),
                        tm, fctx.size(), ))

-    ui.write('</manifest>\n')
+    ui.write(b'</manifest>\n')

-@command('rhannotate',
-         [('r', 'rev', '', 'revision'),
-          ('u', 'user', None, 'list the author (long with -v)'),
-          ('n', 'number', None, 'list the revision number (default)'),
-          ('c', 'changeset', None, 'list the changeset'),
+@command(b'rhannotate',
+         [(b'r', b'rev', b'', b'revision'),
+          (b'u', b'user', None, b'list the author (long with -v)'),
+          (b'n', b'number', None, b'list the revision number (default)'),
+          (b'c', b'changeset', None, b'list the changeset'),
          ],
-         'hg rhannotate [-r REV] [-u] [-n] [-c] FILE...')
+         b'hg rhannotate [-r REV] [-u] [-n] [-c] FILE...')
 def rhannotate(ui, repo, *pats, **opts):
-    rev = urllib.unquote_plus(opts.pop('rev', None))
+    rev = urllib.parse.unquote_plus(opts.pop('rev', None))
     opts['rev'] = rev
-    return commands.annotate(ui, repo, *map(urllib.unquote_plus, pats), **opts)
+    return commands.annotate(ui, repo, *map(urllib.parse.unquote_plus, pats), **opts)

-@command('rhcat',
-         [('r', 'rev', '', 'revision')],
-         'hg rhcat ([-r REV] ...) FILE...')
+@command(b'rhcat',
+         [(b'r', b'rev', b'', b'revision')],
+         b'hg rhcat ([-r REV] ...) FILE...')
 def rhcat(ui, repo, file1, *pats, **opts):
-    rev = urllib.unquote_plus(opts.pop('rev', None))
+    rev = urllib.parse.unquote_plus(opts.pop('rev', None))
     opts['rev'] = rev
-    return commands.cat(ui, repo, urllib.unquote_plus(file1), *map(urllib.unquote_plus, pats), **opts)
+    return commands.cat(ui, repo, urllib.parse.unquote_plus(file1), *map(urllib.parse.unquote_plus, pats), **opts)

-@command('rhdiff',
-         [('r', 'rev', [], 'revision'),
-          ('c', 'change', '', 'change made by revision')],
-         'hg rhdiff ([-c REV] | [-r REV] ...) [FILE]...')
+@command(b'rhdiff',
+         [(b'r', b'rev', [], b'revision'),
+          (b'c', b'change', b'', b'change made by revision')],
+         b'hg rhdiff ([-c REV] | [-r REV] ...) [FILE]...')
 def rhdiff(ui, repo, *pats, **opts):
     """diff repository (or selected files)"""
     change = opts.pop('change', None)
@@ -171,34 +172,34 @@
```

```
        base = _changectx(repo, change).parents()[0].rev()
        opts['rev'] = [str(base), change]
    opts['nodates'] = True
-   return commands.diff(ui, repo, *map(urllib.unquote_plus, pats), **opts)
+    return commands.diff(ui, repo, *map(urllib.parse.unquote_plus, pats), **opts)

-@command('rhlog',
+@command(b'rhlog',
                 [
-                 ('r', 'rev', [], 'show the specified revision'),
-                 ('b', 'branch', [],
-                    'show changesets within the given named branch'),
-                 ('l', 'limit', '',
-                      'limit number of changes displayed'),
-                 ('d', 'date', '',
-                      'show revisions matching date spec'),
-                 ('u', 'user', [],
-                   'revisions committed by user'),
-                 ('', 'from', '',
-                   ''),
-                 ('', 'to', '',
-                   ''),
-                 ('', 'rhbranch', '',
-                   ''),
-                 ('', 'template', '',
-                    'display with template')],
-                'hg rhlog [OPTION]... [FILE]')
+                 (b'r', b'rev', [], b'show the specified revision'),
+                 (b'b', b'branch', [],
+                    b'show changesets within the given named branch'),
+                 (b'l', b'limit', b'',
+                      b'limit number of changes displayed'),
+                 (b'd', b'date', b'',
+                      b'show revisions matching date spec'),
+                 (b'u', b'user', [],
+                   b'revisions committed by user'),
+                 (b'', b'from', b'',
+                   b''),
+                 (b'', b'to', b'',
+                   b''),
+                 (b'', b'rhbranch', b'',
+                   b''),
+                 (b'', b'template', b'',
+                    b'display with template')],
+                b'hg rhlog [OPTION]... [FILE]')
 def rhlog(ui, repo, *pats, **opts):
     rev      = opts.pop('rev')
     bra0     = opts.pop('branch')
-    from_rev = urllib.unquote_plus(opts.pop('from', None))
-    to_rev   = urllib.unquote_plus(opts.pop('to'  , None))
-    bra      = urllib.unquote_plus(opts.pop('rhbranch', None))
+    from_rev = urllib.parse.unquote_plus(opts.pop('from', None))
+    to_rev   = urllib.parse.unquote_plus(opts.pop('to'  , None))
```

```
+    bra      = urllib.parse.unquote_plus(opts.pop('rhbranch', None))
     from_rev = from_rev.replace("", '\\"')
     to_rev   = to_rev.replace("", '\\"')
     if hg.util.version() >= '1.6':
@@ -206,28 +207,30 @@
     else:
       opts['rev'] = ['%s:%s' % (from_rev, to_rev)]
     opts['branch'] = [bra]
-    return commands.log(ui, repo, *map(urllib.unquote_plus, pats), **opts)
+    return commands.log(ui, repo, *map(urllib.parse.unquote_plus, pats), **opts)

-@command('rhmanifest',
-                 [('r', 'rev', '', 'show the specified revision')],
-                 'hg rhmanifest [-r REV] [PATH]')
-def rhmanifest(ui, repo, path='', **opts):
+@command(b'rhmanifest',
+                 [(b'r', b'rev', b'', b'show the specified revision')],
+                 b'hg rhmanifest [-r REV] [PATH]')
+def rhmanifest(ui, repo, path=b'', **opts):
     """output the sub-manifest of the specified directory"""
-    ui.write('<?xml version="1.0"?>\n')
-    ui.write('<rhmanifest>\n')
-    ui.write('<repository root="%s">\n' % _u(repo.root))
+    ui.write(b'<?xml version="1.0"?>\n')
+    ui.write(b'<rhmanifest>\n')
+    ui.write(b'<repository root="%s">\n' % _u(repo.root))
     try:
-        _manifest(ui, repo, urllib.unquote_plus(path), urllib.unquote_plus(opts.get('rev')))
+        _manifest(ui, repo,
+                  urllib.parse.unquote_plus(path.decode('utf-8')).encode('utf-8'),
+                  urllib.parse.unquote_plus(opts.get('rev').decode('utf-8')).encode('utf-8'))
     finally:
-        ui.write('</repository>\n')
-        ui.write('</rhmanifest>\n')
+        ui.write(b'</repository>\n')
+        ui.write(b'</rhmanifest>\n')

-@command('rhsummary',[], 'hg rhsummary')
+@command(b'rhsummary', [], b'hg rhsummary')
 def rhsummary(ui, repo, **opts):
     """output the summary of the repository"""
-    ui.write('<?xml version="1.0"?>\n')
-    ui.write('<rhsummary>\n')
-    ui.write('<repository root="%s">\n' % _u(repo.root))
+    ui.write(b'<?xml version="1.0"?>\n')
+    ui.write(b'<rhsummary>\n')
+    ui.write(b'<repository root="%s">\n' % _u(repo.root))
     try:
         _tip(ui, repo)
         _tags(ui, repo)
@@ -234,6 +237,6 @@
         _branches(ui, repo)
         # TODO: bookmarks in core (Mercurial>=1.8)
```

```
        finally:
-           ui.write('</repository>\n')
-           ui.write('</rhsummary>\n')
+           ui.write(b'</repository>\n')
+           ui.write(b'</rhsummary>\n')
```

It would be great, if Redmine could support Mercurial on Python 3. Maybe the mercurial.pycompat module could be of help to create a patch that does not break backwards compatibility.

History

#1 - 2020-07-31 16:52 - Harald Klimach

The patch above does not work properly for everything (file views and diffs of files), so I had to revise it a little bit. Still this changed version only supports Mercurial on Python 3:

```
Index: lib/redmine/scm/adapters/mercurial/redminehelper.py
===================================================================
--- lib/redmine/scm/adapters/mercurial/redminehelper.py    (revision 19937)
+++ lib/redmine/scm/adapters/mercurial/redminehelper.py    (working copy)
@@ -45,17 +45,20 @@
       </repository>
     </rhmanifest>
 """
-import re, time, cgi, urllib
+import re, time, html, urllib
 from mercurial import cmdutil, commands, node, error, hg, registrar

 cmdtable = {}
 command = registrar.command(cmdtable) if hasattr(registrar, 'command') else cmdutil.command(cmdtable)

-_x = cgi.escape
-_u = lambda s: cgi.escape(urllib.quote(s))
+_x = lambda s: html.escape(s.decode('utf-8')).encode('utf-8')
+_u = lambda s: html.escape(urllib.parse.quote(s)).encode('utf-8')

+def unquoteplus(*args, **kwargs):
+    return urllib.parse.unquote_to_bytes(*args, **kwargs).replace(b'+', b' ')
+
 def _changectx(repo, rev):
-    if isinstance(rev, str):
+    if isinstance(rev, bytes):
         rev = repo.lookup(rev)
     if hasattr(repo, 'changectx'):
         return repo.changectx(rev)
@@ -70,10 +73,10 @@
         except TypeError:  # Mercurial < 1.1
             return repo.changelog.count() - 1
     tipctx = _changectx(repo, tiprev())
-    ui.write('<tip revision="%d" node="%s"/>\n'
+    ui.write(b'<tip revision="%d" node="%s"/>\n'
```

```
            % (tipctx.rev(), _x(node.hex(tipctx.node()))))

-_SPECIAL_TAGS = ('tip',)
+_SPECIAL_TAGS = (b'tip',)

 def _tags(ui, repo):
     # see mercurial/commands.py:tags
@@ -84,7 +87,7 @@
             r = repo.changelog.rev(n)
         except error.LookupError:
             continue
-        ui.write('<tag revision="%d" node="%s" name="%s"/>\n'
+        ui.write(b'<tag revision="%d" node="%s" name="%s"/>\n'
                  % (r, _x(node.hex(n)), _x(t)))

 def _branches(ui, repo):
@@ -104,130 +107,140 @@
             return repo.branchheads(branch)
     def lookup(rev, n):
         try:
-            return repo.lookup(rev)
+            return repo.lookup(str(rev).encode('utf-8'))
         except RuntimeError:
             return n
     for t, n, r in sorted(iterbranches(), key=lambda e: e[2], reverse=True):
         if lookup(r, n) in branchheads(t):
-            ui.write('<branch revision="%d" node="%s" name="%s"/>\n'
+            ui.write(b'<branch revision="%d" node="%s" name="%s"/>\n'
                      % (r, _x(node.hex(n)), _x(t)))

 def _manifest(ui, repo, path, rev):
     ctx = _changectx(repo, rev)
-    ui.write('<manifest revision="%d" path="%s">\n'
+    ui.write(b'<manifest revision="%d" path="%s">\n'
             % (ctx.rev(), _u(path)))

     known = set()
-    pathprefix = (path.rstrip('/') + '/').lstrip('/')
+    pathprefix = (path.decode('utf-8').rstrip('/') + '/').lstrip('/')
     for f, n in sorted(ctx.manifest().iteritems(), key=lambda e: e[0]):
-        if not f.startswith(pathprefix):
+        fstr = f.decode('utf-8')
+        if not fstr.startswith(pathprefix):
             continue
-        name = re.sub(r'/.*', '/', f[len(pathprefix):])
+        name = re.sub(r'/.*', '/', fstr[len(pathprefix):])
         if name in known:
             continue
         known.add(name)

         if name.endswith('/'):
-            ui.write('<dir name="%s"/>\n'
-                     % _x(urllib.quote(name[:-1])))
```

```
+            ui.write(b'<dir name="%s"/>\n'
+                     % _x(urllib.parse.quote(name[:-1]).encode('utf-8')))
         else:
             fctx = repo.filectx(f, fileid=n)
             tm, tzoffset = fctx.date()
-            ui.write('<file name="%s" revision="%d" node="%s" '
-                     'time="%d" size="%d"/>\n'
+            ui.write(b'<file name="%s" revision="%d" node="%s" '
+                     b'time="%d" size="%d"/>\n'
                     % (_u(name), fctx.rev(), _x(node.hex(fctx.node())),
                       tm, fctx.size(), ))

-    ui.write('</manifest>\n')
+    ui.write(b'</manifest>\n')

-@command('rhannotate',
-         [('r', 'rev', '', 'revision'),
-          ('u', 'user', None, 'list the author (long with -v)'),
-          ('n', 'number', None, 'list the revision number (default)'),
-          ('c', 'changeset', None, 'list the changeset'),
+@command(b'rhannotate',
+         [(b'r', b'rev', b'', b'revision'),
+          (b'u', b'user', None, b'list the author (long with -v)'),
+          (b'n', b'number', None, b'list the revision number (default)'),
+          (b'c', b'changeset', None, b'list the changeset'),
          ],
-         'hg rhannotate [-r REV] [-u] [-n] [-c] FILE...')
+         b'hg rhannotate [-r REV] [-u] [-n] [-c] FILE...')
 def rhannotate(ui, repo, *pats, **opts):
-    rev = urllib.unquote_plus(opts.pop('rev', None))
+    rev = unquoteplus(opts.pop('rev', b''))
     opts['rev'] = rev
-    return commands.annotate(ui, repo, *map(urllib.unquote_plus, pats), **opts)
+    return commands.annotate(ui, repo, *map(unquoteplus, pats), **opts)

-@command('rhcat',
-             [('r', 'rev', '', 'revision')],
-             'hg rhcat ([-r REV] ...) FILE...')
+@command(b'rhcat',
+             [(b'r', b'rev', b'', b'revision')],
+             b'hg rhcat ([-r REV] ...) FILE...')
 def rhcat(ui, repo, file1, *pats, **opts):
-    rev = urllib.unquote_plus(opts.pop('rev', None))
+    rev = unquoteplus(opts.pop('rev', b''))
     opts['rev'] = rev
-    return commands.cat(ui, repo, urllib.unquote_plus(file1), *map(urllib.unquote_plus, pats), **opts)
+    return commands.cat(ui, repo, unquoteplus(file1), *map(unquoteplus, pats), **opts)

-@command('rhdiff',
-             [('r', 'rev', [], 'revision'),
-              ('c', 'change', '', 'change made by revision')],
-             'hg rhdiff ([-c REV] | [-r REV] ...) [FILE]...')
+@command(b'rhdiff',
```

```
+              [(b'r', b'rev', [], b'revision'),
+               (b'c', b'change', b'', b'change made by revision')],
+              b'hg rhdiff ([-c REV] | [-r REV] ...) [FILE]...')
 def rhdiff(ui, repo, *pats, **opts):
     """diff repository (or selected files)"""
     change = opts.pop('change', None)
     if change:  # add -c option for Mercurial<1.1
         base = _changectx(repo, change).parents()[0].rev()
-        opts['rev'] = [str(base), change]
+        opts['rev'] = [base, change]
     opts['nodates'] = True
-    return commands.diff(ui, repo, *map(urllib.unquote_plus, pats), **opts)
+    return commands.diff(ui, repo, *map(unquoteplus, pats), **opts)

-@command('rhlog',
+@command(b'rhlog',
                  [
-                  ('r', 'rev', [], 'show the specified revision'),
-                  ('b', 'branch', [],
-                     'show changesets within the given named branch'),
-                  ('l', 'limit', '',
-                       'limit number of changes displayed'),
-                  ('d', 'date', '',
-                       'show revisions matching date spec'),
-                  ('u', 'user', [],
-                    'revisions committed by user'),
-                  ('', 'from', '',
-                    ''),
-                  ('', 'to', '',
-                    ''),
-                  ('', 'rhbranch', '',
-                    ''),
-                  ('', 'template', '',
-                     'display with template')],
-                 'hg rhlog [OPTION]... [FILE]')
+                  (b'r', b'rev', [], b'show the specified revision'),
+                  (b'b', b'branch', [],
+                     b'show changesets within the given named branch'),
+                  (b'l', b'limit', b'',
+                       b'limit number of changes displayed'),
+                  (b'd', b'date', b'',
+                       b'show revisions matching date spec'),
+                  (b'u', b'user', [],
+                    b'revisions committed by user'),
+                  (b'', b'from', b'',
+                    b''),
+                  (b'', b'to', b'',
+                    b''),
+                  (b'', b'rhbranch', b'',
+                    b''),
+                  (b'', b'template', b'',
+                     b'display with template')],
+                 b'hg rhlog [OPTION]... [FILE]')
```

```
 def rhlog(ui, repo, *pats, **opts):
     rev      = opts.pop('rev')
     bra0     = opts.pop('branch')
-    from_rev = urllib.unquote_plus(opts.pop('from', None))
-    to_rev   = urllib.unquote_plus(opts.pop('to'  , None))
-    bra      = urllib.unquote_plus(opts.pop('rhbranch', None))
-    from_rev = from_rev.replace('"', '\\"')
-    to_rev   = to_rev.replace('"', '\\"')
-    if hg.util.version() >= '1.6':
-      opts['rev'] = ['"%s":"%s"' % (from_rev, to_rev)]
+    from_rev = unquoteplus(opts.pop('from', b''))
+    to_rev   = unquoteplus(opts.pop('to'  , b''))
+    bra      = unquoteplus(opts.pop('rhbranch', b''))
+    from_rev = from_rev.replace(b'"', b'\\"')
+    to_rev   = to_rev.replace(b'"', b'\\"')
+    if (from_rev != b'') or (to_rev != b''):
+        if from_rev != b'':
+            quotefrom = b'"%s"' % (from_rev)
+        else:
+            quotefrom = from_rev
+        if to_rev != b'':
+            quoteto = b'"%s"' % (to_rev)
+        else:
+            quoteto = to_rev
+        opts['rev'] = [b'%s:%s' % (quotefrom, quoteto)]
     else:
-      opts['rev'] = ['%s:%s' % (from_rev, to_rev)]
-    opts['branch'] = [bra]
-    return commands.log(ui, repo, *map(urllib.unquote_plus, pats), **opts)
+        opts['rev'] = rev
+    if (bra != b''):
+        opts['branch'] = [bra]
+    return commands.log(ui, repo, *map(unquoteplus, pats), **opts)

-@command('rhmanifest',
-                 [('r', 'rev', '', 'show the specified revision')],
-                 'hg rhmanifest [-r REV] [PATH]')
-def rhmanifest(ui, repo, path='', **opts):
+@command(b'rhmanifest',
+                  [(b'r', b'rev', b'', b'show the specified revision')],
+                  b'hg rhmanifest -r REV [PATH]')
+def rhmanifest(ui, repo, path=b'', **opts):
     """output the sub-manifest of the specified directory"""
-    ui.write('<?xml version="1.0"?>\n')
-    ui.write('<rhmanifest>\n')
-    ui.write('<repository root="%s">\n' % _u(repo.root))
+    ui.write(b'<?xml version="1.0"?>\n')
+    ui.write(b'<rhmanifest>\n')
+    ui.write(b'<repository root="%s">\n' % _u(repo.root))
     try:
-        _manifest(ui, repo, urllib.unquote_plus(path), urllib.unquote_plus(opts.get('rev')))
+        _manifest(ui, repo, unquoteplus(path), unquoteplus(opts.get('rev')))
     finally:
```

```
-       ui.write('</repository>\n')
-       ui.write('</rhmanifest>\n')
+       ui.write(b'</repository>\n')
+       ui.write(b'</rhmanifest>\n')

-@command('rhsummary',[], 'hg rhsummary')
+@command(b'rhsummary', [], b'hg rhsummary')
 def rhsummary(ui, repo, **opts):
     """output the summary of the repository"""
-    ui.write('<?xml version="1.0"?>\n')
-    ui.write('<rhsummary>\n')
-    ui.write('<repository root="%s">\n' % _u(repo.root))
+    ui.write(b'<?xml version="1.0"?>\n')
+    ui.write(b'<rhsummary>\n')
+    ui.write(b'<repository root="%s">\n' % _u(repo.root))
     try:
         _tip(ui, repo)
         _tags(ui, repo)
@@ -234,6 +247,6 @@
         _branches(ui, repo)
         # TODO: bookmarks in core (Mercurial>=1.8)
     finally:
-       ui.write('</repository>\n')
-       ui.write('</rhsummary>\n')
+       ui.write(b'</repository>\n')
+       ui.write(b'</rhsummary>\n')
```

#2 - 2020-08-08 23:42 - S DW

Mercurial integration with redmine on the NixOS linux distribution is broken, and I think it could be related to this issue?
The error we are getting is either:

```
Aug 08 18:07:33 nixos bundle[871]: *** failed to import extension redminehelper from
/nix/store/c2za42x86iqk8wrxphl4pc2rkb5kbwxq-redmine-4.1.1/share/redmine/lib/redmine/scm/adapters/mercurial/redminehelper.py: unicode
'rhannotate' found in cmdtable
Aug 08 18:07:33 nixos bundle[871]: *** (use b'' to make it byte string)
Aug 08 18:07:33 nixos bundle[871]: hg: unknown command 'rhsummary'
Aug 08 18:07:33 nixos bundle[871]: (did you mean summary?)
```

Or, depending on redmine/mercurial versions used:

```
Aug 08 18:30:38 nixos bundle[1133]: *** failed to import extension redminehelper from
/nix/store/3sdqxliq5qd15wfzyiclqs5ax1b6ggnc-redmine-4.1.1/share/redmine/lib/redmine/scm/adapters/mercurial/redminehelper.py: module
'cgi' has no attribute 'escape'
Aug 08 18:30:38 nixos bundle[1133]: hg: unknown command 'rhsummary'
Aug 08 18:30:38 nixos bundle[1133]: (did you mean summary?)
```

Link to the NixOS issue: https://github.com/NixOS/nixpkgs/issues/94949

#3 - 2022-02-07 21:48 - François Cerbelle

Mercurial switched to python 3, python 2.x is EOL/abandonned.
Debian switched to Rail 6.x in all versions. Thus, it is impossible to use Redmine later than 4.0.7.
It became a real challenge to keep a Redmine system uptodate with dependency versions and security.

Any chance to have this bug/script fixed in the 4.0.x branch ? at least included in a roadmaped version ?
The provided patch merged ? It is better to have something partially working than no mercurial at all.

Not complaining, just asking

#4 - 2022-02-08 08:36 - Harald Klimach

François Cerbelle wrote:

> *Any chance to have this bug/script fixed in the 4.0.x branch ? at least included in a roadmaped version ?*
> *The provided patch merged ? It is better to have something partially working than no mercurial at all.*

Can't say anything about roadmaps or integration into the project, but did you try to apply the patch above? Though I didn't test it, I think it should work for that branch aswell.

#5 - 2022-02-17 10:07 - salman mp

+1

same problem

#6 - 2022-03-22 07:40 - Go MAEDA

- Target version set to Candidate for next major release