第 3 章 运行 ~~Symfony~~symfony

如上章所述，~~Symfony~~symfony 是由许多 PHP 文件组成的框架。 ~~Symfony~~symfony 的项目需要使用这些文件，所以安装 ~~Symfony~~symfony 其实就是让项目中可以使用这些文件。

~~Symfony~~symfony 是基于 PHP5 的框架。所以用以下命令确认你安装了正确的 PHP 版本：

```
> php -v

PHP 5.2.0 (cli) (built: Nov 2 2006 11:57:36)
Copyright (c) 1997-2006 The PHP Group
Zend Engine v2.2.0, Copyright (c) 1998-2006 Zend Technologies
```

如果版本号大于 5.0，你就可以开始安装了，安装过程将在此章节介绍。

安装沙盒(Sandbox)

如果你只是想要快速安装，试用一下 ~~Symfony~~symfony，你应该使用沙盒。

沙盒里有一个空的 ~~Symfony~~symfony 项目,这个项目包括基本的配置，一个默认的应用程序，还有 ~~Symfony~~symfony 所需要的库(symfony、pake、lime、Creole、Propel 和 Phing)。 它可以独立运行，不需要特别的服务器配置。

沙盒可以从 http://www.symfony-project.com/get/sf_sandbox.tgz 下载。解压缩到 web 服务器的根目录中 (通常是 web/ 或者 www/)。 为了统一性， 本章将假设你把它解压到 sf_sandbox 目录下。

NOTE 把所有的文件放在 web 根目录下对于测试没有什么问题，不过在正式服务器上这么作是一个坏习惯。这样会把所有程序的内部文件暴露给最终用户。

执行 ~~Symfony~~symfony 命令来测试安装是否成功。 在 sf_sandbox/目录下，输入以下命令： Linux 系统下：

```
> ./symfony -V
```

Windows 系统下：

```
> symfony -V
```

你会看到沙盒的版本号：

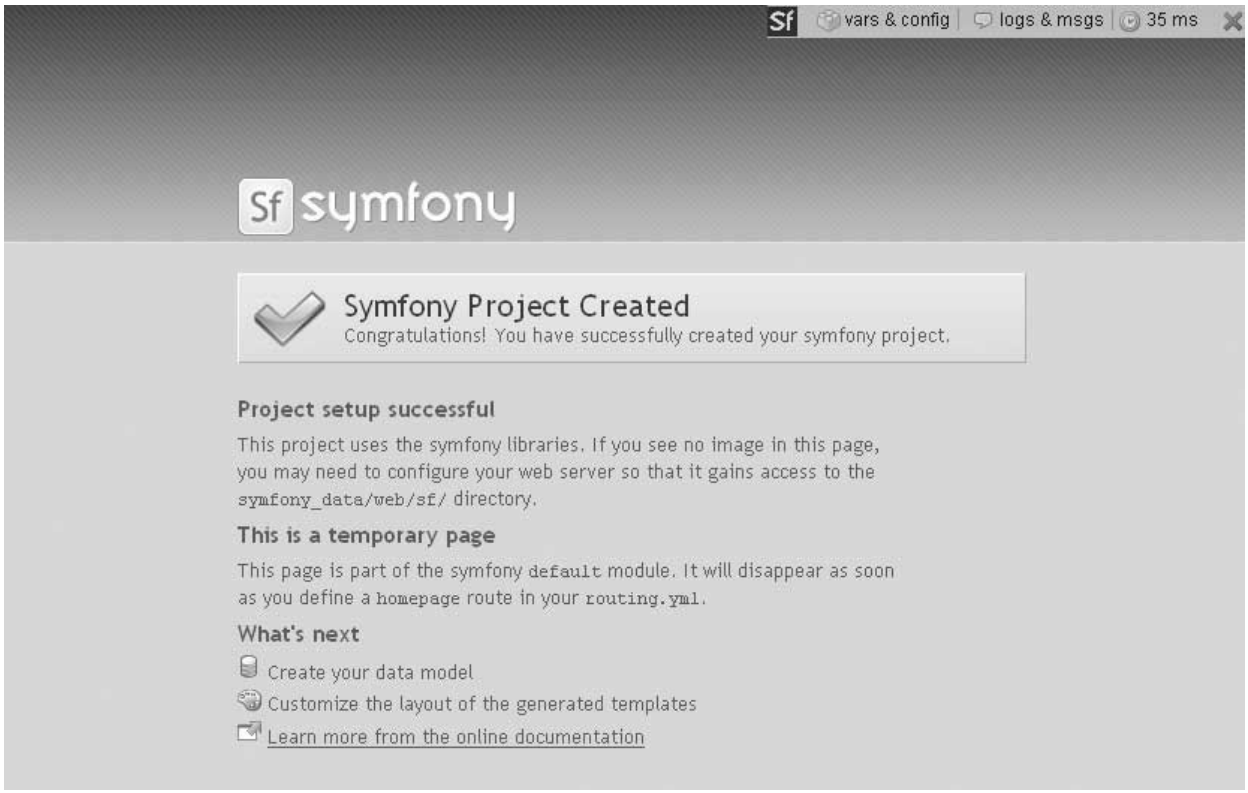
```
symfony version 1.0.0
```

现在请确认你的 web 服务器可以从下面的地址来访问沙盒：

http://localhost/sf_sandbox/web/frontend_dev.php/

如果你看到一个类似图 3-1 一样的成功页面，这就意味着安装已经完成。 如果没看到， 将会有一个错误信息告诉你如何去修改配置文件。 你也可以参考下面的"安装问题"章节。

图 3-1 - 沙盒的祝贺页面



沙盒是用来给你在自己的电脑上面练习的，并不适合开发复杂的应用程序。不过，沙盒里的 symfony 的功能是完整的，与通过 PEAR 安装的没有差别。

要卸载沙盒，只要把 web/目录下的 sf_sandbox/删除即可。

安装 ~~Symfony~~symfony 库

开发程序的时候， 你也许会安装 ~~Symfony~~symfony 两次： 一次是你的开发环境，另外一次是在服务器上(除非服务器上已经安装过 ~~Symfony~~symfony)。对于每台服务器而言，为了避免重复你也许会把所有的 symfony 文件放在一个地方，不管你开发几个程序。

因为 ~~Symfony~~symfony 框架更新的很快，一个新的稳定版本可能在你安装后的几天内就发布了。 所以你需要认真考虑 symfony 框架更新的问题，这也是另外一个所有的项目应该共用同一个 symfony 的理由。

当要在真正程序开发中安装库的时候,你有 2 个选择：

- 对大多数人而言推荐用 PEAR 安装方式。 他很容易共享和升级，安装过程直接了当。
- Subversion (SVN) 安装模式通常是高级 PHP 程序开发者使用的， 可以获得最新的补丁，增加自己开发的功能，发布 ~~Symfony~~symfony 的项目。

~~Symfony~~symfony 集成了一些其他的包：

- pake 是一个命令行工具。
- lime 是单元测试工具。
- Creole 是数据库抽象引擎。类似于 PHP 数据对象 (PDO)，他提供了程序代码与 SQL 数据库代码之间的一个接口，以便切换到其他数据库。
- Propel 是 ORM 工具。它提供持续对象与查询服务。
- Phing 是 Propel 的命令行接口。

Pake 和 lime 是 symfony 小组开发的。Creole、Propel 和 Phing 是由其他小组开发并置于 GNU Lesser Public General License (LGPL)协议下。所有这些包都绑定在 ~~Symfony~~symfony 中。

Pear 方式安装 ~~Symfony~~symfony

~~Symfony~~symfony 的 PEAR 包包含了 symfony 库。它也包含一个将 symfony 命令加入你的命令行的脚本。

安装第一步是把 symfony 频道加入 PEAR，执行以下命令：

```
> pear channel-discover pear.symfony-project.com
```

用以下命令查看这个频道中的可用库列表：

```
> pear remote-list -c symfony
```

现在可以安装稳定版本的 symfony 了。 执行以下命令：

```
> pear install symfony/symfony

downloading symfony-1.0.0.tgz ...
Starting to download symfony-1.0.0.tgz (1,283,270 bytes)
.....................................................................
.....................................................................
.............done: 1,283,270 bytes
install ok: channel://pear.symfony-project.com/symfony-1.0.0
```

~~Symfony~~symfony 文件和命令行工具已经安装好了。在命令行执行 symfony 来确

认安装是否成功， 查看版本号：

```
> symfony -V

symfony version 1.0.0
```

TIP 如果要安装最新的beta版本， 用命令 pear install symfony/symfony-beta 来安装。 Beta版本通常是不稳定的，不推荐在生产环境中用。

~~Symfony~~symfony 库安装在以下的目录中：

- $php_dir/symfony/ 存放主要库文件。
- $data_dir/symfony/ 存放~~Symfony~~symfony 程序的结构；默认的模块；配置文件，i18n数据，和其他。
- $doc_dir/symfony/ 存放文档。
- $test_dir/symfony/ 存放单元测试。

_dir结尾的变量是PEAR配置的一部分，可以用以下命令查看它的值：

```
> pear config-show
```

从SVN库中获得

对于生产服务器， 或者不能从PEAR安装的时候， 你可以直接从symfony的subversion库里通过捡出最新版的symfony：

```
> mkdir /path/to/symfony
> cd /path/to/symfony
> svn checkout http://svn.symfony-project.com/tags/RELEASE_1_0_0/ .
```

命令symfony，只有在用PEAR安装时候才有效，实际上是调用了/path/to/symfony/data/bin/symfony脚本。 所以SVN安装模式需就用下面的命令查看版本：

```
> php /path/to/symfony/data/bin/symfony -V

symfony version 1.0.0
```

如果选择用SVN安装方式，你也许已经有了一个symfony项目。你需要在你的项目中更改两个变量使其可以找到symfony文件：

```
<?php

$sf_symfony_lib_dir  = '/path/to/symfony/lib/';
$sf_symfony_data_dir = '/path/to/symfony/data/';
```

第 19 章提供了用另外一种把项目和 symfony 安装关联起来(包括了符号连接和相对路径)的方法。

TIP 或者，你也可以下载 PEAR 包（http://pear.symfony-project.com/get/symfony-1.0.0.tgz）然后把它解压.这与从版本库中取得是一样的。

配置一个程序

就如在第 2 章所学，symfony 把相关的应用程序放在项目中。项目中所有的应用程序都共享同一个数据库。为了设置一个应用程序，你必须先建立一个项目。

建立一个项目

每个 symfony 项目都有一个预定义的目录结构。~~Symfony~~symfony 命令行会自动建立一个新的项目的框架，合适的树状结构和访问权限。所以要建立一个新项目，只需简单的建立一个目录，然后告诉 symfony 在此建立一个项目即可。

PEAR 安装方式用以下命令：

```
> mkdir ~/myproject
> cd ~/myproject
> symfony init-project myproject
```

SVN 安装方式，用以下命令建立项目：

```
> mkdir ~/myproject
> cd ~/myproject
> php /path/to/symfony/data/bin/symfony init-project myproject
```

symfony 命令只能在项目的根目录下使用(前面提到的例子 myproject/)，因为用这个命令所作的事情都是基于项目的。

~~Symfony~~symfony 会建立一个下面这样的目录结构：

```
apps/
batch/
cache/
config/
data/
doc/
lib/
log/
plugins/
test/
```

```
web/
```

TIP init-project 在项目根目录中增加一个 symfony 脚本。这个 PHP 脚本和用 PEAR 安装的 symfony 命令做的一样，所以没有原生命令行(SVN 安装方式)支持的话你可以用 php symfony 来替代 symfony。

建立一个应用程序

项目现在还没法用，因为他还至少需要一个应用程序。用 symfony init-app 命令传送一个应用程序的名字作为一个参数去初始化它：

```
> symfony init-app myapp
```

这将在项目根的 apps/目录下建立一个叫 myapp/的目录，它包含了一个默认的应用程序配置和一系列的子目录：

```
apps/
  myapp/
    config/
    i18n/
    lib/
    modules/
    templates/
```

在项目 web 目录里还会建立这个应用程序的两个默认环境对应的前端控制器的 PHP 文件：

```
web/
  index.php
  myapp_dev.php
```

index.php 是新建应用程序的生产环境前端控制器。 因为当你在项目中创建了第一个应用程序的时候，symfony 建立一个叫 index.php 的文件来代替 myapp.php (如果你现在增加一个新的应用程序 mynewapp， 新的生产环境前台会是 mynewapp.php)。 要在开发环境中呼叫前端控制器 myapp_dev.php 来运行你的应用程序，你会在第 5 章了解更多关于环境的知识。

配置 Web 服务器

web/目录下的这些脚本是应用程序的入口。 Web 服务器要先配置才能在 Internet 中被访问。在开发服务器中，与专业主机服务解决方案一样，也需要访问 Apache 配置文件去设置一个虚拟主机。 在一个共享的服务器中，你也许只能修改.htaccess 文件。

设置虚拟主机

例 3-1 是一个 Apache 配置的示例，可以了解如何在 httpd.conf 中添加新的虚拟主机。

例 3-1 - Apache 配置示例，apache/conf/httpd.conf

```
<VirtualHost *:80>
  ServerName myapp.example.com
  DocumentRoot "/home/steve/myproject/web"
  DirectoryIndex index.php
  Alias /sf /$sf_symfony_data_dir/web/sf
  <Directory "/$sf_symfony_data_dir/web/sf">
    AllowOverride All
    Allow from All
  </Directory>
  <Directory "/home/steve/myproject/web">
    AllowOverride All
    Allow from All
  </Directory>
</VirtualHost>
```

在例 3-1 配置中，需要用真实的路径替换掉/path/to/symfony/data。 例如，在*nix 中用 PEAR 安装的话，你要输入类似这样的配置行：

```
    Alias /sf /usr/local/lib/php/data/symfony/web/sf
```

NOTE 给 web/sf/设置别名并非是强制的。 这可以让 Apache 找到网页 debug 工具条所用的图片，样式表和 JavaScript 文件， 管理界面生成器， symfony 的默认页面还有 Ajax 支持。另外一个设置别名的方法是建立一个符号连接(symlink)或者把/path/to/symfony/data/web/sf/复制到 myproject/web/sf/中。

重启 Apache。 你新建立的应用程序现在能用以下的 URL 来访问：

http://localhost/myapp_dev.php/

你可以看到一个类似先前图 3-1 的成功页面。

SIDEBAR URL 重写

~~Symfony~~symfony 用 URL 重写来显示"漂亮的 URL"--有意义的地址对搜索引擎更友好并且对使用者隐藏了所有的数据。你会在第 9 章学习更多这方面的知识--路由(routing)。

如果你的 Apache 编译的时候没有选择 mod_rewrite 模块，检查 httpd.conf 中是否包含了 mod_rewrite 动态共享对象(DSO)。

```
AddModule mod_rewrite.c LoadModule rewrite_module
modules/mod_rewrite.so
```

对于 Internet 信息服务(IIS)，你需要安装和运行 isapi/rewrite。
~~Symfony~~symfony 在线手册有关于 IIS 安装的详细指导。

配置一个共享服务器

在共享服务器上配置一个应用程序有一些麻烦，因为服务器通常有你无法改变的特殊目录结构。

CAUTION 直接在共享服务器上测试和开发并不是一个好的做法。一个原因是因为这会让应用程序在未完成的时候可以被访问到，泄露其内部信息将会带来很大的安全隐患。另一个原因是,共享主机的性能往往不能满足开启调试工具时快速浏览应用程序的需要。因此你不应在共享服务器上开始你的开发，而应该在本地建立你的应用程序，完成后再移植到共享服务其上。第 16 章将告诉你移植技术和工具。

我们假设你的共享服务器需要把网页目录命名为 www/来取代 web/，并且不允许你修改 httpd.conf 而只允许你修改在网页目录中的.htaccess 文件。

在一个 symfony 项目中，每一个目录路径都是可配置的。第 19 章将带给你更多的信息，但是与此同时，你可以把 web 目录改为 www 目录，并且修改应用程序的配置，就如例 3-2 所示。这些将在应用程序的 config.php 文件底部加上。

例 3-2 - 在 apps/myapp/config/config.php 修改默认目录结构设定

```
$sf_root_dir = sfConfig::get('sf_root_dir');
sfConfig::add(array(
  'sf_web_dir_name' => $sf_web_dir_name = 'www',
  'sf_web_dir'      => $sf_root_dir.DIRECTORY_SEPARATOR.
$sf_web_dir_name,
  'sf_upload_dir'   => $sf_root_dir.DIRECTORY_SEPARATOR.
$sf_web_dir_name.DIRECTORY_SEPARATOR.sfConfig::get('sf_upload_dir_nam
e'),
));
```

项目的网页根目录默认包含了一个 .htaccess 文件。 就如例 3-3 所示。适当的修改它以配合你的共享服务器的需求。

例 3-3 - 默认的 .htaccess 配置，当前在 myproject/www/.htaccess

```
Options +FollowSymLinks +ExecCGI

<IfModule mod_rewrite.c>
```

```
  RewriteEngine On

  # we skip all files with .something
  RewriteCond %{REQUEST_URI} \..+$
  RewriteCond %{REQUEST_URI} !\.html$
  RewriteRule .* - [L]

  # we check if the .html version is here (caching)
  RewriteRule ^$ index.html [QSA]
  RewriteRule ^([^.]+)$ $1.html [QSA]
  RewriteCond %{REQUEST_FILENAME} !-f

  # no, so we redirect to our front web controller
  RewriteRule ^(.*)$ index.php [QSA,L]
</IfModule>

# big crash from our front web controller
ErrorDocument 500 "<h2>Application error</h2>symfony
applicationfailed to start properly"
```

现在可以访问你的应用程序了。你可以通过这个URL访问symfony成功页面：

http://www.example.com/myapp_dev.php/

SIDEBAR 其他服务器配置

~~Symfony~~symfony和其他服务器配置兼容。例如，你可以用alias代替虚拟主机来访问symfony程序。 你也能在IIS上运行symfony程序。关于配置有很多技巧， 本书并不准备解释所有的技巧。

想要找到具体的服务器配置指南，可以参考symfony的wiki (http://www.symfony-project.com/trac/wiki)，这里有详尽的指导。

安装问题

这些通常会有错误说明，甚至会提供网络上针对此问题的资源连接。如果在安装中遇到问题，尽量把错误或例外显示在shell或者浏览器上。

常见问题

如果你还是无法运行symfony，检查以下几点：

- 一些PHP环境同时包含了PHP4和PHP5的命令。 因此，在命令行用php5替代php， 也就是说试着用php5 symfony代替symfony。你也许在.htaccess配置中需要增加SetEnv PHP_VER 5参数，或者把web/目录

中的.php 换成.php5。在 PHP4 命令行下试着访问 symfony 就会有类似下面的提示：

```
Parse error, unexpected ',', expecting '(' in .../symfony.php on line 19.
```

- 在 php.ini 中的内存限制，至少需要设置为 16M。 通常的症状就是通过 PEAR 方式安装 symfony 的时候出现的错误信息。

```
Allowed memory size of 8388608 bytes exhausted
```

- 必须在 php.ini 中把 zend.ze1_compatibility_mode 参数设置为 off。否则通过浏览器去访问脚本的话会出现"implicit cloning"错误：

```
Strict Standards: Implicit cloning object of class
'sfTimer'because of 'zend.ze1_compatibility_mode'
```

- 在你的项目中位于 Web 服务器上的 log/和 cache/目录必须是可写的。如果在没有正确设定的时候访问 symfony 程序会出现以下提示：

```
sfCacheException [message] Unable to write cache
file"/usr/myproject/cache/frontend/prod/config/config_config_handlers.yml.php"
```

- 系统的路径需要包含 php 命令的路径，你的 php.ini 的包含路径必须包括 PEAR 的路径（如果你使用 PEAR）。
- 有时，服务器上会有多个 php.ini 文件(例如，如果你使用 WAMP 包)。可以用 phpinfo()函数去了解程序所用的 php.ini 文件所在的确切位置。

NOTE 虽然不是强制性的，但是这里强烈推荐，为了运行得更顺畅，在 php.ini 中设置 magic_quotes_gpc 和 register_globals 参数为 off。

~~Symfony~~symfony 资源

你可以在这些地方找到一些已经发现的问题的答案：

- symfony 安装论坛 (http://www.symfony-project.com/forum/) 这里有各种平台，环境配置，主机上安装 symfony 的问题讨论。
- 用户邮件列表档案(http://groups.google.fr/group/symfony-users)也可以搜索。你也许会找到一些人遇到同样的问题。
- symfony wiki (http://www.symfony-project.com/trac/wiki#Installingsymfony) 有由 symfony 用户提供的详细安装教程。

如果没有找到答案，试着把问题放到 symfony 社区。你可以在论坛，邮件列表

甚至在#symfony IRC 频道得到大家的回应。

源代码版本控制

设置程序完成后，推荐进行版本控制。 版本控制能跟踪对代码的所有修改，可以回退到以前的版本，更容易地给程序打补丁和更有效地进行团队合作开发。 ~~Symfony~~symfony 生来就支持 CVS，虽然更推荐使用 Subversion (http://subversion.tigris.org/)。下面的例子展示了 Subversion 的命令，我们假设你已经有 Subversion 服务器并且希望在项目中建立一个新的版本库。Windows 使用者推荐用叫做 TortoiseSVN (http://tortoisesvn.tigris.org/) 的 Subversion 客户端。在 Subversion 文档中可以找到关于版本控制命令的更多信息。

下面的例子假设系统环境参数中已经定义了 $SVNREP_DIR。 如果还没有定义，你要以实际存放位置代替 $SVNREP_DIR。

让我们在 myproject 项目中建立一个新的版本库：

```
> svnadmin create $SVNREP_DIR/myproject
```

建立 trunk、 tags 和 branches 作为版本库的基础结构(layout)用以下命令：

```
> svn mkdir -m "layout creation" file:///$SVNREP_DIR/myproject/trunk
file:///$SVNREP_DIR/myproject/tags file:///
$SVNREP_DIR/myproject/branches
```

这会是你的第一个版本。现在你需要把项目中除 cache/ 和 log/ 目录之外所有的文件导入版本库：

```
> cd ~/myproject
> rm -rf cache/*
> rm -rf log/*
> svn import -m "initial import" . file:///
$SVNREP_DIR/myproject/trunk
```

输入以下命令检查已经提交的文件：

```
> svn ls file:///$SVNREP_DIR/myproject/trunk/
```

看上去没问题。现在 SVN 库包含了你所有的项目文件的参考版本（还有历史）。这意味着~myproject/目录需要与 SVN 库关联。要实现关联，首先修改 myproject/目录的名字（如果一切正常你很快就可以删了它了）然后在一个新目录里签出 SVN 库里的文件：

```
> cd ~
```

```
> mv myproject myproject.origin
> svn co file:///$SVNREP_DIR/myproject/trunk myproject
> ls myproject
```

现在你可以改写~/myproject/下的文件并提交到版本库中去。
myproject.origin/目录已经没用了，别忘了把它删除掉。

还有一件事情需要配置。 如果你提交当前工作目录到版本库中， 也许包含了一些无用的文件，例如项目中的 cache 和 log 目录。所以必须为这个项目设置一个 SVN 忽略列表。当然，你还需要重新设置 cache/ 和 log/目录的权限：

```
> cd ~/myproject
> chmod 777 cache
> chmod 777 log
> svn propedit svn:ignore log
> svn propedit svn:ignore cache
```

SVN 默认的文字编辑器会启动。如果没有，在 Subversion 中设置你想用的文字编辑器：

```
> export SVN_EDITOR=<name of editor>
> svn propedit svn:ignore log
> svn propedit svn:ignore cache
```

现在只要把 myproject/子目录的所有文件都添加到 SVN 中，SVN 会在提交的时候忽略掉列表中的文件：

```
*
```

保存，退出。完成了。

总结

如果在本地服务器上想测试或者尝试一下 symfony，最好安装一个已经预配置好环境的沙盒。

如果是真正的开发或者在生产服务器上，最好使用 PEAR 安装或者用 SVN 捡出。这样做会安装 symfony 的库，还需要初始化一个项目和应用程序。应用程序设置的最后一步是服务器配置，这有很多种做法。symfony 能完美地在虚拟主机下运行，这也是推荐的解决方案。

如果在安装中遇到问题， 你能从 symnfony 网站上找到许多教程、回答、FAQ。如果需要，你可以把问题提交到 symfony 社区，这样会很快得到答案。

当项目初始化好后， 开始版本控制流程是一个好习惯。

现在你已经准备好使用 symfony 了，是时候去建立一个基础的网页程序了。