1. Start a new application

1.1 Git

1.1.1 Introduction

Git is a distributed VCS (Version Control System). If you are not using an VCS in your company, stop reading right now and go to install one. I would recommand Git, a comparison of VCS tools or a detailed documentation about Git are out of the scope of this book. If you are already working with a different VCS (SVN, Mercurial or any other tool) and you are not willing to give Git a try that's perfectly fine, you can skip the current and the next section. If you are not using a VCS yet, this can be a great opportunity to start working with Git.

If you didn't yet, you can install Git from here. For detailed informations on how to install Git on different platforms, check the online documentation.

If you didn't worked with github before, you may need to learn how to set up an SSH key

If this is the first time you will use Git on your development environment, don't forget to setup you identity.

```
$ git config --global user.name "Your name"
$ git config --global user.email "your_email@example.com"
```

You can also setup aliases for Git, which are simply shorcuts to commands you use often. Here is the list of my aliases that I found very useful:

```
$ git config --global alias.co checkout
$ git config --global alias.br branch
$ git config --global alias.ci commit
$ git config --global alias.st status
$ git config --global alias.poh "push origin HEAD"
$ git config --global alias.pom "pull origin master"
$ git config --global alias.dif "diff HEAD..origin/master"
$ git config --global alias.last 'log -1 HEAD'
```

Don't worry if you don't understand any of those commands, we will discover them later in this book.

1.1.2 Your first repository

Go to your work directory and type in:

```
$ mkdir symfony-tutorial
$ cd symfony-tutorial
$ git init
Initialized empty Git repository in /home/projects/symfony-tutorial/.git/
$ touch composer.json
$ git add composer.json
$ git commit -m "initial commit"
```

We just created a new repository with one empty file, and made our first commit. We will see later what's the purpose of the *composer.json* file. Git keeps track of all your actions, to see your repository history, type

```
$ git log
commit 10343840c09043b26df51ef9806091a8dd0c646b
Author: ilyes <ilyes@kooli.me>
Date:   Sat May 9 16:40:58 2015 +0300

    initial commit
```

Git generates a *unique* hash for each commit (10343840c09043b26df51ef9806091a8dd0c646b), so most probably you will see a different hash in your repository.

> Git generates an SHA-1 hash computed from your repository metadata, your username and the current timestamp. This data combined together guarantees there is a very small probability of colliding with other commits

Git also keeps track about the author, the commit date and the commit message. Talking about commit messages, **initial commit** is a valid message, however is not very informative. Git gives you the possibility to rewrite commit messages even after you commited them, let's change our commit message to something more significant.

```
$ git commit --amend -m "added composer.json"
[master fa7887a] added composer.json
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 composer.json
```

As it's name implies, *composer.json* will contain json data, let's add an empty json object to it. Edit *composer.json* with your favorite text editor and add `{}` to it.
Let's inspect the status of our repository.

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   composer.json

no changes added to commit (use "git add" and/or "git commit -a")
```

> If you have git colors output enabled, you should see**composer.json** colored in red.
> To enable git colors output, run `git config --global color.ui true`

Git tells us that we have one modified file that is not *staged* for commit. You can edit as many files as you want, only *staged* files will be taken into consideration when committing. Let's stage the file we just edited using the command `git add composer.json`. Git doesn't show any feedback when the command runs successfully but will show an error message in case of failure.

Let's run `git status` again and see what happens.

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   composer.json
```

The modified file is now ready to be committed. If you have git colors output enabled, you should see **composer.json** colored in green. Let's commit our latest changes.

```
 $ git commit -m "added empty object to composer.json"
[master 2426add] added empty object to composer.json
 1 file changed, 1 insertion(+)
```

Done, we successfully commited two changes to our repository.

1.1.3 Repository history

Let's inspect our repository history one more time.

```
 $ git log
commit 2426add179a4c9a0df6a0bd43e28216da44e22c9
Author: ilyes <ilyes@kooli.me>
Date:   Sat May 9 18:01:33 2015 +0300

    added empty object to composer.json

commit fa7887ace14536a24684390df5d6915d38019be4
Author: ilyes <ilyes@kooli.me>
Date:   Sat May 9 16:40:58 2015 +0300

    added composer.json
```

In addition to viewing a summary of your repository history, you can get more detailed informations about a specific commit. Let's view the details of our last commit using `git show` command.

```
$ git show 2426add179a4c9a0df6a0bd43e28216da44e22c9
commit 2426add179a4c9a0df6a0bd43e28216da44e22c9
Author: ilyes <ilyes@kooli.me>
Date:   Sat May 9 18:01:33 2015 +0300

    added empty object to composer.json

diff --git a/composer.json b/composer.json
index e69de29..9e26dfe 100644
--- a/composer.json
+++ b/composer.json
@@ -0,0 +1 @@
+{}
\ No newline at end of file
```

> Don't forget to change `2426add179a4c9a0df6a0bd43e28216da44e22c9` to the hash displayed in your repository history.

Beside the commit hash, the author, the date, and the commit message, Git also shows a *diff* between the specified commit and the commit right before it. The `+{}` indicates the change we made in that commit.

1.1.4 Rewrite the history

When working on a new feature or a bug fix, you may add many commits to your repository. Try to write an informative commit message that tells what and why the change occured. However, when checking your repository history, you may notice that two or more commits could be grouped in one commit because they share the same logic. Let's see how to do it.

First, let's add a line break to composer.json, so it's content looks like

```
{

}
```

Let's commit this change with the message "add line break in composer.json". Running `git log` will show something like

```
$ git log
commit effeaa33d1334ccae894bd308ef41062096d4f77
Author: ilyes <ilyes@kooli.me>
Date:   Sun May 10 12:54:49 2015 +0300

    add line break in composer.json

commit 2eef0a49c0eae24304407b470499aff9ecc08e9d
Author: ilyes <ilyes@kooli.me>
Date:   Sat May 9 18:42:01 2015 +0300

    added empty object to composer.json

commit fa7887ace14536a24684390df5d6915d38019be4
Author: ilyes <ilyes@kooli.me>
Date:   Sat May 9 16:40:58 2015 +0300

    added composer.json
```

The last two commits are a perfect candidate to be merged in a single commit, let's do it.

```
git rebase -i 2eef0a49c0eae24304407b470499aff9ecc08e9d^
#Git will launch your configured text editor with the content:
pick 2eef0a4 added empty object to composer.json
pick effeaa3 add line break in composer.json

# Rebase fa7887a..effeaa3 onto fa7887a
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#  x, exec = run command (the rest of the line) using shell
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
```

Edit the second line and change **pick** to **squash**, save the file and exit.

Git will launch your text editor again with content:

```
# This is a combination of 2 commits.
# The first commit's message is:

added empty object to composer.json

# This is the 2nd commit message:

add line break in composer.json

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# rebase in progress; onto fa7887a
# You are currently editing a commit while rebasing
# branch 'master' on 'fa7887a'.
#
# Changes to be committed:
#       modified:   composer.json
#
```

Just save the file and exit. Git will inform you that the rebase was successful.

```
[detached HEAD a7bd6eb] added empty object to composer.json
 1 file changed, 3 insertions(+)
Successfully rebased and updated refs/heads/master.
```

1.2 Composer

1.2.1 Instalation

Composer is a dependency management tool for PHP applications. To install Composer, simply run:

```
$ curl -sS https://getcomposer.org/installer|php -- \
--install-dir=/bin --filename=composer
#!/usr/bin/env php
All settings correct for using Composer
Downloading...

Composer successfully installed to: /home/skafandri/bin/composer
Use it: php /home/skafandri/bin/composer
```

You can check the composer version by typing

```
$ composer --version
Composer version 1.0-dev
(4eb8ecff9cd136c4c2829164db1a55ad9d2de616) 2015-05-10 19:56:28
```

For more details about other installation alternatives, check the composer online documenation.

1.2.2 Create an application

Remove everything form you project folder (or create another one)

```
$ rm -rf .git
$ unlink  cmoposer.json
```

Now proceed with installation

```
$ composer create-project symfony/framework-standard-edition "2.6.7"
Installing symfony/framework-standard-edition (v2.6.7)
 - Installing symfony/framework-standard-edition (v2.6.7)
   Downloading: 100%

Created project in symfony-tutorial
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
 - Installing doctrine/lexer (v1.0.1)
   Loading from cache

 ......

 - Installing sensio/generator-bundle (v2.5.3)
   Loading from cache

instalation suggests...

Generating autoload files
```

Then you will be prompted for missing parameters

```
Would you like to install Acme demo bundle? [y/N] N
Creating the "app/config/parameters.yml" file
Some parameters are missing. Please provide them.
database_driver (pdo_mysql):
database_host (127.0.0.1):
database_port (null):
database_name (symfony):
database_user (root):
database_password (null):
mailer_transport (smtp):
mailer_host (127.0.0.1):
mailer_user (null):
mailer_password (null):
locale (en):
secret (ThisTokenIsNotSoSecretChangeIt):
Clearing the cache for the dev environment with debug true
Trying to install assets as symbolic links.
...
The assets were installed using symbolic links.
```

Our application is ready to run, you can run the PHP built in web server with the command

```
$ app/console server:run
Server running on http://127.0.0.1:8000

Quit the server with CONTROL-C.
```

Now you should be able to access the application at the address http://127.0.0.1:8000

If you get the exception **NotFoundHttpException: No route found for "GET /"** congratulations, your application is up and running.

Composer generated the directory structure of your Symfony application.

- **app** contains the application loader and global configuration files
- **src** contains the source code of the application, now should contain a sample AppBundle
- **vendor** contains code of external packages required by the application
- **web** is the web directory that your web server should use as DocumentRoot

Composer generated a **composer.json** file containing the application dependencies and some other configurations. We will install two additional packages, *doctrine/migrations* and *doctrine/mongodb-odm-bundle*.

Edit composer.json and add "doctrine/migrations": "1.0.*@dev" and "doctrine/doctrine-migrations-bundle": "1.0.*" to the end of the **require** section, your composer.json should look like

```
"require": {
      "php": ">=5.3.3",
      "symfony/symfony": "2.6.7",
      "doctrine/orm": "~2.2,>=2.2.3,<2.5",
      "doctrine/dbal": "<2.5",
      "doctrine/doctrine-bundle": "~1.2",
      "twig/extensions": "~1.0",
      "symfony/assetic-bundle": "~2.3",
      "symfony/swiftmailer-bundle": "~2.3",
      "symfony/monolog-bundle": "~2.4",
      "sensio/distribution-bundle": "~3.0,>=3.0.12",
      "sensio/framework-extra-bundle": "~3.0,>=3.0.2",
      "incenteev/composer-parameter-handler": "~2.0",
      "doctrine/migrations": "1.0.*@dev",
      "doctrine/doctrine-migrations-bundle": "1.0.*"
  },
```

Now we will tell composer to *update* the application with the new requirements

```
$ composer update
Loading composer repositories with package information
Updating dependencies (including require-dev)
- Installing doctrine/migrations (dev-master a4f14d3)
  Cloning a4f14d3a3d397104e557ec65d1a4e43bb86e4ddf

- Installing doctrine/doctrine-migrations-bundle (1.0.0)
  Downloading: 100%

  Writing lock file
Generating autoload files
Updating the "app/config/parameters.yml" file
Clearing the cache for the dev environment with debug true
Trying to install assets as symbolic links.
...
The assets were installed using symbolic links.
```

> Note: You may get some other packages updated, depending on the requirement of the newly added packages.

1.3 Configuration

By default, Symfony uses many configuration files for different purposes. `app/config/routing.yml` contains the routing informations of your application. Your file should look lie

```
app:
    resource: "@AppBundle/Controller/"
    type:     annotation
```

This tells Symfony to look for routing configuration within the functions *annotation* blocks in any class in the folder `srs/AppBundle/Controller`. Under `srs/AppBundle/Controller` we have only one file with one class **DefaultController**

```
<?php

namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class DefaultController extends Controller
{
    /**
     * @Route("/app/example", name="homepage")
     */
    public function indexAction()
    {
        return $this->render('default/index.html.twig');
    }
}
```

`@Route("/app/example", name="homepage")` tells Symfony to define a route with the name **homepage** and the path /**app/example**, we will cover the controller content and the routing options later. This configuration is set using an *annotation*. There are 4 ways to set configuration for your

Symfony application: annotations, yaml, xml and PHP.

Let's change this route definition from annotationto yaml.

1.Remove the annotation block and the first use statement from the DefaultController so it looks like

```
<?php

namespace AppBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class DefaultController extends Controller
{

    public function indexAction()
    {
        return $this->render('default/index.html.twig');
    }

}
```

2.Edit app/config/routing.yml

```
app:
    resource: "@AppBundle/Resources/config/routing.yml"
```

3.Add a new file src/AppBundle/Resources/config/routing.yml

```
app_home:
    path:      /
    defaults:  { _controller: AppBundle:Default:index}
```

Done, if you visit http://127.0.0.1:8000/ you should see a blank page with one word "Homepage."

1.4 Profiler

Did you noticed that toolbar in the bottom? That's called the Symfony debug toolbar. It shows some informations as the Symfony and PHP versions, the route and the controller rendering the current request, processing time and memory usage.. You can click on the third link that

looks like **5e780b app dev** to access the Symfony profiler. There you can find a detailed profile about the last requests. Feel free to explore the profiler sections. By default the profiler is enabled only in *dev* mode. We will cover Symfony modes in details later.

1.5 Homework

1. Initialize a new git repository in your project folder, create a new branch with the name **ch1**, commit all the files to that branch and **push** the changes to a **remote** repository that you create on Github

   > You will need `git branch` and `git remote` commands

2. Add "doctrine/mongodb-odm" and "doctrine/mongodb-odm-bundle" packages to your application

   > You may need to install a missing PHP extention

3. Update your Symfony version to **2.7**

4. Create a new configuration file **app/config/doctrine.yml** and move the "Doctrine Configuration" section from **app/config/config.yml** into the new created file. Exclude the new file from Git.