

《RAILS 3 框架应用程序图例》

www.lycom.de

.~. 道喜技术博客 .~. .~. 天天红玉世界 .~.

德国 · 慕尼黑

初级水平

Rails 程序模板与 **Bundler** 命令

– 命令 *bundle check* 和 *bundle install* –

作者信息:

作者姓名:	骆古道
QQ 号:	
MSN 号:	hxidao@hotmail.com
Skype 号:	gudao
地址:	Saturnstr. 48, 85609 Aschheim, Germany
手机:	0049 172 9305165
电子邮件地址:	gudao.luo@gmail.com

于 德国 · 慕尼黑, 2011.01.03-2011.01.25

目录

1	摘要	1
2	正文	2
2.1	命令 BUNDLE CHECK	2
2.1.1	程序模板文件代码	3
2.1.2	程序模板文件运行命令	4
2.1.3	程序模板文件命令与无程序模板文件命令比较	5
2.1.4	程序模板文件命令运行正确结果	6
2.2	命令 BUNDLE INSTALL	7
2.2.1	程序模板文件代码	7
2.2.2	程序模板文件运行命令	8
2.2.3	程序模板文件命令和无程序模板文件命令比较	9
2.2.4	程序模板文件运行命令正确结果	10
2.3	深入命令 BUNDLE INSTALL	12
2.3.1	程序模板文件代码	12
2.3.2	程序模板文件运行命令	13
2.3.3	运行程序模板文件命令正确结果	14
2.3.4	程序模板文件命令和无程序模板文件命令比较	15
3	结语 / 结束语	16
4	附录	I
5	参考文献附录	II

代码目录

代码 1: Rails 程序模板文件 bundle_check_001.templ.rb.....	3
代码 2: Rails 程序模板文件 bundle_check_001 完全等价代码.....	3
代码 3: Rails 程序模板文件 bundle_install_002.templ.rb.....	7
代码 4: Rails 程序模板文件 bundle_install_003.templ.rb.....	13

命令目录

命令 1: 本地 Rails 程序模板文件 bundle_check_001.templ.rb 命令.....	4
命令 2: 远程 Rails 程序模板文件 bundle_check_001.templ.rb 命令.....	4
命令 3: 删除已创建的 Rails 应用程序 bundle_check_001 (可选命令)	4
命令 4: 本地 Rails 程序模板文件 bundle_install_002.templ.rb 命令.....	8
命令 5: 远程 Rails 程序模板文件 bundle_install_002.templ.rb 命令.....	8
命令 6: 删除已创建的 Rails 应用程序 bundle_install_002 (可选命令)	8
命令 7: 本地 Rails 程序模板文件 bundle_install_003.templ.rb 命令.....	14
命令 8: 远程 Rails 程序模板文件 bundle_install_003.templ.rb 命令.....	14
命令 9: 删除已创建的 Rails 应用程序 bundle_install_003 (可选命令)	14

图示目录

图 1: 命令“rails new“详细说明.....	4
图 2: 程序模板文件 bundle_check_001.tpl.rb 命令与无程序模板文件命令比较.....	5
图 3: 本地程序模板文件 bundle_check_001.tpl.rb 运行部分结果.....	6
图 4: 远程程序模板文件 bundle_check_001.tpl.rb 运行部分结果.....	6
图 5: 程序模板文件 bundle_install_002.tpl.rb 命令与无程序模板文件命令比较.....	9
图 6: 本地 Rails 程序模板文件 bundle_install_002.tpl.rb 运行部分结果.....	10
图 7: 远程 Rails 程序模板文件 bundle_install_002.tpl.rb 运行部分结果.....	11
图 8: 本地 Rails 程序模板文件 bundle_install_003.tpl.rb 运行部分结果.....	15
图 9: 远程 Rails 程序模板文件 bundle_install_003.tpl.rb 运行部分结果.....	16
图 10: 程序模板文件 bundle_install_003.tpl.rb 命令与无程序模板文件命令比较.....	17
图 11: 程序模板文件代码比较.....	18
图 12: 两个 Gemfile 文件代码结果.....	18

附录目录

A:	I
----------	---

1 摘要

在 Rails 3 框架中，引入了一个最重要的 gem 包是 Bundler。Bundler 包是 Ruby 语言软件包管理系统。我们将会逐步地和详细地说明，如何正确地使用该软件包的一些重要命令，为什么要使用这样的命令，在使用过程中应该注意哪些问题。

在这一篇文章中，我们将会说明如何使用软件包 Bundler 的命令：**bundle check** 和 **bundle install**。

Rails 程序模板文件是创建 Rails 应用程序的一个扩充方法。它为我们提供了一种记录创建程序的整个过程的**轨迹**。

在系统终端中，该文件的每一行代码可以找到相应的命令；执行这样或那样的一组命令，往往这种组合常常是**随意**

的，但是其组合也是想达到一种目的。而 Rails 模板文件组合的命令要达到我们**既定**的和可以**重复**执行命令为目标的。

对于一个应用程序而言，软件包命令需要运行的次数是不一样的。有一些命令需要也只能**执行一次**；有一些

命令视需要可以**执行多次**。需要执行多次的命令往往是因为应用程序本身或者其环境等发生了变化。

本文重点是要说明使用和理解 Rails 程序模板文件的**意义**所在。

所有模板文件名称是这样**组成**的：应用程序名称_应用程序唯一号.tmpl.rb。

2 正文

我们将需要如下操作系统和语言运行环境：

- 互联网
- Mac OS X 版本 10.5.8
- Ruby 版本 1.9.2-p136
- Rails 版本 3.0.3
- SQLite3 版本 3.6.23.1

我们介绍这种软件包的命令，都是以 Rails 程序模板方式进行叙述的。其好处是便于读者能够以最直接、最清晰、最简单和最快速的方式重复地运行需要学习的命令。

2.1 命令 **bundle check**

软件包 Bundler 的命令 `bundle check` **含义** 是什么？它是确定是否已经安装了你的应用程序需求，并且对于软件包 Bundler 而言，这种需求是可用的。其原文是：“Determine whether the requirements for your application are installed and available to bundler”。

这一段 **解释** 里面有几层意思。首先，这种需求是源于程序需求的说明；其次，这仅仅是为了检测这种程序需求是否满足软件包之间相关的依赖性，也就是说，我们只是在进行一种判断过程；最后，这种需求是以代码文件名称为 **Gemfile** 形式表现的。

2.1.1 程序模板文件代码

Rails 模板文件代码 **想要** 完成事情是：在创建 Rails 应用程序以后，运行开发应用程序所必需的 **初始化** 代码，如命令 `bundle check`。

```
# bundle_check_001.tmp1.rb
def echo_recipe(name, tag)
  say "\033[36m" + "echo #{tag} ".rjust(10) + "\033[0m" + " Running #{name} recipe..."
end

echo_recipe "command bundle check", "begin"

run "bundle check"

echo_recipe "command bundle check", "end"
```

代码 1: Rails 程序模板文件 `bundle_check_001.tmp1.rb`

代码 1 和代码 2，从运行结果完全而言是完全等价的，所不同的是仅仅系统终端显示内容不同而已。

```
run "bundle check"
```

代码 2: Rails 程序模板文件 `bundle_check_001` 完全等价代码

2.1.2 程序模板文件运行命令

在使用命令 1 时，文件 `bundle_check_001.tpl.rb` 可以按照本文说明创建产生，也可以使用命令 2 中链接下载到本地。

```
rails new bundle_check_001 -m ./bundle_check_001.tpl.rb
```

命令 1: 本地 *Rails* 程序模板文件 `bundle_check_001.tpl.rb` 命令

```
rails new bundle_check_001 -m http://www.lycom.de/files/rails-templates/bundle\_check\_001.tpl.rb
```

命令 2: 远程 *Rails* 程序模板文件 `bundle_check_001.tpl.rb` 命令

```
rm -rf ./bundle_check_001
```

命令 3: 删除已创建的 *Rails* 应用程序 `bundle_check_001` (可选命令)

命令名称

与参数“new”一起使用的由用户命名的参数

与参数“-m”一起使用的由用户命名的参数

```
rails new bundle_check_001 -m ./bundle_check_001.tpl.txt
```

命令固定参数

Rails应用程序名称

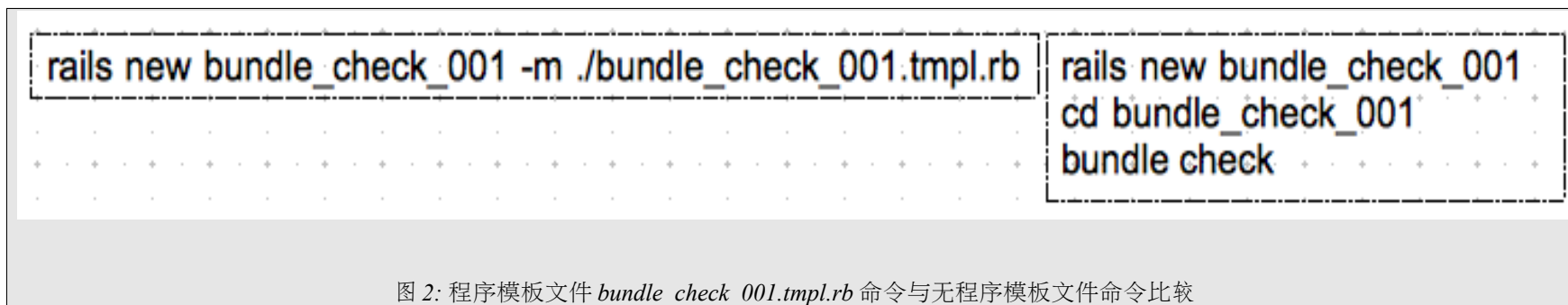
命令固定参数

Rails应用程序模板文件及其路径

图 1: 命令“rails new”详细说明

2.1.3 程序模板文件命令与无程序模板文件命令比较

下面**左边**是使用程序模板文件的命令，**右边**是没有使用程序模板文件的命令。



2.1.4 程序模板文件命令运行正确结果

如图 3 所示是利用如命令 1 所示的“本地”模板文件命令”所得到的。其结果告诉我们，应用程序文件 Gemfile 所需要的软件包相互依赖性是正确的。

```
create vendor/plugins/.gitkeep
apply /Users/gudao/dev/rails3/bundle_check_001.templ.rb
echo begin Running command 'bundle check' recipe...
run bundle check from "."
The Gemfile's dependencies are satisfied
echo end Running command 'bundle check' recipe...
You have new mail in /var/mail/gudao
localhost:rails3 gudao$
```

图 3: 本地程序模板文件 *bundle_check_001.templ.rb* 运行部分结果

如图 4 所示是利用如命令 2 所示的“远程”模板文件命令”所得到的。

```
create vendor/plugins/.gitkeep
apply http://www.lycom.de/files/rails-templates/bundle_check_001.templ.rb
echo begin Running command 'bundle check' recipe...
run bundle check from "."
The Gemfile's dependencies are satisfied
echo end Running command 'bundle check' recipe...
localhost:rails3 gudao$
```

图 4: 远程程序模板文件 *bundle_check_001.templ.rb* 运行部分结果

2.2 命令 `bundle install`

下面将要介绍的命令“`bundle install`”与命令“`bundle check`”有什么相同与相异之处呢？

软件包 Bundler 的命令 `bundle install` 的 **含义** 是什么？它将要确保在你的文件 `Gemfile` 中所有依赖软件包，对于你的应用程序是可用的。其原文是： „Make sure all dependencies in your Gemfile are available to your application. “。

这里作进一步 **解释**：首先，它检查了文件 `Gemfile` 的所有软件包的依赖性；其次，把文件 `Gemfile` 的所有软件包安装到语言运行环境下；最后，所有软件包是针对应用程序本身的检测。

2.2.1 程序模板文件代码

```
# bundle_install_002.tmp1.rb
def echo_recipe(name, tag)
  say "\033[36m" + "echo #{tag}".rjust(10) + "\033[0m" + "  Running #{name} recipe..."
end

echo_recipe "command bundle install", "begin"
run "bundle install"
echo_recipe "command bundle install", "end"
```

代码 3: Rails 程序模板文件 `bundle_install_002.tmp1.rb`

2.2.2 程序模板文件运行命令

```
rails new bundle_install_002 -m ./bundle_install_002.templ.rb
```

命令 4: 本地 *Rails* 程序模板文件 *bundle_install_002.templ.rb* 命令

```
rails new bundle_install_002 -m http://www.lycom.de/files/rails-templates/bundle\_install\_002.templ.rb
```

命令 5: 远程 *Rails* 程序模板文件 *bundle_install_002.templ.rb* 命令

```
rm -rf ./bundle_install_002
```

命令 6: 删除已创建的 *Rails* 应用程序 *bundle_install_002* (可选命令)

2.2.3 程序模板文件命令和无程序模板文件命令比较

下面**左边**是使用程序模板文件的命令，**右边**是没有使用程序模板文件的命令。

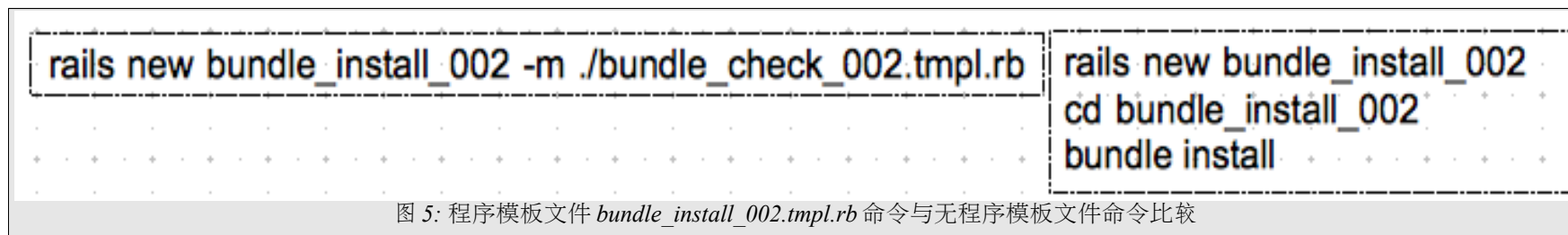


图 5: 程序模板文件 `bundle_install_002.templ.rb` 命令与无程序模板文件命令比较

2.2.4 程序模板文件运行命令正确结果

如图 6 所示是利用如命令 4 所示的“本地模板文件命令”所得到的。

```
    create vendor/plugins/.gitkeep
    apply /Users/gudao/dev/rails3/bundle_install_002.tpl.rb
  echo begin Running command 'bundle install' recipe...
    run bundle install from "."
Fetching source index for http://rubygems.org/
Using rake (0.8.7)
Using abstract (1.0.0)
Using activesupport (3.0.3)
Using builder (2.1.2)
Using i18n (0.5.0)
Using activemodel (3.0.3)
Using erubis (2.6.6)
Using rack (1.2.1)
Using rack-mount (0.6.13)
Using rack-test (0.5.7)
Using tzinfo (0.3.24)
Using actionpack (3.0.3)
Using mime-types (1.16)
Using polyglot (0.3.1)
Using treetop (1.4.9)
Using mail (2.2.14)
Using actionmailer (3.0.3)
Using arel (2.0.7)
Using activerecord (3.0.3)
Using activerecord (3.0.3)
Using bundler (1.0.7)
Using thor (0.14.6)
Using railties (3.0.3)
Using rails (3.0.3)
Using sqlite3 (1.3.3)
Using sqlite3-ruby (1.3.3)
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled gem is installed.
  echo end Running command 'bundle install' recipe...
localhost:rails3 gudao$
```

图 6: 本地 Rails 程序模板文件 `bundle_install_002.tpl.rb` 运行部分结果

如图 7 所示是利用如命令 5 所示的“远程模板文件命令”所得到的。

```
create vendor/plugins/.gitkeep
apply http://www.lycom.de/files/rails-templates/bundle_install_002.tpl.rb
echo begin Running command 'bundle install' recipe...
run bundle install from "."
Fetching source index for http://rubygems.org/
Using rake (0.8.7)
Using abstract (1.0.0)
Using activesupport (3.0.3)
Using builder (2.1.2)
Using il8n (0.5.0)
Using activemodel (3.0.3)
Using erubis (2.6.6)
Using rack (1.2.1)
Using rack-mount (0.6.13)
Using rack-test (0.5.7)
Using tzinfo (0.3.24)
Using actionpack (3.0.3)
Using mime-types (1.16)
Using polyglot (0.3.1)
Using treetop (1.4.9)
Using mail (2.2.14)
Using actionmailer (3.0.3)
Using arel (2.0.7)
Using activerecord (3.0.3)
Using activeresource (3.0.3)
Using bundler (1.0.7)
Using thor (0.14.6)
Using railties (3.0.3)
Using rails (3.0.3)
Using sqlite3 (1.3.3)
Using sqlite3-ruby (1.3.3)
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled gem is installed.
echo end Running command 'bundle install' recipe...
localhost:rails3 gudao$
```

图 7: 远程 Rails 程序模板文件 `bundle_install_002.tpl.rb` 运行部分结果

这个结果 **告诉** 我们，应用程序文件 **Gemfile** 所需要的软件包相互依赖性是正确的，并且应用程序所需要的所有软件包也已经正确地安装了。

就执行结果而言，这里所执行的命令与命令“`rails new bundle_install_002`”是完全一样的。那 **为什么** 我们又需要把程序模板文件与应用程序创建命令一起使用呢？

这里我们 **假定** 这样的一个问题，在创建了我们上面的应用程序之前，我们知道，想要增加一个新的 **gem** 软件包。我们该如何做呢？换言之，要是仅仅使用一条含有 **Rails** 模板文件的应用程序创建命令，是否也能够完成这个任务呢？这是我们

进一步地讨论 **Rails** 程序模板文件的 **作用**。

2.3 深入命令 `bundle install`

在创建一个新的应用程序时，我们事先知道，为新建的应用程序需要默认文件 `Gemfile` 所没有的软件包。对于这样的情况，就要创建一个合适的程序模板文件，以满足命令“`rails new`”的需要。

2.3.1 程序模板文件代码

```
# bundle_install_003.tpl.rb
def echo_recipe(name, tag)
  say "\033[36m" + "echo #{tag} ".rjust(10) + "\033[0m" + "  Running #{name} recipe..."
end

echo_recipe "use append_file and 'bundle install'", "begin"
append_file 'Gemfile', %{\ gem 'sunspot_rails' }.strip
run "bundle install"
echo_recipe "use append_file and 'bundle install'", "end"
```

代码 4: Rails 程序模板文件 `bundle_install_003.tpl.rb`

2.3.2 程序模板文件运行命令

```
rails new bundle_install_003 -m ./bundle_install_003.templ.rb
```

命令 7: 本地 *Rails* 程序模板文件 *bundle_install_003.templ.rb* 命令

```
rails new bundle_install_003 -m http://www.lycom.de/files/rails-templates/bundle\_install\_003.templ.rb
```

命令 8: 远程 *Rails* 程序模板文件 *bundle_install_003.templ.rb* 命令

```
rm -rf ./bundle_install_003
```

命令 9: 删除已创建的 *Rails* 应用程序 *bundle_install_003* (可选命令)

2.3.3 运行程序模板文件命令正确结果

如图 8 所示是利用如命令 7 所示的“本地模板文件命令”所得到的。

```
create vendor/plugins/.gitkeep
apply /Users/gudao/dev/rails3/bundle_install_003.tmpl.rb
echo begin Running use append_file and 'bundle install' recipe...
append Gemfile
run bundle install from "."
Fetching source index for http://rubygems.org/
Using rake (0.8.7)
Using abstract (1.0.0)
Using activesupport (3.0.3)
Using builder (2.1.2)
Using i18n (0.5.0)
Using activemodel (3.0.3)
Using erubis (2.6.6)
Using rack (1.2.1)
Using rack-mount (0.6.13)
Using rack-test (0.5.7)
Using tzinfo (0.3.24)
Using actionpack (3.0.3)
Using mime-types (1.16)
Using polyglot (0.3.1)
Using treetop (1.4.9)
Using mail (2.2.14)
Using actionmailer (3.0.3)
Using arel (2.0.7)
Using activerecord (3.0.3)
Using activerecord (3.0.3)
Using activerecord (3.0.3)
Using bundler (1.0.7)
Using escape (0.0.4)
Using nokogiri (1.4.4)
Using pr_geohash (1.0.0)
Using thor (0.14.6)
Using railties (3.0.3)
Using rails (3.0.3)
Using rsolr (0.12.1)
Using sqlite3 (1.3.3)
Using sqlite3-ruby (1.3.3)
Using sunspot (1.2.1)
Using sunspot_rails (1.2.1)
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled gem is installed.
echo end Running use append_file and 'bundle install' recipe...
localhost:rails3 gudao$
```

图 8: 本地 Rails 程序模板文件 `bundle_install_003.tmpl.rb` 运行部分结果

如图 9 所示是利用如命令 8 所示的“远程模板文件命令”所得到的。

```
    create vendor/plugins/.gitkeep
    apply http://www.lycom.de/files/rails-templates/bundle_install_003.tmpl.rb
  echo begin Running use append_file and 'bundle install' recipe...
    append Gemfile
    run bundle install from "."
Fetching source index for http://rubygems.org/
Using rake (0.8.7)
Using abstract (1.0.0)
Using activesupport (3.0.3)
Using builder (2.1.2)
Using il8n (0.5.0)
Using activemodel (3.0.3)
Using erubis (2.6.6)
Using rack (1.2.1)
Using rack-mount (0.6.13)
Using rack-test (0.5.7)
Using tzinfo (0.3.24)
Using actionpack (3.0.3)
Using mime-types (1.16)
Using polyglot (0.3.1)
Using treetop (1.4.9)
Using mail (2.2.14)
Using actionmailer (3.0.3)
Using arel (2.0.7)
Using activerecord (3.0.3)
Using activerecord (3.0.3)
Using bundler (1.0.7)
Using escape (0.0.4)
Using nokogiri (1.4.4)
Using pr_geohash (1.0.0)
Using thor (0.14.6)
Using railties (3.0.3)
Using rails (3.0.3)
Using rsolr (0.12.1)
Using sqlite3 (1.3.3)
Using sqlite3-ruby (1.3.3)
Using sunspot (1.2.1)
Using sunspot_rails (1.2.1)
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled gem is installed.
  echo end Running use append_file and 'bundle install' recipe...
localhost:rails3 gudao$
```

图 9: 远程 Rails 程序模板文件 `bundle_install_003.tmpl.rb` 运行部分结果

2.3.4 程序模板文件命令和无程序模板文件命令比较

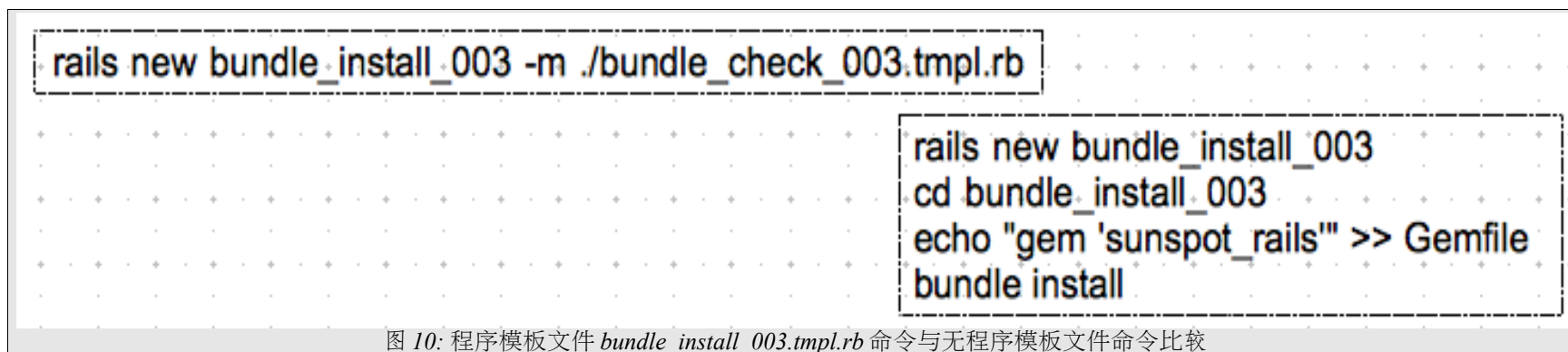


图 10: 程序模板文件 `bundle_install_003.tpl.rb` 命令与无程序模板文件命令比较

2.3.5 程序模板文件代码比较

比较

第二节与第三节的程序模板文件代码的区别，可以看到，仅仅后者增加了一个函数 `append_file`。

<pre>def echo_recipe(name, tag) say "\033[36m" + "echo #{tag}".rjust(10) + \ "\033[0m" + " Running #{name} recipe..." end echo_recipe "command 'bundle install'", "begin" run "bundle install" echo_recipe "command 'bundle install'", "end"</pre>	<pre>def echo_recipe(name, tag) say "\033[36m" + "echo #{tag}".rjust(10) + \ "\033[0m" + " Running #{name} recipe..." end echo_recipe "use append_file and 'bundle install'", "begin" append_file 'Gemfile', %{"gem 'sunspot_rails' }.strip run "bundle install" echo_recipe "use append_file and 'bundle install'", "end"</pre>
--	--

图 11: 程序模板文件代码比较

该函数的任务是：在文件 `Gemfile` 中的最后增加代码“`gem 'sunspot_rails'`”。如图 12 所示是这两个程序的 `Gemfile` 文件最终代码：

<pre># 应用程序bundle_install_002 source 'http://rubygems.org' gem 'rails', '3.0.3' gem 'sqlite3-ruby', :require => 'sqlite3'</pre>	<pre># 应用程序bundle_install_003 source 'http://rubygems.org' gem 'rails', '3.0.3' gem 'sqlite3-ruby', :require => 'sqlite3' gem 'sunspot_rails'</pre>
--	--

图 12: 两个 `Gemfile` 文件代码结果

3 结语 / 结束语

利用 Rails 程序模板文件一起**创建** Rails 应用程序，就会使得我们把需要解决的问题说明得更简单和更清晰一些，并且把创建应用程序的过程更程序化和更规范化一些。

有了 Rails 程序**模板**文件以后，所有创建应用程序的核心就转移到了该文件的代码。

Rails 程序模板文件的**代码**，不仅仅是让我们能够实现应用程序软件包的安装、更新和删除等千变万化的可能性，而且更重要的是，能够让我们可以简化和避免在命令行下重复输入执行这些命令。

在 Rails 程序模板文件代码中，可以看到软件包**命令代码**如上面提到的软件包 Bundler 的相关命令，也可以看到完成修改 Ruby 语言程序文件的**程序代码**。

在 rails 程序模板文件代码中，可以让我们定义**嵌入**与 Rails 应用程序相关的不同框架，如代码测试框架和持久层框架等。网站 <http://railswizard.org/> 已经为这一实现给我们提供了可能。

必须指出的是，每一个 Rails 程序模板文件是为了一个**目标**而创建的。但是，我们可以从每一个程序模板文件代码中获取适合于我们所需要的代码。

4 附录

A:

5 参考文献附录

- <http://tomafro.net/2010/02/updated-rails-template-for-bundler>
- http://gembundler.com/bundle_install.html
- <https://github.com/carlhuda/bundler>
- <http://railswizard.org/>