XUL::App - Jifty way of doing XUL

XUL::App - *Jifty* way of doing XUL

XUL::App - *Jifty* 风格的 XUL 开发

☺*Agent Zhang*☺

章亦春

2007.11

☼ Let's start with
a *hello world* Firefox extension!

让我们从一个 *hello world*
Firefox 扩展开始！

```
$ xulapp app --name HelloWorld
 Creating new application HelloWorld
 Creating directory lib/
 Creating directory lib/HelloWorld/
 Writing file lib/HelloWorld/App.pm
 Creating directory js/
 Creating directory css/
 Creating directory icons/
 Creating directory icons/default/
 Creating directory t/
 Creating directory po/
 Creating directory docs/
$
```

```
$ cd HelloWorld/
$ ls
 css  docs  icons  js  lib  po  t
```

☺ Let's create a *window view* named HelloWin!

让我们来创建一个名叫 HelloWin 的*窗口视图*!

```
$ xulapp view --name HelloWin --type window
 Creating directory lib/HelloWorld/View/
 Writing file lib/HelloWorld/View/HelloWin.pm
$
```

```
$ wc -l lib/HelloWorld/View/HelloWin.pm
 26 lib/HelloWorld/View/HelloWin.pm
$
```

☺ Now let's *register* our HelloWin view
in our HelloWorld::App module.

现在让我们在 HelloWorld::App 模块中
*注册*一下我们的 HelloWin 视图。

```
$ vim lib/HelloWorld/App.pm
```

```
# File lib/HelloWorld/App.pm
package HelloWorld::App;
our $$VERSION = '0.01';
use XUL::App::Schema;
use XUL::App schema {
    xpifile 'helloworld.xpi' =>
        name is 'HelloWorld',
        id is 'helloworld@agentz.agentz-office', # FIXME
        version is '0.0.1',
        targets {
            Firefox => ['2.0' => '3.0a5'], # FIXME
        },
        creator is 'The HelloWorld development team',
        developers are ['agentz'],
        contributors are [];
        homepageURL is 'http://helloworld.agentz.org',
        ...
};
1;
```

```
# File lib/HelloWorld/App.pm
package HelloWorld::App;
our $$VERSION = '0.01';
use XUL::App::Schema;
use XUL::App schema {
    # Code that we added by hand:
    xulfile 'hellowin.xul' =>
        generated from 'HelloWorld::View::HelloWin';

    xpifile 'helloworld.xpi' =>
        name is 'HelloWorld',
        id is 'helloworld@agentz.agentz-office', # FIXME
        version is '0.0.1',
        targets {
            Firefox => ['2.0' => '3.0a5'], # FIXME
        },
        creator is 'The HelloWorld development team',
        ...
```

☺ Now let's add some *contents* to our HelloWin view class.

现在让我们往 HelloWin 视图类中添加一些*内容*。

```
$ vim lib/HelloWorld/View/HelloWin.pm
```

```
# File lib/HelloWorld/View/HelloWin.pm
package HelloWorld::View::HelloWin;
use base 'XUL::App::View::Base';
use Template::Declare::Tags 'XUL';
 template main => sub {
    show 'header';  # from XUL::App::View::Base
    window {
        attr {
            id => "helloworld-hellowin",
            xmlns => $::XUL_NAME_SPACE,
            title => 'HelloWorld',
            ...
        }
    }
 };
 1;
```

```
# File lib/HelloWorld/View/HelloWin.pm
package HelloWorld::View::HelloWin;
use base 'XUL::App::View::Base';
use Template::Declare::Tags 'XUL';
 template main => sub {
    show 'header';  # from XUL::App::View::Base
    window {
        attr {
            id => "helloworld-hellowin",
            xmlns => $::XUL_NAME_SPACE,
            title => 'HelloWorld',
            ...
        }
        # Code that we added by hand:
        label { "Hello, world!" }
    }
    ...
```

☆ Hey, it *runs* now!

嘿，它现在已经能跑起来了！

```
# Assuming the Firefox profile dev already exists:
$ xulapp debug hellowin.xul --profile dev
 Writing file hellowin.xul
 Registering extension hellowin in profile dev
 Setting configure variables in dev's prefs.js
 MOZ_NO_REMOTE=1 && firefox -jsconsole -P dev -chrome \
     chrome://helloworld/content/hellowin.xul
```


HelloWorld
Hello, world!

☺ If the dev profile does *not* exist,
xulapp will automatically create one for you.

如果 dev 配置文件*不存在*的话，那么 xulapp 会
自动为你创建一个。

```
# Assuming the Firefox profile dev does NOT exist:
$ xulapp debug hellowin.xul --profile dev
 Writing file hellowin.xul
 Creating profile dev
 Success: created profile 'dev'
 Start Firefox temporarily to initialize the profile
 Registering extension hellowin in profile dev
 Setting configure variables in dev's prefs.js
 MOZ_NO_REMOTE=1 && firefox -jsconsole -P dev -chrome \
     chrome://helloworld/content/hellowin.xul
```


HelloWorld
Hello, world!

♡ It's time to
bundle and *distribute* our toy!

到了打包和发布我们的玩具的时候了！

```
$ xulapp bundle .
 Writing file hellowin.xul
 Writing bundle file ./helloworld.xpi
$
```

Our helloworld.xpi bundle ➥

Our helloworld.xpi bundle ➥

- ✓ **contains *0* Perl**

Our helloworld.xpi bundle ➥

- ✓ **contains *0* Perl**
- ✓ **has *0* dependencies**

(except Firefox itself)

Our helloworld.xpi bundle ➥

- ✓ **contains *0* Perl**
- ✓ **has *0* dependencies**

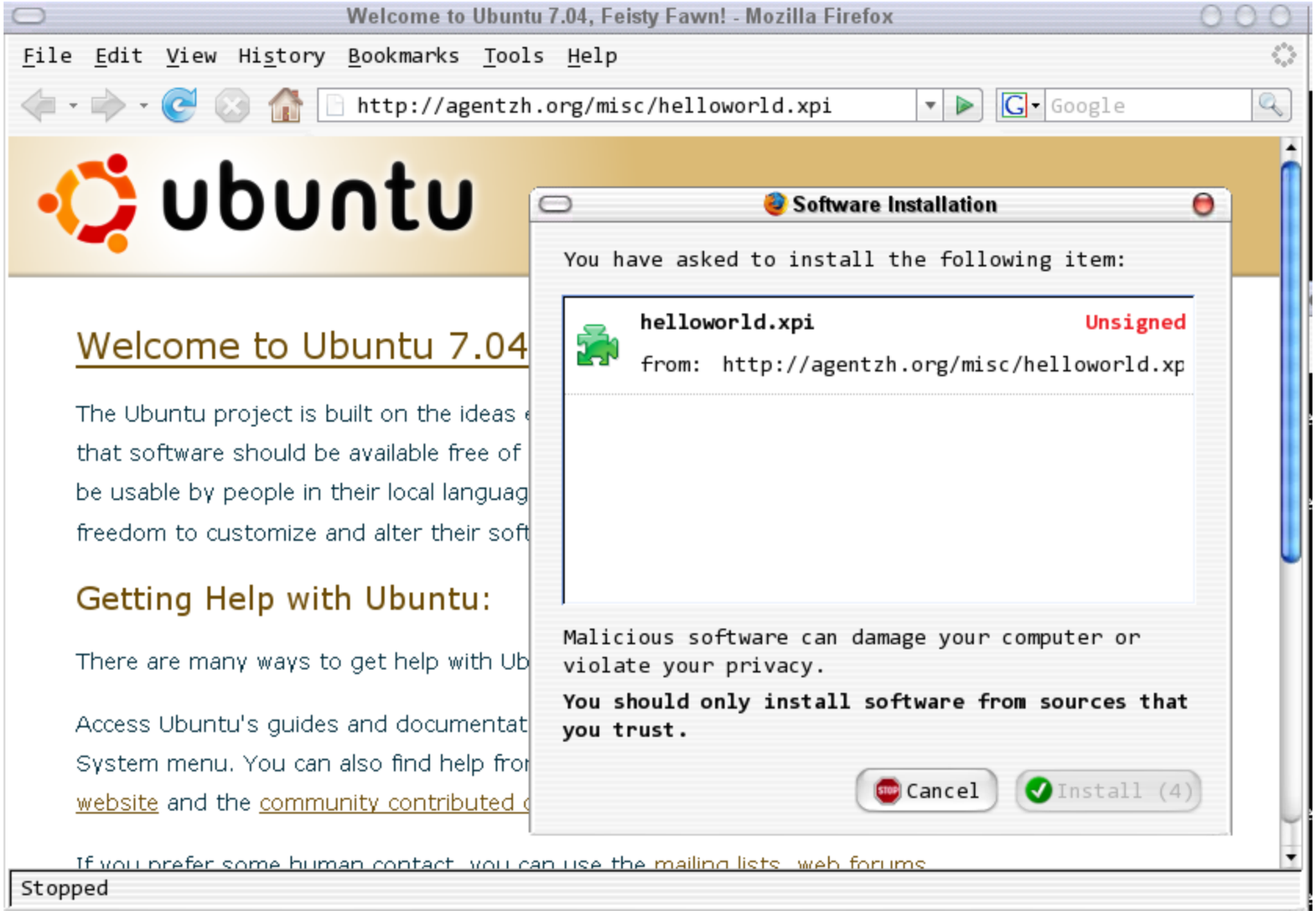
  (except Firefox itself)
- ✓ **runs happily *everywhere***

  (Win32, Linux, Mac, and etc.)

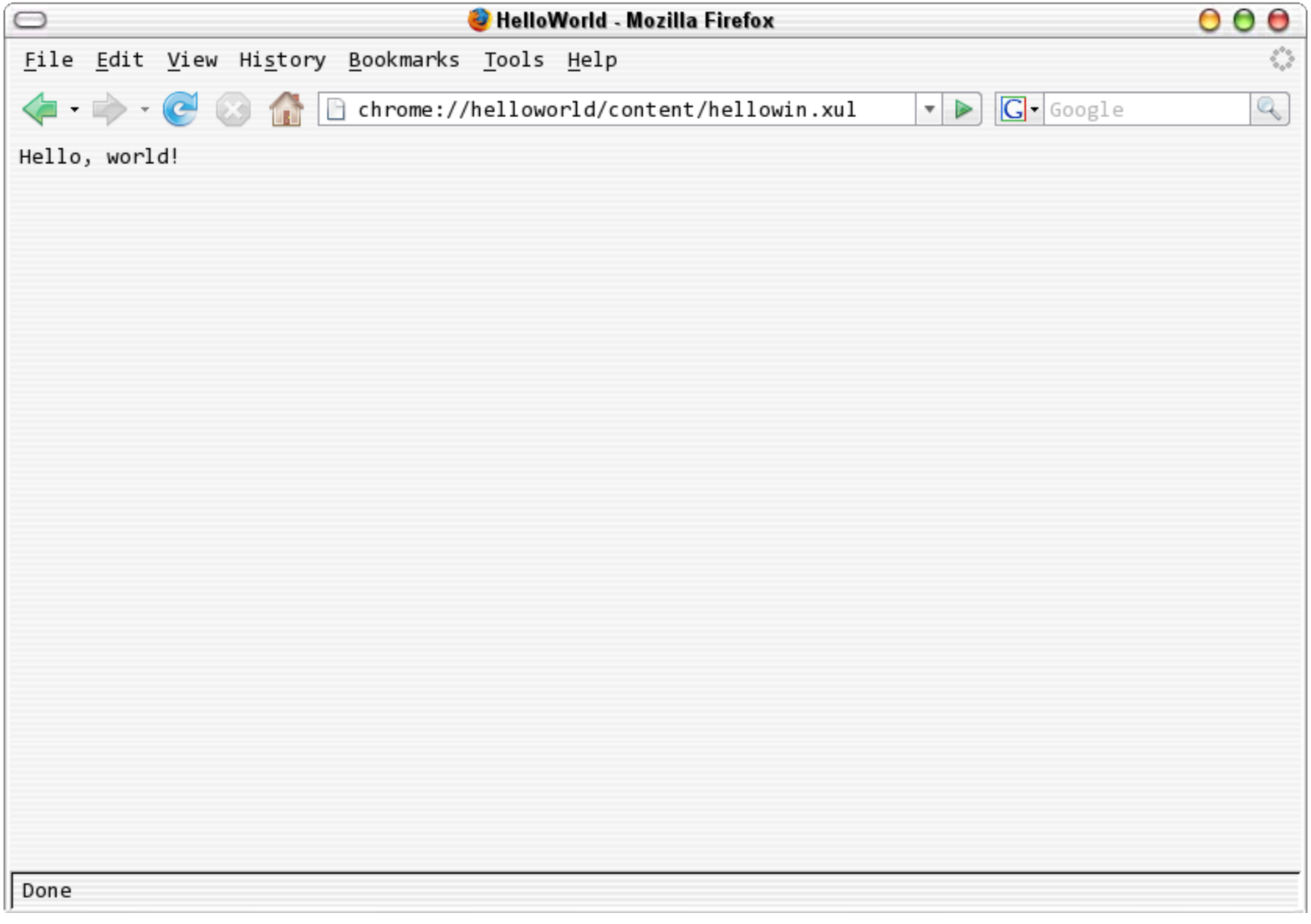
☺ Put helloworld.xpi onto an *HTTP server* and make sure the mime-type for .xpi is application/x-xpinstall.

将 helloworld.xpi 放到一个 HTTP 服务器上，并确保 .xpi 的 mime 类型为 application/x-xpinstall.

On the *end user* side...



On the *end user* side...



☹ But requiring the *end user* to enter the chrome URL to access our extension is *NOT* professional.

但是要求最终用户靠输入 *chrome URL* 来访问我们的扩展是很不专业的。

☼ Let's add a button to
Firefox's *Tools menu* for our baby then!

那么就让我们来为我们的东东
向 Firefox 工具菜单添加一个按钮吧！

☺ First of all, create an *overlay view* named Overlay!

让我们来创建一个名叫 Overlay 的覆盖视图!

```
$ xulapp view --name Overlay --type overlay
 Writing file lib/HelloWorld/View/Overlay.pm
```

```
$ wc -l lib/HelloWorld/View/Overlay.pm
 27 lib/HelloWorld/View/Overlay.pm
$
```

☺ Now let's *register* our Overlay view
in our HelloWorld::App module.

现在让我们在 HelloWorld::App 模块中
*注册*一下我们的 Overlay 视图。

```
$ vim lib/HelloWorld/App.pm
```

```
# File lib/HelloWorld/App.pm
package HelloWorld::App;
our $$VERSION = '0.01';
use XUL::App::Schema;
use XUL::App schema {
    # Code that we added by hand:
    xulfile 'hellowin.xul' =>
        generated from 'HelloWorld::View::HelloWin';

    xpifile 'helloworld.xpi' =>
        name is 'HelloWorld',
        id is 'helloworld@agentz.agentz-office',
        version is '0.0.1',
        ...
```

```
# File lib/HelloWorld/App.pm
package HelloWorld::App;
our $$VERSION = '0.01';
use XUL::App::Schema;
use XUL::App schema {
    # Code that we added by hand:
    xulfile 'overlay.xul' =>
        generated from 'HelloWorld::View::Overlay',
        overlays 'chrome://browser/content/browser.xul';

    xulfile 'hellowin.xul' =>
        generated from 'HelloWorld::View::HelloWin';

    xpifile 'helloworld.xpi' =>
        name is 'HelloWorld',
        ...
```

☺ Now let's add some *contents* to our *Overlay* view class.

现在让我们往 Overlay 视图类中添加一些*内容*。

```
# File lib/HelloWorld/View/Overlay.pm
...
package HelloWorld::View::Overlay;
use base 'XUL::App::View::Base';
use Template::Declare::Tags 'XUL';
 template main => sub {
     show 'header';  # from XUL::App::View::Base
     overlay {
         attr {
             id => "helloworld-overlay",
             xmlns => $::XUL_NAME_SPACE,
         }
         # Add your elements here...

     }
 };
 1;
```

```
# File lib/HelloWorld/View/Overlay.pm
...
 template main => sub {
     show 'header';  # from XUL::App::View::Base
     overlay {
         ...
         # Add your elements here...
         menupopup {
             attr { id => "menu_ToolsPopup" }
             menuitem {
                 attr {
                     oncommand => "toOpenWindowByType(
                         'helloworld',
                         'chrome://helloworld/content/hellowin.xul')",
                     insertafter =>  "javascriptConsole,devToolsSeparator",
                     label => "Hello World",
                 }
             }
         }
         ...
```

☆ Job's *done*! Let's test it!

工作*完成*了！让我们来测试它！

```
$ xulapp overlay --profile dev
 Writing file overlay.xul
 Writing file hellowin.xul
 Registering extension hellowin in profile dev
 Setting configure variables in dev's prefs.js
 firefox -jsconsole -P dev
```


Welcome to Ubuntu 7.04, Feisty Fawn! - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///usr/share/ubuntu-
Google
Getting Started
Latest BBC Headlines
ubuntu
Welcome to Ubuntu 7.04, Feisty Fawn!
The Ubuntu project is built on the ideas enshrined in the Ubuntu philosophy: that software should be available free of charge, that software tools should be usable by people in their local language, and that people should have the freedom to customize
Done

♡ ***rebundle*** **it!**

重新打包！

```
$ xulapp bundle .
 Writing file overlay.xul
 Writing file hellowin.xul
 Writing bundle file ./helloworld.xpi
$
```

- "Hey, I want to support *multiple languages*,
  my users come from different countries."

嘿，我想要支持*多国语言*,
我的用户来自不同的国家.

☺ *No problem,*
it's *easy* to implement with XUL::App!

没问题,
使用 XUL::App 来实现是*很容易的*！

☼ Let's Add a *zh-CN* locale
to our extension!

让我们为我们的扩展
添加一个简体中文的 locale!

Step 1 Change every occurrence of
string literals "..." in our view classes
to the form _("...")

第一步 将我们的视图类中出现的
每一个字符串字面值 "..."
都替换为 _("...") 的形式。

```
# File lib/HelloWorld/View/HelloWin.pm
package HelloWorld::View::HelloWin;
use base 'XUL::App::View::Base';
use Template::Declare::Tags 'XUL';
 template main => sub {
    show 'header';  # from XUL::App::View::Base
    window {
        attr {
            id => "helloworld-hellowin",
            xmlns => $::XUL_NAME_SPACE,
            title => 'HelloWorld',
            ...
        }
        # Code that we added by hand:
        label { "Hello, world!" }
    }
    ...
```

```
# File lib/HelloWorld/View/HelloWin.pm
package HelloWorld::View::HelloWin;
use base 'XUL::App::View::Base';
use Template::Declare::Tags 'XUL';
 template main => sub {
     show 'header';  # from XUL::App::View::Base
     window {
         attr {
             id => "helloworld-hellowin",
             xmlns => $::XUL_NAME_SPACE,
             title => _('HelloWorld'),
             ...
         }
         # Code that we added by hand:
         label { _("Hello, world!") }
     }
     ...
```

```
# File lib/HelloWorld/View/Overlay.pm
...
    # Add your elements here...
    menupopup {
        attr { id => "menu_ToolsPopup" }
        menuitem {
            attr {
                oncommand => "toOpenWindowByType(
                    'helloworld',
                    'chrome://helloworld/content/hellowin.xul')",
                insertafter => "javascriptConsole,devToolsSeparator",
                label => "Hello World",
            }
        }
    }
...
```

```
# File lib/HelloWorld/View/Overlay.pm
...
    # Add your elements here...
    menupopup {
        attr { id => "menu_ToolsPopup" }
        menuitem {
            attr {
                oncommand => "toOpenWindowByType(
                    'helloworld',
                    'chrome://helloworld/content/hellowin.xul')",
                insertafter => "javascriptConsole,devToolsSeparator",
                label => _("Hello World"),
            }
        }
    }
...
```

Step 2 Generate the *PO file* for zh-cn.

第二步 为我们的 zh-cn 生成 *PO 文件*.

```
$ xulapp po --lang zh-cn
Write po/zh-cn.po
```

```
$ wc -l po/zh-cn.po
 28 po/zh-cn.po
```

Step 3 Edit the *PO file* to do the *actual* translation

第三步 编辑 *PO 文件*进行实际的翻译工作。

```
$ vim po/zh-cn.po
```

```
...
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: lib/HelloWorld/View/Overlay.pm:28
msgid "Hello World"
msgstr ""

#: lib/HelloWorld/View/HelloWin.pm:23
msgid "Hello, world!"
msgstr ""

#: lib/HelloWorld/View/HelloWin.pm:17
msgid "HelloWorld"
msgstr ""
```

```
...
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

#: lib/HelloWorld/View/Overlay.pm:28
msgid "Hello World"
msgstr "你好 世界"

#: lib/HelloWorld/View/HelloWin.pm:23
msgid "Hello, world!"
msgstr "你好，世界"

#: lib/HelloWorld/View/HelloWin.pm:17
msgid "HelloWorld"
msgstr "你好世界"
```

☺ Done and done; let's *test* it!

搞定了，搞定了；
让我们*测试*一下！

```
$ xulapp overlay --profile dev --lang zh-cn
 Writing file zh-CN.dtd
 Writing file overlay.xul
 Writing file hellowin.xul
 Registering extension hellowin in profile dev
 Setting configure variables in dev's prefs.js
 LANG="zh-CN.UTF-8" LC_CTYPE="zh-CN.UTF-8" firefox -jsconsole -P dev
```


Welcome to Ubuntu 7.04! - Mozilla Firefox
文件 (F) 编辑 (E) 查看 (V) 历史 (S) 书签 (B) 工具 (T) 帮助 (H)
file:///usr/share/ubuntu
Yahoo
Getting Started
Latest BBC Headlines
ubuntu
Welcome to Ubuntu 7.04!
Ubuntu项目是建立在信奉 Ubuntu Philosophy基础上的：软件应免费提供，软件工具应能以人们本地语种的形式可用且不牺牲任何功能，人们应拥有定制及改变他们软件的自由，这包括以任何他们认为适宜的方式。
获得Ubuntu的帮助：
完成

♡ Let's create an en-US locale
as the *fall back*.

让我们创建一下 en-US 作为默认方式。

```
$ xulapp po --lang en-us
Write po/en-us.po
```

```
...
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

#: lib/HelloWorld/View/Overlay.pm:28
msgid "Hello World"
msgstr ""

#: lib/HelloWorld/View/HelloWin.pm:23
msgid "Hello, world!"
msgstr ""

#: lib/HelloWorld/View/HelloWin.pm:17
msgid "HelloWorld"
msgstr ""
```

☺ Strings that're *not* translated are kept intact.

没有被翻译的字符串会保持原样。

☺ Let's *test* the en-US locale now.

现在让我们测试一下 en-US 语言支持。

```
$ xulapp overlay --profile dev --lang en-us
 Writing file en-US.dtd
 Writing file overlay.xul
 Writing file hellowin.xul
 Registering extension hellowin in profile dev
 Setting configure variables in dev's prefs.js
 LANG="en-US.UTF-8" LC_CTYPE="en-US.UTF-8" firefox -jsconsole -P dev
```


Welcome to Ubuntu 7.04, Feisty Fawn! - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///usr/share/ubuntu-
Google
Getting Started
Latest BBC Headlines
ubuntu
Welcome to Ubuntu 7.04, Feisty Fawn!
The Ubuntu project is built on the ideas enshrined in the Ubuntu philosophy: that software should be available free of charge, that software tools should be usable by people in their local language, and that people should have the freedom to customize
Done

☺ We can add as *many* locales as we wish ;)

我们想加多少种语言支持，
就可以加多少 ;)

What if our view classes' .pm files *change*?
Do we have to *redo* the whole translation?

如果我们*改变*了视图类的 .pm 文件的话呢？
我们是否必须*重做*所有的翻译？

```
$ xulapp po
Updated po/zh-en.po
Updated po/zh-cn.po
```

☺ XUL::App will try its best to *reuse*
the existing translation items in the .po files.

XUL::App 将会尽最大努力*复用*
.po 文件中已有的翻译条目。

☺ Now we can *rebundle* our baby
and make both Chinese and US users happy.

现在我们可以*重新打包*我们的孩子
来让中国用户和美国用户同时开心。

```
$ xulapp bundle .
 Writing file overlay.xul
 Writing file hellowin.xul
 Writing file en-US.dtd
 Writing file zh-CN.dtd
 Writing bundle file ./helloworld.xpi
$
```

♡ any questions? ♡