1

Symfony example - Lab Sheet 1

This lab sheet ensures:

1. You have all software setup for the module on your computer
2. You have run and used the kinds of web applications you'll be creating in the module

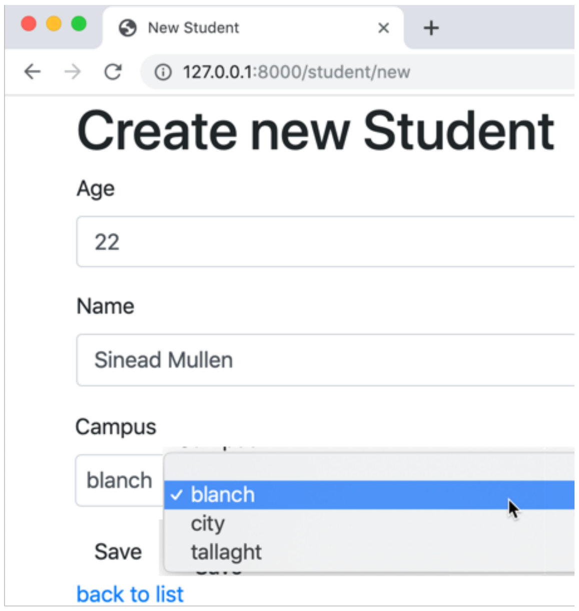


Figure 1.1: Screenshot showing new Student form with Campus choice dropdown menu

1.1 Preparation

1.2 Ensure PHP is installed on your computer

You need PHP version 7.2.5 or later. **NOTE: At the time of writing, there is an issue with PHP 7.4, so if installing PHP, installed the latest 7.3 version but avoid 7.3 for now ...***

Open a command line terminal (e.g. the `cmd` application in Windows) and check your PHP version at the command line with:

```
$ php -v
PHP 7.3.1 (cli) (built: May  9 2018 19:49:10)
Copyright (c) 1997-2017 The PHP Group
Zend Engine v3.1.0, Copyright (c) 1998-2017 Zend Technologies
```

If your version is older than 7.2.5, or you get an error about command not understood, then complete the steps in Appendix B.

1.3 Ensure the `Composer` PHP command line tool is installed on your computer

Type `composer` in a command line terminal. You should seem something like this:

```
$ composer

   ------
  / ____/___  ____ ___  ____  ____  ________  _____
 / /   / __ \/ __ `__ \/ __ \/ __ \/ ___/ _ \/ ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__  )  __/ /
\____/\____/_/ /_/ /_/ .___/\____/____/\___/_/
                    /_/
Composer version 1.7.2 2018-08-16 16:57:12
// more command summary lines here ....
```

However, if you get an error saying no such application, then install Composer from:

- https://getcomposer.org/doc/00-intro.md#installation-windows

See Appendix C.

1.4 Ensure the Git version control utilities are installed

(Git is required for the Symfony command line tool) Run `$ git` at the command line. You should see somegthing like this:

```
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one
```

If not, then visit https://git-scm.com/download/win and run the installer. Then close and open a new terminal window and check Git is working.

1.5 Ensure the `symfony` command line tool is installed on your computer

Having the Symfony command line tool will make things easier (less typing!). Check it at the command line with:

```
$ symfony
Symfony CLI version v4.12.4 (c) 2017-2020 Symfony SAS
Symfony CLI helps developers manage projects, from local code to remote infrastructure

These are common commands .... // more lines here
```

If you get a suggestion to **update** your version of the Symfony command line tool, say **YES**!

If you get an error saying no such application, then install Symfony from:

`- [https://symfony.com/download](https://symfony.com/download)`

See Appendix C.

1.6 Ensure the MySQL and SQLite PHP database extensions are enabled (needed for Windows)

If PHP, Composer and Symfony are working, you have PHP setup on your computer sufficient for this module. However, the MySQL and SQLite database extensions may not be setup.

You can either work ahead, hoping it is setup, and fix it if you hit a problem when trying to create a database. Or you can check, and fix it now.

PHP extensions are already installed with PHP, but may not be activated. All we have to do is ensure there is no semi-colon character `;` at the beginning of lines `extension=php_pdo_mysql.dll` and `extension=php_pdo_sqlite.dll`.

See Appendix B for steps to enable these extensions.

1.7 Open the PHPStorm code editor

On Blanchardstown campus college computers you can open up PHPStorm from the `Jetbrains` folder on the **Desktop**. Run the `phpstorm64.exe` application.

You will need to setup/re-activate your free education Jetbrains licence:

1. Get your free one-year subscription to Jetbrains products using your **TUDublin** university email address

    - `https://www.jetbrains.com/shop/eform/students`

On your own laptop/computer you can download and install the PHPStorm editor for free with your Jetbrains account.

1.8 Download project template and open in a code editor

1. Start your IDE editor (e.g. Notepad++ or PHPStorm)
2. Download from Moodle the ZIP project `crud01.zip`, and unzip to the Desktop.
3. Open a terminal window (either a Terminal application like `cmd`, or open a Terminal window inside your IDE)
4. In the terminal `cd` into folder `crud1`

1.9 Run the Symfony web sever

Let's run the web server on our machine (`localhost:8000`) by entering terminal command:

```
$ symfony serve
```

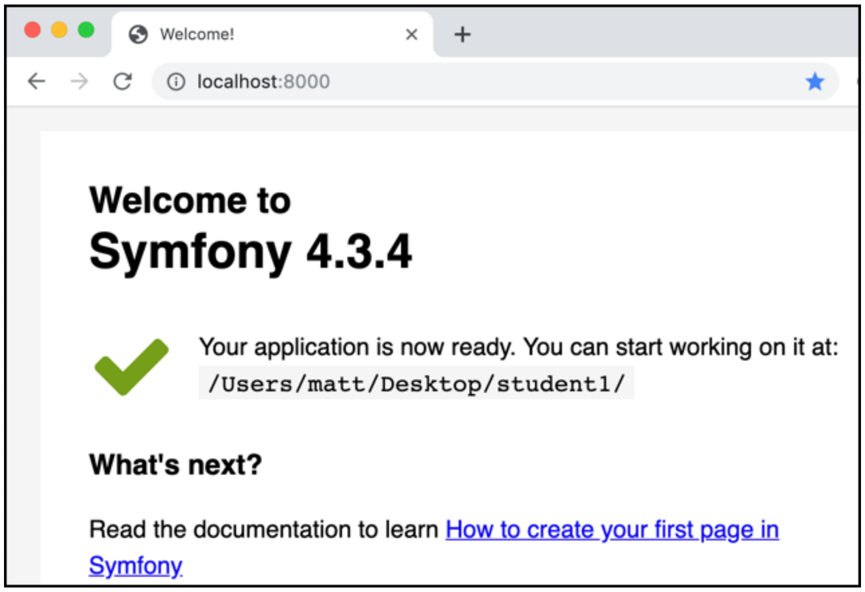
Note, you might get some warnings/info messages about version of PHP etc. - just ignore them!

```
$ symfony serve
Sep 27 07:22:19 |DEBUG| PHP    Using PHP version 7.3.8 (from default version in $PATH)
Sep 27 07:22:19 |INFO | PHP    listening path="/usr/local/php5-7.3.8-20190811-205217/sbin/php-fpm"
Sep 27 07:22:19 |DEBUG| PHP    started
Sep 27 07:22:19 |INFO | PHP    ready to handle connections

 [OK] Web server listening on http://127.0.0.1:8000 (PHP FPM 7.3.8)
```

1.10 Visit the home page `localhost:8000`

Open a web browser and visit our website home page at `http://localhost:8000`.

Since we didn't create a home page, we'll see a default Symfony home page. See Figure 1.2 shows a screenshot of PHPStorm and our new class PHP code.



Welcome to
Symfony 4.3.4

Your application is now ready. You can start working on it at:
/Users/matt/Desktop/student1/

What's next?

Read the documentation to learn How to create your first page in Symfony

Figure 1.2: Default Symfony home page.

2

Connect to and create our MySQL database

We need to set things up so our PHP web applicaton can communicate with MySQL and also setup our example database. Do the following:

1. Start up MySQL Workbench (`root` password is `Pass$$` on college computers …)

    - you may need to 'Clear the Vault' before being able to run an instance of MySQL …

2. Due to a change in MySQL during 2019 we need to run a special command in MySQL to allow PHP programs to communicate with MySQL:

    - in an SQL window in MySQL workbench execute the following command

    ```
    alter user 'root'@'localhost' identified with mysql_native_password by 'Pass$$';
    ```

That's it - PHP should now be able to communicate with MySQL - let's find out …

Tell Symfony to create its database:

```
$ php bin/console doctrine:database:create

Created database 'crud01' for connection named default
```

If you see the `created database` message then things are going well.

Now do the following:

1. If there is a folder `src/Migrations` DELETE it (since we have a new database, we don't want any old migrations to mess it up)

2. Create a new migration:

```
php bin/console make:migration
```

3. Run the migration:

```
php bin/console doctrine:migrations:migrate

  - say `y` when asked
```

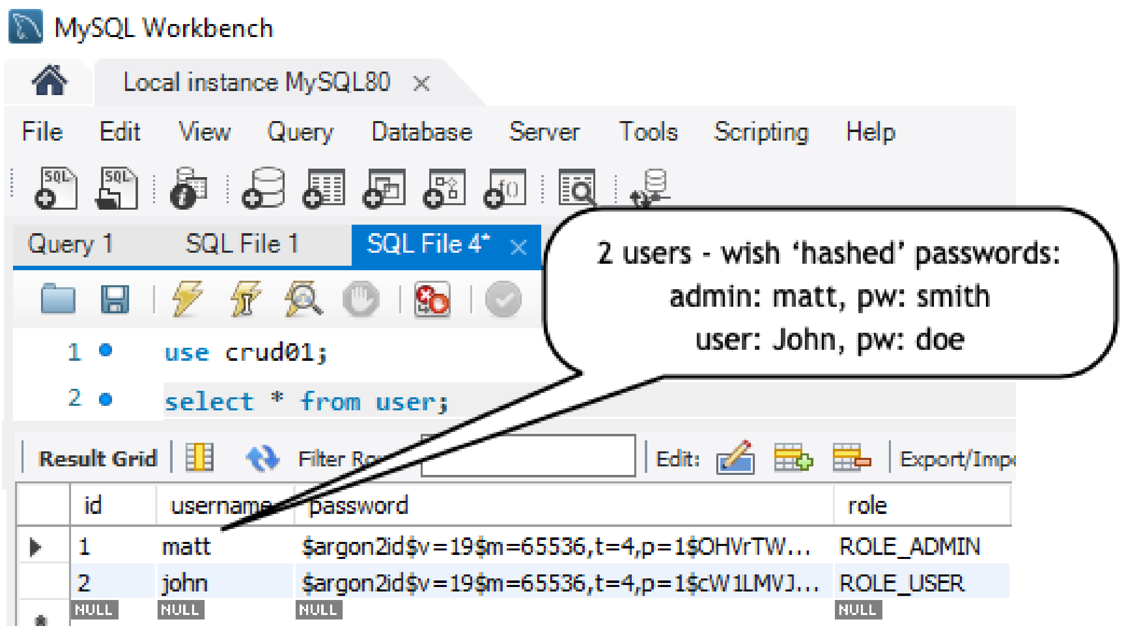
4. Load test data (fixtures)

```
php bin/console doctrine:fixtures:load

  - say `y` when asked
```

5. Now in MySQL workbench let's see what we've created, executre SQL:

```
use crud01;

select * from user;
```

See Figure 2.1 shows a screenshot of our database contents in the MySQL Workbench DB client.


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Query 1 SQL File 1 SQL File 4*
2 users - wish 'hashed' passwords:
admin: matt, pw: smith
user: John, pw: doe
Result Grid Filter Ro Edit: Export/Imp

```
use crud01;
select * from user;
```

id	username	password	role
1	matt	$argon2id$v=19$m=65536,t=4,p=1$OHVrTW...	ROLE_ADMIN
2	john	$argon2id$v=19$m=65536,t=4,p=1$cW1LMVJ...	ROLE_USER
NULL	NULL	NULL	NULL

Figure 2.1: User details in the database.

2.1 SOLING COMMON PROBLEMS: Error(s) when executing MIGRATIONS (table already exists etc.)

Each Migration is the incremental bit of SQL that needs to be executed to update the MySQL database structure to match our PHP Entity Classe.

Sometimes things will get out of synch - so that when we try to execute a migration with: doctrine:migrations:migrate, we get some errors about tables/properties already existing

The quickest and easiest way to get past this problem is to start again with a BRAND NEW EMPTY database, and NO MIGRATIONS - do this with the following 3 steps:

1. Change the database name in file `.env`
    - personally I just add 1 to the number of this database, e.g. change `crud01` to `crud02` and so on
2. Delete all historic migrations, just delete the whole folder `/src/Migrations`
3. Run your steps to create new db / create SQL for migrations / run SQL for migrations / load any fixture data:
    - create a new database (using the credentials in `.env`):

      ```
      php bin\console doctrine:database:create
      ```
    - write the SQL we need to create database to match the classes in `/src/Entity`:

      ```
      php bin\console make:migration
      ```
    - execute the SQL to create / alter the tables in the database:

      ```
      php bin\console doctrine:migrations:migrate
      ```
    - load any startup data defined in our **fixtures** classes:

      ```
      php bin\console doctrine:fixtures:load
      ```

3

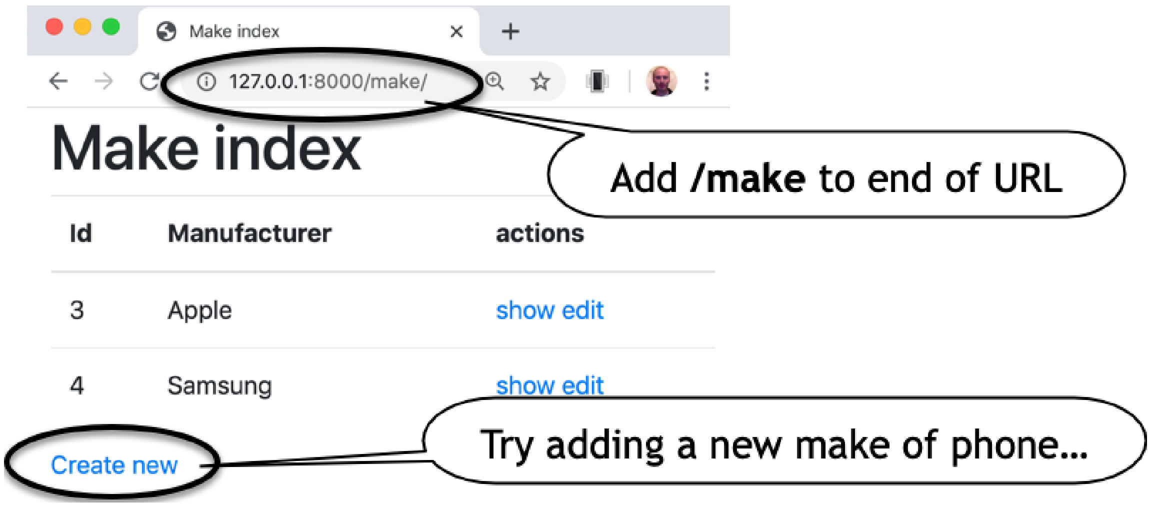
Explore existing phone/make CRUD pages

3.1 Add `make` to end of URL for phone make admin pages

Visit `localhost:8000/make` - the default for CRUD admin pages is to add the lower case name of the entity to the end of the URL.

`- try adding a new make or editing an existing one ...`

See Figure 3.1.



Make index

Id	Manufacturer	actions
3	Apple	show edit
4	Samsung	show edit

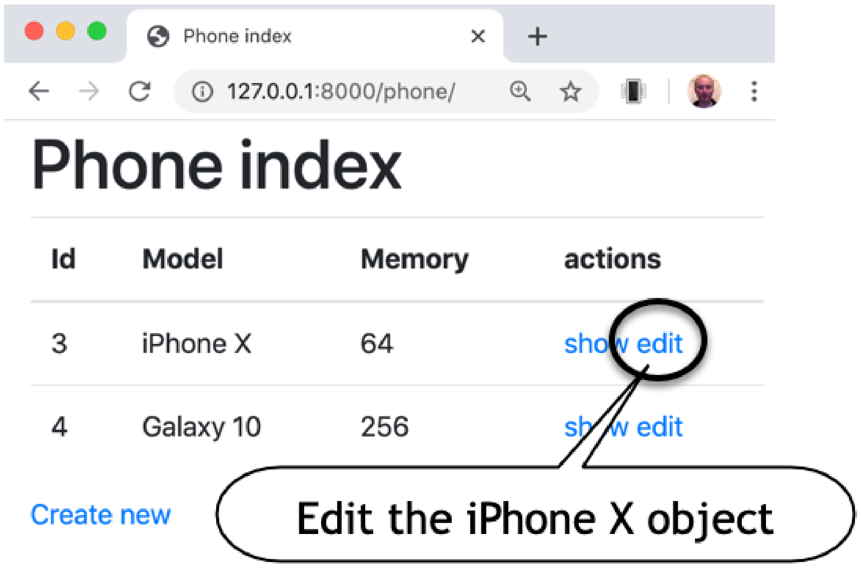
Figure 3.1: Screenshot of phone make CRUD pages.

3.2 Browse the Phone records

Visit `localhost:8000/phone` for the Phone object admin pages.

```
- try adding a new make or editing an existing one ...
```

See Figure 3.2.



Phone index

Id	Model	Memory	actions
3	iPhone X	64	show edit
4	Galaxy 10	256	show edit

Create new

Figure 3.2: Screenshot of phone model CRUD pages.

3.3 Many Phones to One Model

Edit a Phone record, and you'll see a list of the makes appear as a drop-down list to choose from. This is a **relationship** between each Phone object and a Phone Make object (many-to-one)

See Figure 3.3.

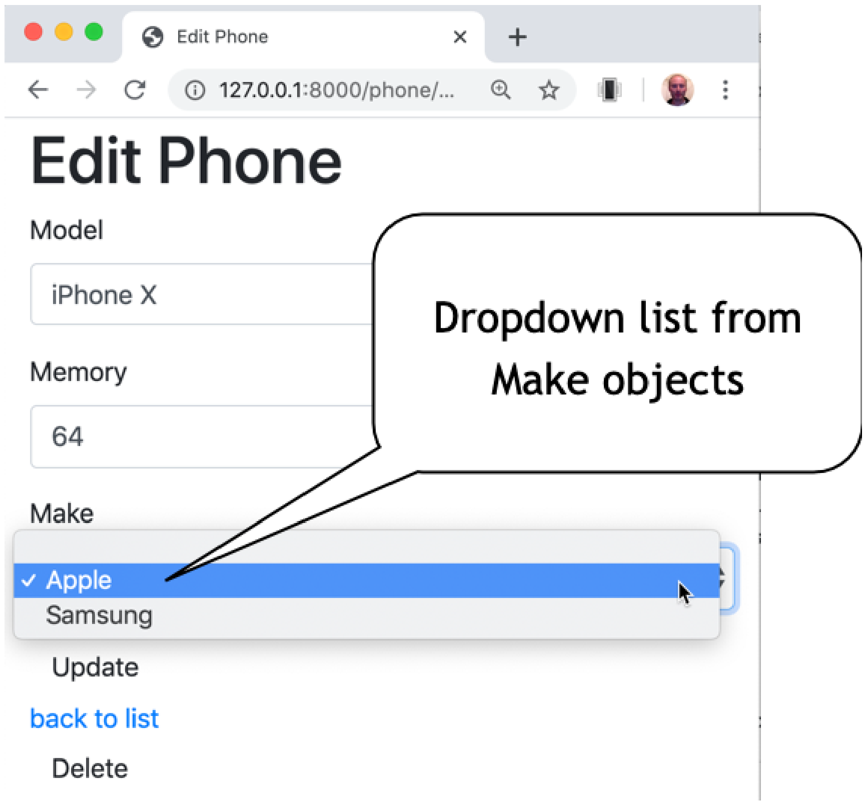


Figure 3.3: Editing phone - makes as dropdown menu.

4

Create your own CRUD for a Student entity class

4.1 Create Student class

Let's create a Student class and generate automatic CRUD web pages.

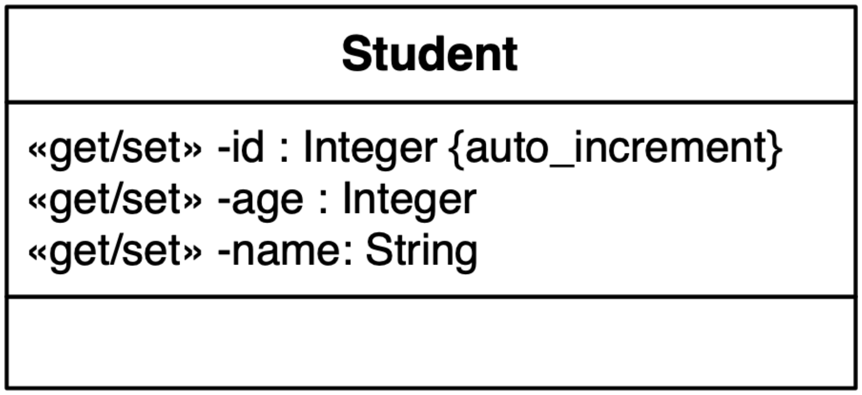


Figure 4.1: Class diagram for Student entity.

Do the following:

1. At the command line type:

   ```
   php bin/console make:entity Student
   ```

   You should see the following:

```
> php bin/console make:entity Student

created: src/Entity/Student.php
created: src/Repository/StudentRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command.
```

- notice that it tells us that is has created 2 new classes `src/Entity/Student.php` and `src/Repository/StudentRepository.php`
    - `Student.php` is a simple class with private propetries and getters/setters, with special ‘annotation' comments so these objects can map directly to rows in a database table…

2. Now we need to ask this console ‘make' tool to add an integer `age` property for us:
    - we need to enter the property name `age`
    - we need to specify its data type `integer`
    - (to keep things simple) we don't mind if our properies start as null (just press `<RETURN>` for this question)
    - you should see the following when answering these questions at the `make` command tool prompt:

    ```
    New property name (press <return> to stop adding fields):
    > age

    Field type (enter ? to see all types) [string]:
    > integer

    Can this field be null in the database (nullable) (yes/no) [no]:
    >

    updated: src/Entity/Student.php
    ```

3. Now add a string `name` property:
    - we need to enter the property name `name`
    - we need to specify its data type `string` (since default just press `<RETURN>`)
    - accept default string length of 255 (since default just press `<RETURN>`)
    - can be nullable (since default just press `<RETURN>`)

- you should see the following when answering these questions at the `make` command tool prompt:

```
Add another property? Enter the property name (or press <return> to stop adding fields):
> name

Field type (enter ? to see all types) [string]:
>

Field length [255]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Student.php
```

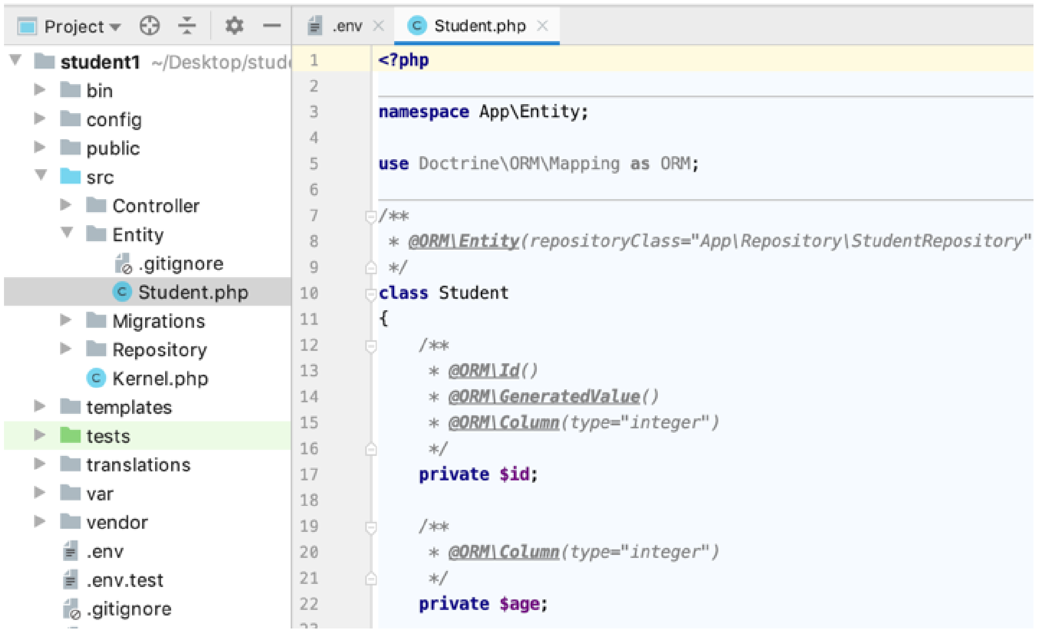
4. That's all our fields created, so just press `<RETURN>` to complete creation of our entity:

```
Add another property? Enter the property name (or press <return> to stop adding fields):
>

 Success!

Next: When you're ready, create a migration with make:migration
```

4.2 Take a look at the created entity class

Take a look at what's been created for us: `src/Entity/Student.php`. If you ignore the comments, mostly this is a class

See Figure 4.2 shows a screenshot of PHPStorm and our new class PHP code.



```php
<?php

namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass="App\Repository\StudentRepository")
 */
class Student
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="integer")
     */
    private $age;
```

Figure 4.2: New Student.php entity.

4.3 Make 'migration' SQL to create database to correspond to our entity class

We can also use the 'make' command line tool to look at our classes and create the SQL commands we need to update our database create/update tables for storing the object data in tables and rows.

Enter the following at the command line `php bin/console make:migration`:

```
> php bin/console make:migration

  Success!

 Next: Review the new migration "src/Migrations/Version20190927055812.php"
 Then: Run the migration with php bin/console doctrine:migrations:migrate
 See https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html
```

If you look inside the newly created file you'll see a line like this showing the SQL generated to create a database table to match our `Student.php` class:

```php
$this->addSql(
    'CREATE TABLE student (
        id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
        age INTEGER NOT NULL,
        name VARCHAR(255) NOT NULL
        )'
    );
```

4.4 Execute our 'migration' SQL to create/update database

Now let's tell Symfony to connect to the database and execute the migration SQL - to actually **create** the new `Student` table in the database.

We need to enter the terminal command `php bin/console doctrine:migrations:migrate`:

```
> php bin/console doctrine:migrations:migrate

    Application Migrations


WARNING! You are about to execute a database migration that could result in schema changes and data
Are you sure you wish to continue? (y/n)
```

At this point we must enter `y` to go ahead - saying we are happy for our database structure to be chagned by exectuing our migration SQL:

```
WARNING! You are about to execute a database migration that could result in schema changes and data
Are you sure you wish to continue? (y/n)y

Migrating up to 20190927055812 from 0

  ++ migrating 20190927055812
     -> CREATE TABLE student (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    age INTEGER NOT NULL, name VARCHAR(255) NOT NULL)

  ++ migrated (took 64.4ms, used 18M memory)
  ------------------------
  ++ finished in 70.8ms
  ++ used 18M memory
  ++ 1 migrations executed
  ++ 1 sql queries
```

That's it - we have now created a table in out database to match our PHP entity class.

4.5 Generate the CRUD web form for class `Student`

Let's generate some HTML and PHP code for a web form to list and create-read-update-delete data from our database.

We need to execute this command to create that code `php bin/console make:crud Student`:

```
> php bin/console make:crud Student

 created: src/Controller/StudentController.php
 created: src/Form/StudentType.php
 created: templates/student/_delete_form.html.twig
 created: templates/student/_form.html.twig
 created: templates/student/edit.html.twig
 created: templates/student/index.html.twig
 created: templates/student/new.html.twig
 created: templates/student/show.html.twig

  Success!

 Next: Check your new CRUD by going to /student/
```

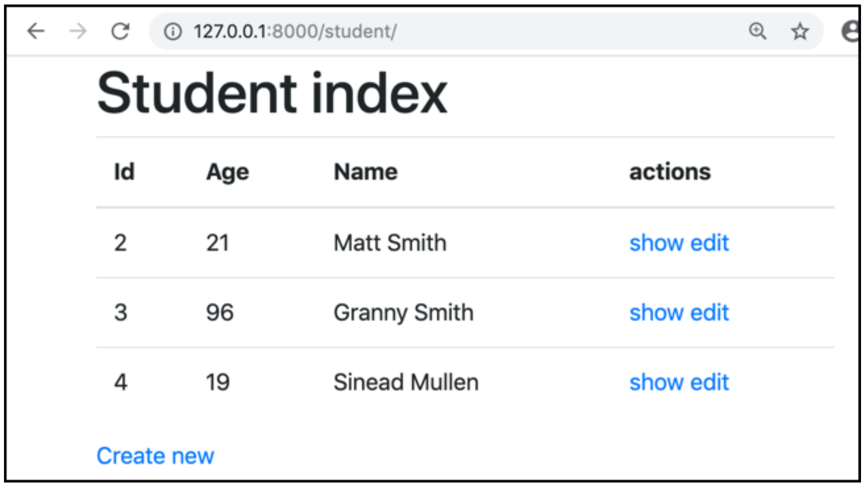
4.6 Visit our generated `Student` crud pages at `/student`

Let's visit our generated CRUD pages, these can be found by adding `/student` at the end of the URL.

Click `Create new` and add a student. Then try clicking edit, and change some values or delete it and create it again.

You should find you have a fully working web-based CRUD interface to your database.

See Figure 4.3 shows a screenshot of several students having been created (and yes, one of my grandmothers did live to 96!).



Student index

Id	Age	Name	actions
2	21	Matt Smith	show edit
3	96	Granny Smith	show edit
4	19	Sinead Mullen	show edit

Create new

Figure 4.3: Screenshot of nice lookling Bootstrap style admin CRUD pages.

4.7 Databases are persistent

Kill the Symfony web server at the command line by pressing `<CTRL>-C`. Then quit the PHPStorm IDE application.

You could also go have a cup of coffee, or perhaps shut down and restart your computer.

Then restart the PHPStorm editor, and restart the web server with:

```
symfony serve
```

Now open a web browser to URL `http://localhost:8000/student` and you should see that the students you created in your database are still there.

Any changes we make are remembered (persisted) as part of our database.

5

Adding a campus entity and relating them

5.1 Create a new class: Campus

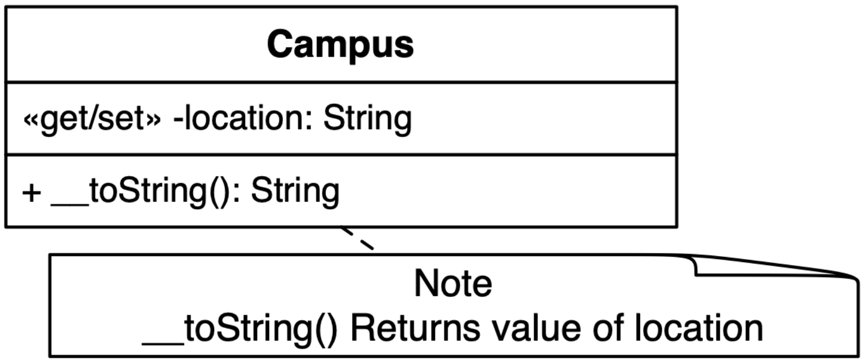

Campus
«get/set» -location: String
+ __toString(): String
Note
__toString() Returns value of location

Figure 5.1: Class diagram for Campus class.

Create entity Campus with single property 'location' (string) add a `__toString()` method to Campus class (`src/Entity/Campus.php`) containing the following

```
public function __toString()
{
    return $this->location;
}
```

(we'll need this toString method in Campus later, so that when creating/editing Students we can choose the related Campus object from a drop down menu - which needs a string description of each Campus).

HINT:

- use the interactive command line entity maker: `php bin/console make:entity Campus` …

5.2 Create CRUD for this Campus class

generate CRUD for Campus:

```
php bin\console make:crud Campus
```

5.3 Create relationship between Student and Campus (each student linked to one campus)

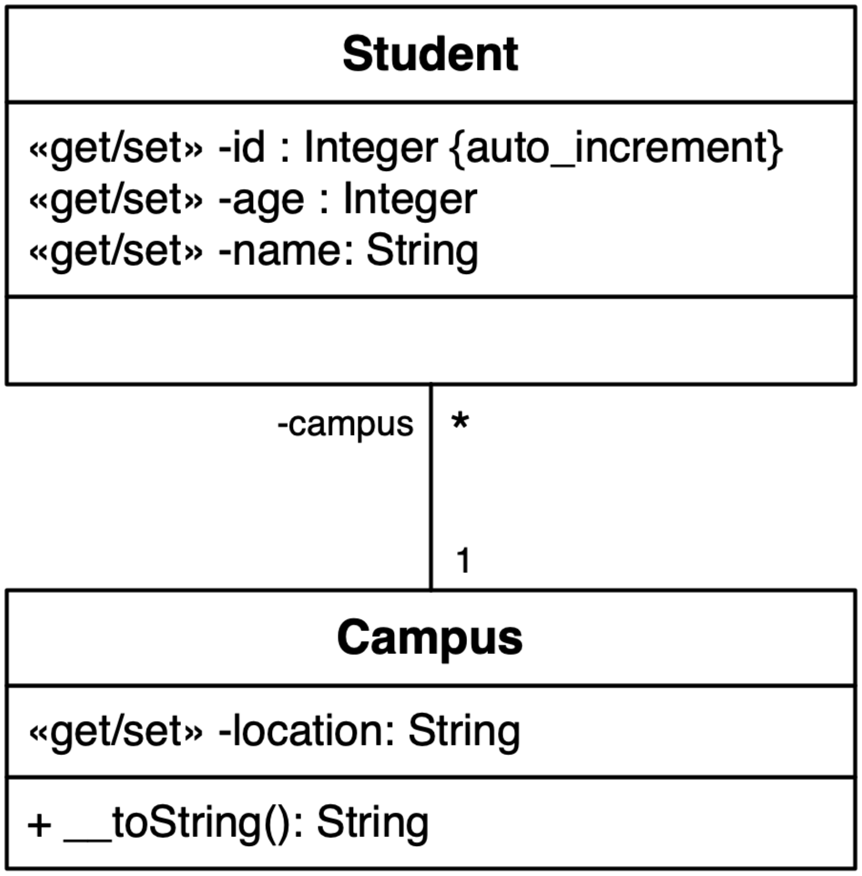


Figure 5.2: Class diagram for Student-Campus multiplicity.

To create this relationship we are going to add a **'campus'** property to the Student class, that is a reference to a Campus object. Here is our detailed new Student class diagram:

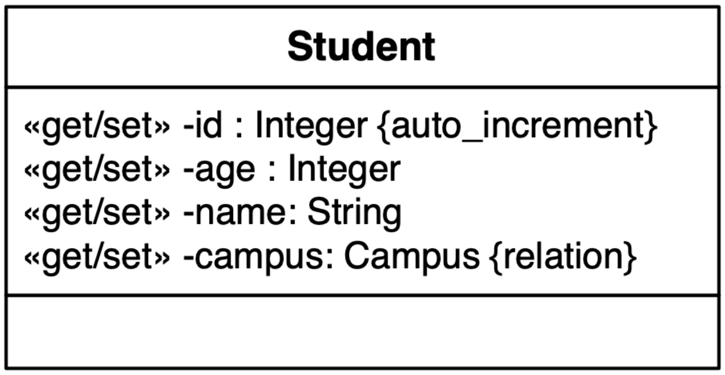

Student
«get/set» -id : Integer {auto_increment}
«get/set» -age : Integer
«get/set» -name: String
«get/set» -campus: Campus {relation}

Figure 5.3: Detailed Student class diagram.

Here is how to add a related property to a class:

add a property `campus` to the `Student` class, of type `relation`, that is `ManyToOne` to the Campus class i.e. many students linked to one campus to ADD a property to an existing class, we need to run the `make:entity` console command again:

```
php bin\console make:entity Student
```

the console should see the entity already exists, and invite us to add a new property…

NOTE:

- once you've given the property name, type, class, and `ManyToOne` relationship type, just keep hitting `<RETURN>` to accept the defaults
  - Symfony will also add a `students` array property to the `Campus` class for you - that's fine

NOTE:

- do **NOT** create **string** property type for `campus`
- the type for property `campus` should be **relation**
  - this means the SQL generated in teh migration will implement an SQL FOREIGN-KEY using the **id**s of Campus objects stored in a `campus_id` TABLE field for table `student`
  - look at the generated SQL when you make the **migration** after adding this relation property to campus
- if it's a string, then it won't link to a Campus object and things will not work later on …

5.4 Update Database Structure (since we changed our classes)

Create and run new DB migration

1. Create a migration by typing:

   ```
   php bin\console make:migration
   ```

2. now run the migration by typing:

   ```
   php bin\console doctrine:migrations:migrate
   ```

5.5 Delete old CRUD and generate new CRUD for both classes

NOTE: This is an **important step** - if you don't delete the old CRUD files, you won't be able to genera

1. delete the old Student CRUD
   - FILE: `src/Controller/StudentController.php`
   - FILE: `src/Form/StudentType.php`
   - folder: `templates/student`
2. generate CRUD for Student

```
php bin\console make:crud Student
```

5.6 Run server add some related records:

1. now run server:

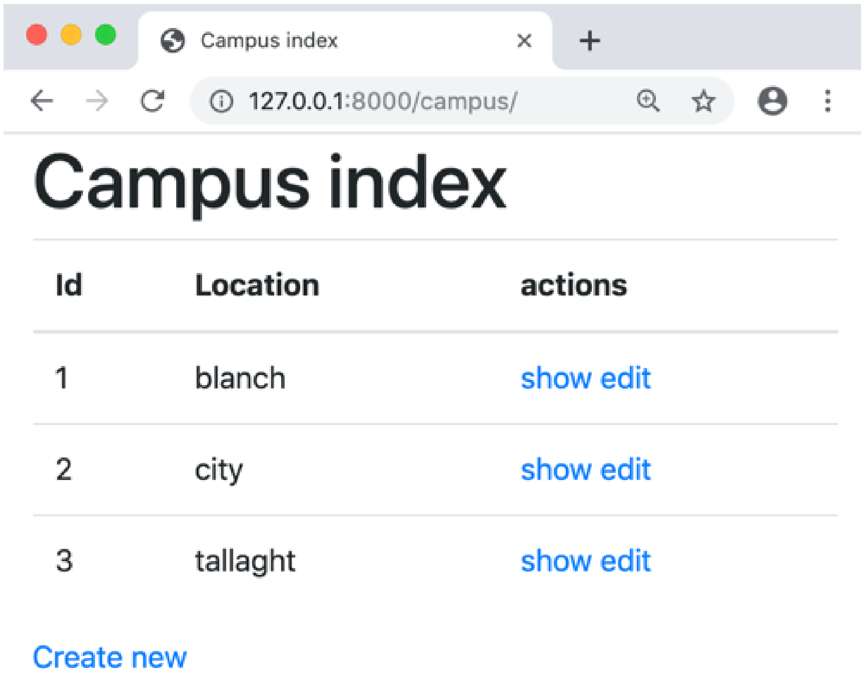
   ```
       symfony serve
   ```

2. visit the Campus CRUD pages and create record for 2 campuses

   - e.g. **Blanch** and **Tallaght**

3. visit the Student CRUD pages, and edit / create Student's related to your new Campuses

When we create/edit a student, we now get a dropdown menu of the Campus objects (the text in the dropdown menu is from the `__toString()` method we created for the Campus class).

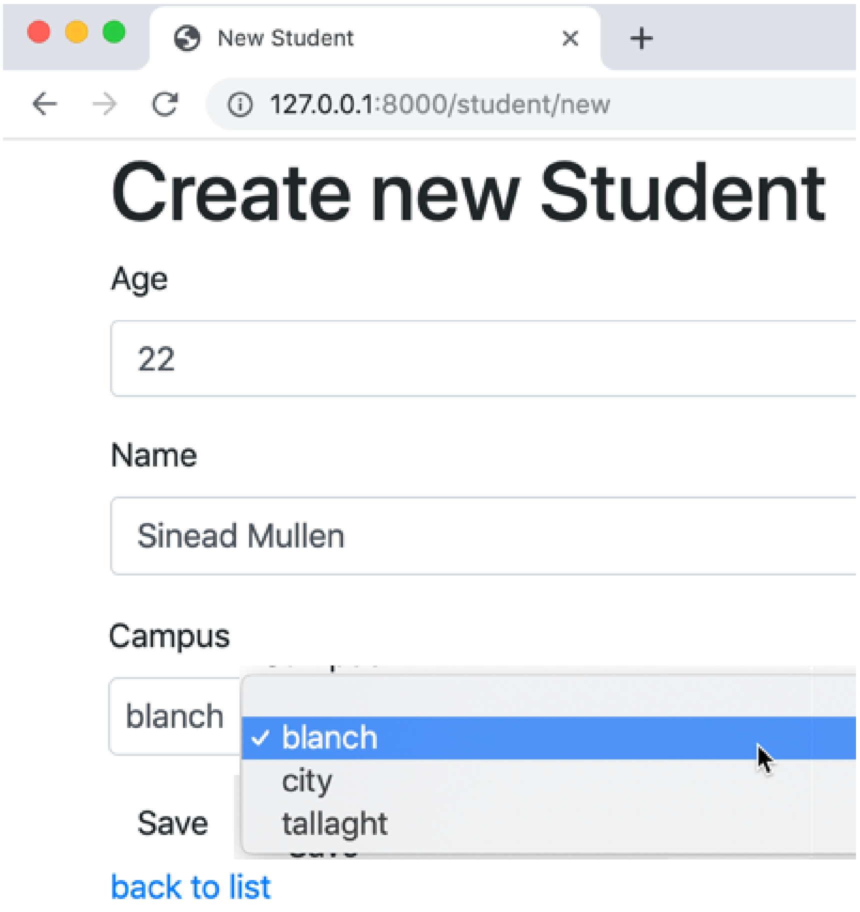


Campus index

Id	Location	actions
1	blanch	show edit
2	city	show edit
3	tallaght	show edit

Create new

Figure 5.4: Screenshot showing list of Campus objects.



New Student

127.0.0.1:8000/student/new

Create new Student

Age

22

Name

Sinead Mullen

Campus

blanch

Figure 5.5: Screenshot showing new Student form with Campus choice dropdown menu

6
Customising the Twig templates

6.1 Let's add a new Campus column to our list of students

We need to edit file: `/templates/student/index.html.twig`

See Figure 6.1 for the location of this file in the project folder structure.

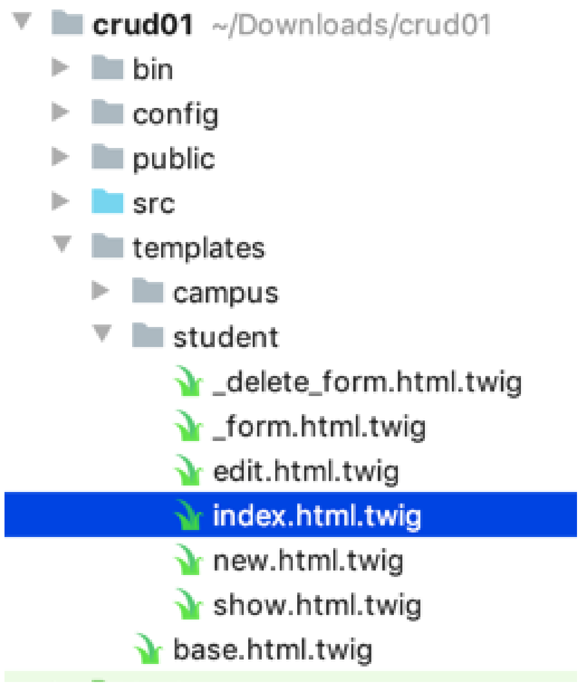
Figure 6.2 shows the 2 places where we need to edit.

Do the following:

1. Edit file: `/templates/student/index.html.twig`
2. Insert a new HTML table column header for `Campus`
3. Insert a new HTML table data item for `student.campus`

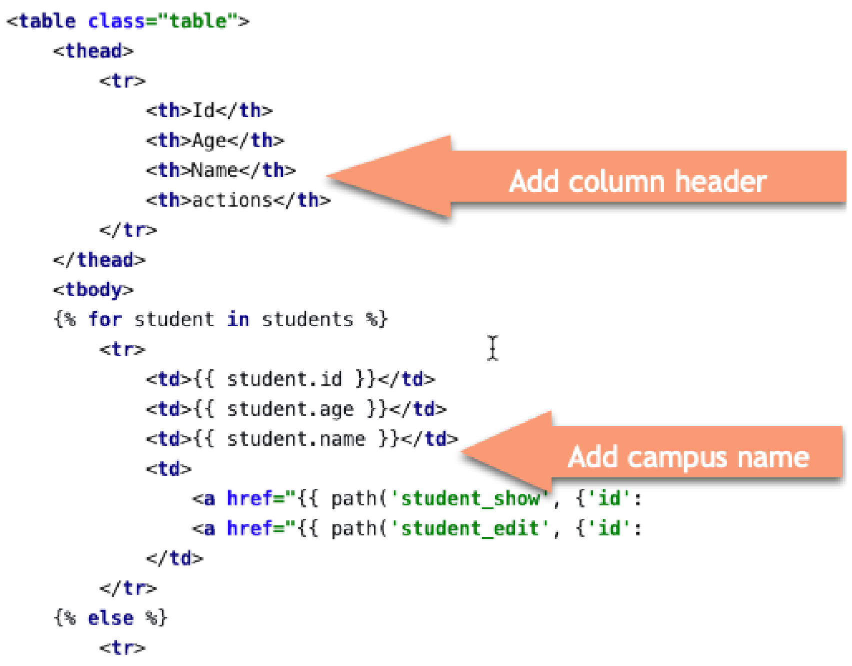
Figure 6.2 shows how the file should look now - after the lines have been inserted.

And when we visit `/student` we should see a campus column added to the student details - see Figure 6.4.



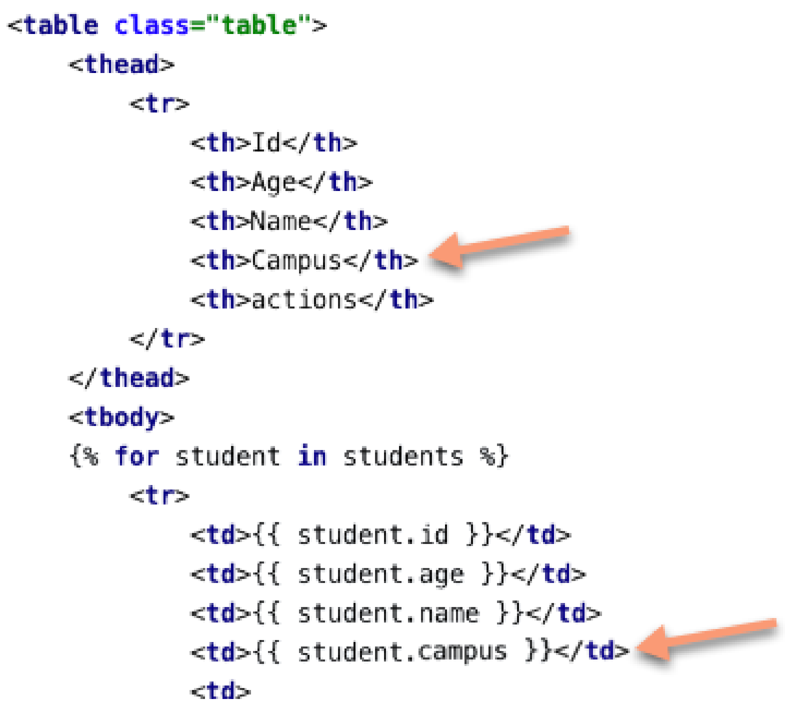
```
crud01 ~/Downloads/crud01
  bin
  config
  public
  src
  templates
    campus
    student
      _delete_form.html.twig
      _form.html.twig
      edit.html.twig
      index.html.twig
      new.html.twig
      show.html.twig
    base.html.twig
```

Figure 6.1: Location of Twig template files.



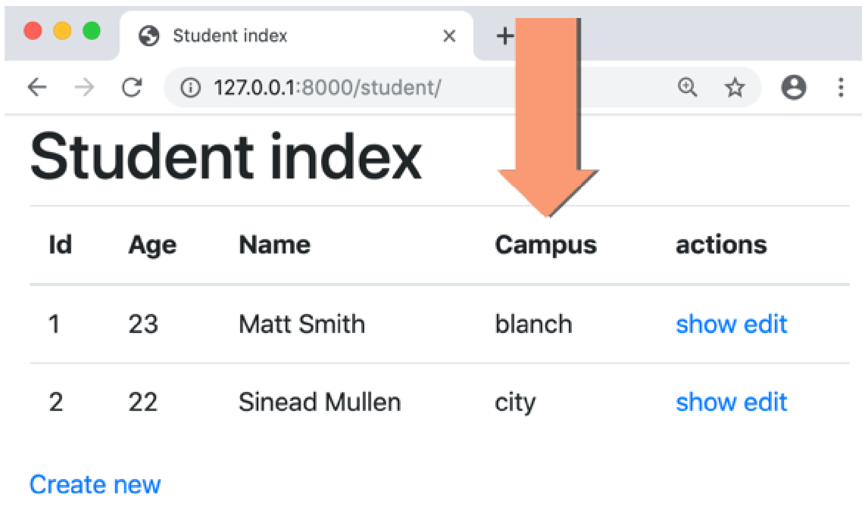
```
<table class="table">
    <thead>
        <tr>
            <th>Id</th>
            <th>Age</th>
            <th>Name</th>
            <th>actions</th>
        </tr>
    </thead>
    <tbody>
    {% for student in students %}
        <tr>
            <td>{{ student.id }}</td>
            <td>{{ student.age }}</td>
            <td>{{ student.name }}</td>
            <td>
                <a href="{{ path('student_show', {'id':
                <a href="{{ path('student_edit', {'id':
            </td>
        </tr>
    {% else %}
        <tr>
```

Figure 6.2: Where we will insert lines into template.



```
<table class="table">
    <thead>
        <tr>
            <th>Id</th>
            <th>Age</th>
            <th>Name</th>
            <th>Campus</th>
            <th>actions</th>
        </tr>
    </thead>
    <tbody>
    {% for student in students %}
        <tr>
            <td>{{ student.id }}</td>
            <td>{{ student.age }}</td>
            <td>{{ student.name }}</td>
            <td>{{ student.campus }}</td>
            <td>
```

Figure 6.3: Twig template after lines inserted.



Student index

Id	Age	Name	Campus	actions
1	23	Matt Smith	blanch	show edit
2	22	Sinead Mullen	city	show edit

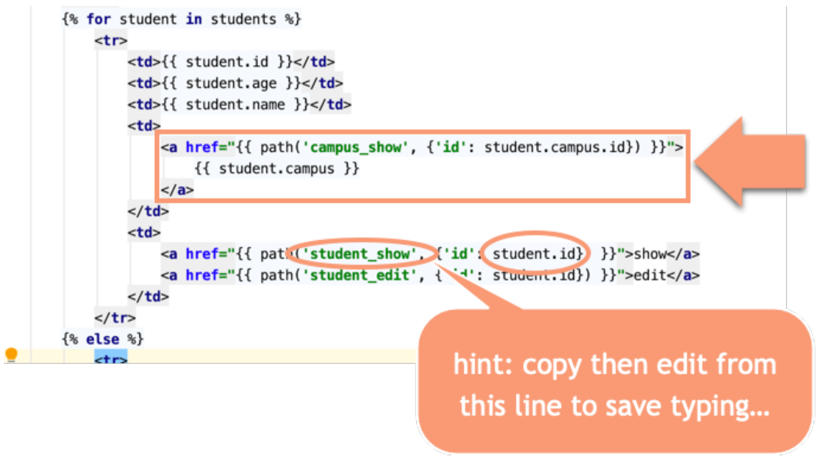
Create new

Figure 6.4: CRUD list page with extra column.

6.2 Turn the Campus name into a LINK to the related Campus object

We can wrap an HTML hyperlink element around the campus name, to connect our Student object to its related Campus object

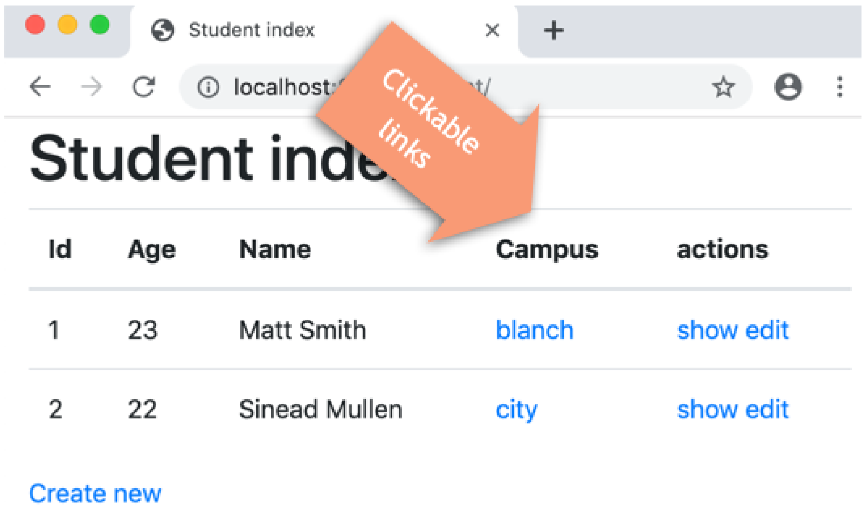
Here's the edit we need to add to file: `/templates/student/index.html.twig`



```
{% for student in students %}
    <tr>
        <td>{{ student.id }}</td>
        <td>{{ student.age }}</td>
        <td>{{ student.name }}</td>
        <td>
            <a href="{{ path('campus_show', {'id': student.campus.id}) }}">
                {{ student.campus }}
            </a>
        </td>
        <td>
            <a href="{{ path('student_show', {'id': student.id}) }}">show</a>
            <a href="{{ path('student_edit', {'id': student.id}) }}">edit</a>
        </td>
    </tr>
{% else %}
    <tr>
```

Figure 6.5: Where to edit to turn campus name into hyperlink.

When that's done, we can now click the campus and jump to the Campus objects 'show' page:



Student index

Id	Age	Name	Campus	actions
1	23	Matt Smith	blanch	show edit
2	22	Sinead Mullen	city	show edit

Create new

Figure 6.6: Web page where campus is clickable link to Campus object.

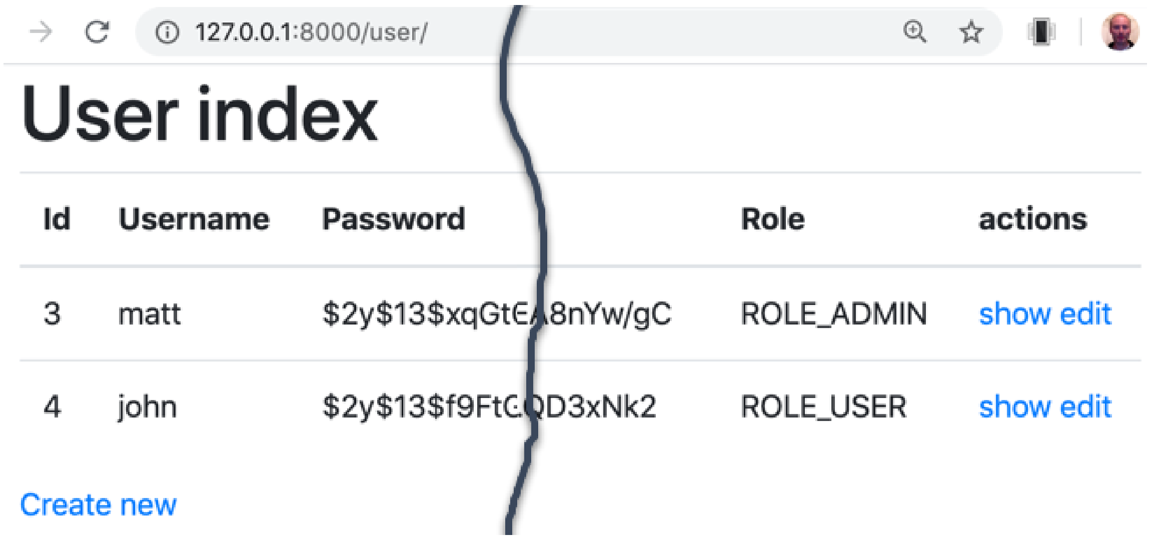
7

Add some login security

7.1 Visit the user admin pages

Visit `localhost:8000/user` - you'll see our 2 users for the system. See Figure 7.1.

```
- matt (password: smith), an **admin** user

- john (password: doe), a normal user
```



127.0.0.1:8000/user/

User index

Id	Username	Password	Role	actions
3	matt	$2y$13$xqGtEA8nYw/gC	ROLE_ADMIN	show edit
4	john	$2y$13$f9FtGQD3xNk2	ROLE_USER	show edit

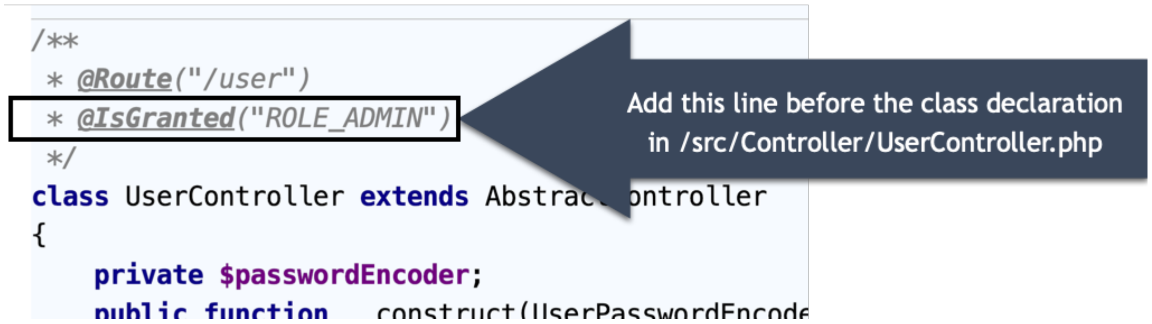
Create new

Figure 7.1: Screenshot of phone make CRUD pages.

7.2 Secure the user admin behind a firewall

Let's only allow logged in ROLE_ADMIN users to access the user CRUD pages.

We do this by adding a requirement that a user must be logged in and have the ROLE_ADMIN user role. This can all be achieved by adding a single line before the declaration of class file `src/Controller/UserController.php`:



```
/**
 * @Route("/user")
 * @IsGranted("ROLE_ADMIN")
 */
class UserController extends Abstrac  ontroller
{
    private $passwordEncoder;
    public function __construct(UserPasswordEncode
```

Add this line before the class declaration
in /src/Controller/UserController.php

Figure 7.2: IsGranted security requirement added to the `UserController` class.

7.3 Login

If you visit `localhost:8000/user` again you'll now be asked to login. See Figure 7.3.

If successfully logged in as an admnin user, you can now visit the user CRUD pages. See Figure 7.4.

Clicking the user in the debug profiler web page footer gives details about the role(s) of the logged in user. See Figure 7.5.

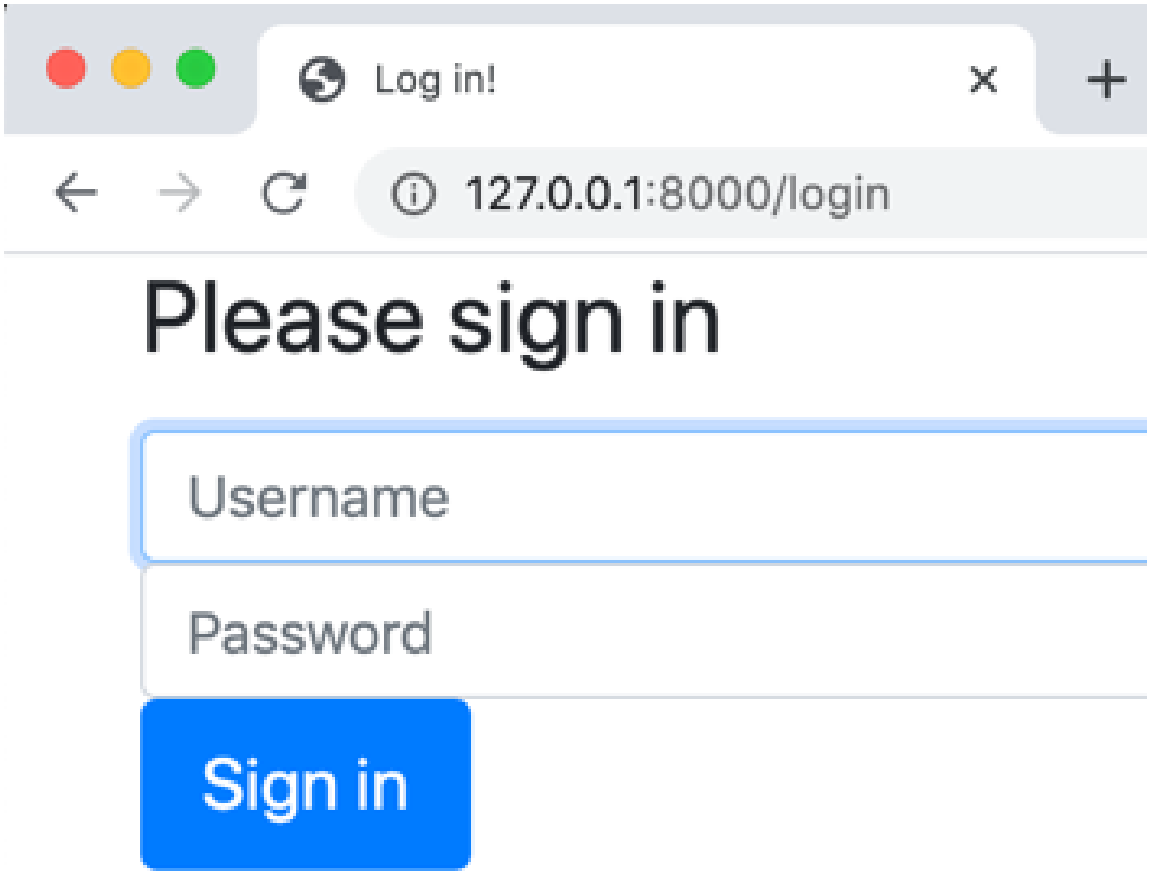
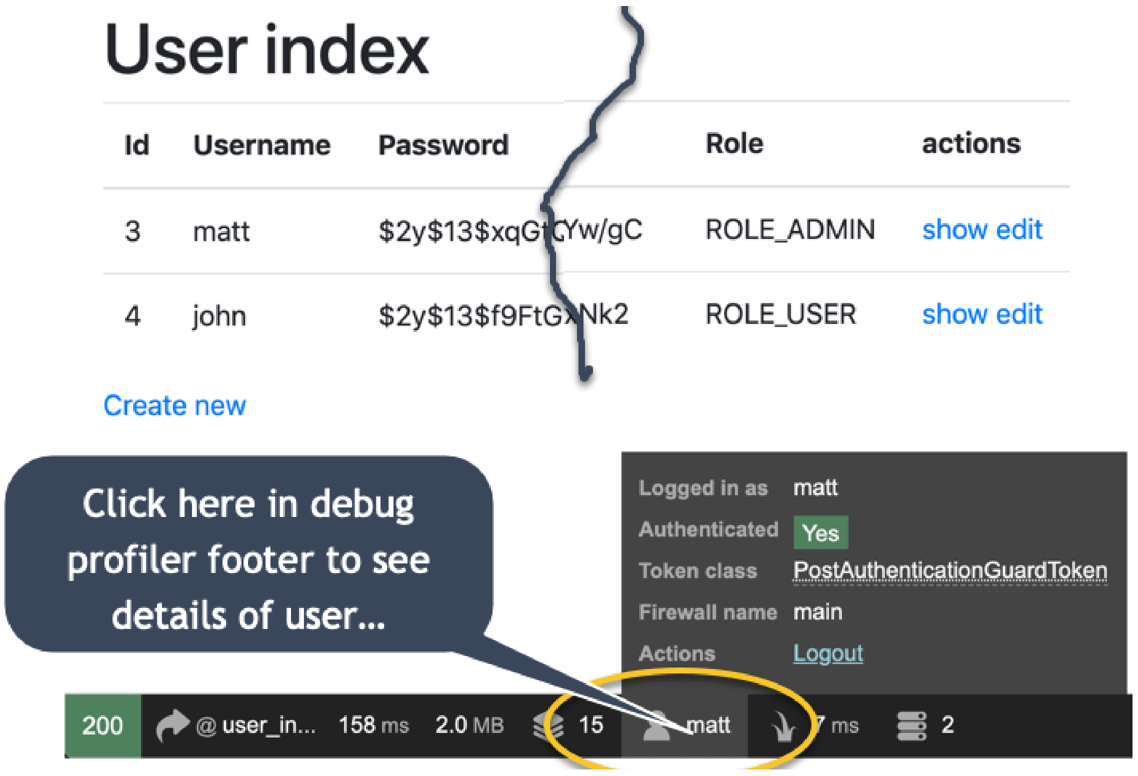


Figure 7.3: Login form.


User index

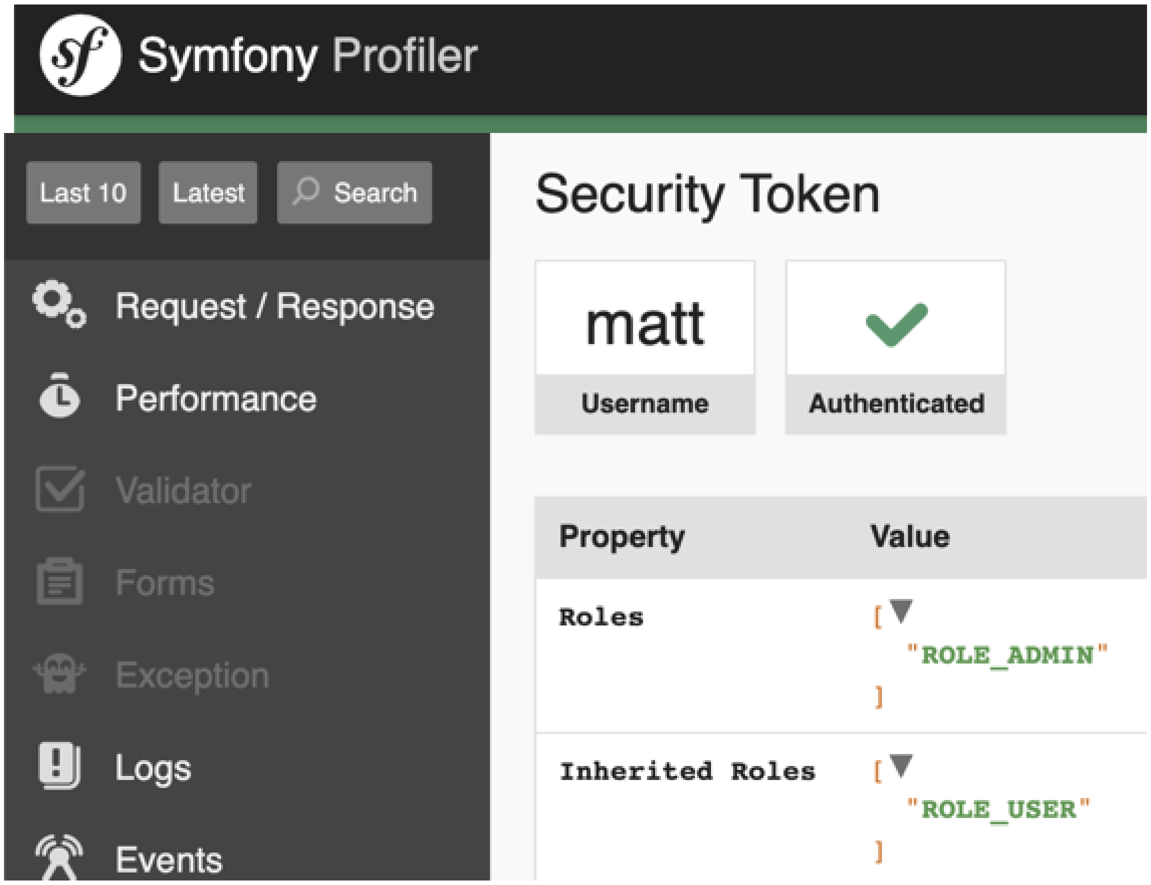
Id	Username	Password	Role	actions
3	matt	$2y$13$xqGtCYw/gC	ROLE_ADMIN	show edit
4	john	$2y$13$f9FtGxNk2	ROLE_USER	show edit

Create new

Logged in as matt
Authenticated Yes
Token class PostAuthenticationGuardToken
Firewall name main
Actions Logout

Click here in debug profiler footer to see details of user...
200
@ user_in...
158 ms
2.0 MB
15
matt
7 ms
2

Figure 7.4: User pages logged in as `matt`.



Security Token

matt
Username

✓
Authenticated

Property	Value
Roles	["ROLE_ADMIN"]
Inherited Roles	["ROLE_USER"]

Figure 7.5: Details of logged-ion user `matt`.

Part I

Appendices

A

Software required for Symfony development

A.1 Don't confuse different software tools

Please do not confuse the following:

- Git and Github
- PHP and PHPStorm

Here is a short description of each:

- Git: A version control system - can run locally or on networked computer. There are several website that support Git projects, including:
  - Github (perhaps the most well known)
  - Gitlab
  - Bitbucket
  - you can also create and run your own Git web server …
- Github: A commercial (but free for students!) cloud service for storing and working with projects using the Git version control system
- PHP: A computer programming language, maintained by an international Open Source community and published at `php.net`
- PHPStorm: A great (and free for student!) IDE - Interactive Development Enviroment. I.e. a really clever text editor created just for working with PHP projects. PHPStorm is one of the professional software tools offered by the **Jetbrains** company.

So in summary, Git and PHP are open source core software. Github and PHPStorm are commercial (but free for students!) tools that support development using Git and PHP.

A.2 Software tools

Ensure you have the following setup for developing Symfony software on your local machine

- PHP 7.2.5 or later (free, open source)
- Composer (up-to-date with `composer self-update`)(free, open source - a PHP program!)
- PHPStorm (with educational free account if you're a student!) - or some other editor of your choice
- MySQL Workbench (Community Edition free)
- Git (free, open source)

See Appendix B for checking, and if necessary, installing PHP on your computer. See Appendix A for details about other software needed for working with PHP projects.

A.3 Test software by creating a new Symfony 4 project

Test your software by using PHP and Composer to create a new Symfony 4 project. We'll follow the steps at the Symfony setup web page.

Follow the steps in Appendix **??**.

B

PHP Windows setup

B.1 Check if you have PHP installed and working

You need PHP version 7.2.5 or later.

Check your PHP version at the command line with:

```
> php -v
PHP 7.3.1 (cli) (built: May  9 2018 19:49:10)
Copyright (c) 1997-2017 The PHP Group
Zend Engine v3.1.0, Copyright (c) 1998-2017 Zend Technologies
```

If your version is older than 7.2.5, or you get an error about command not understood, then complete the steps below.

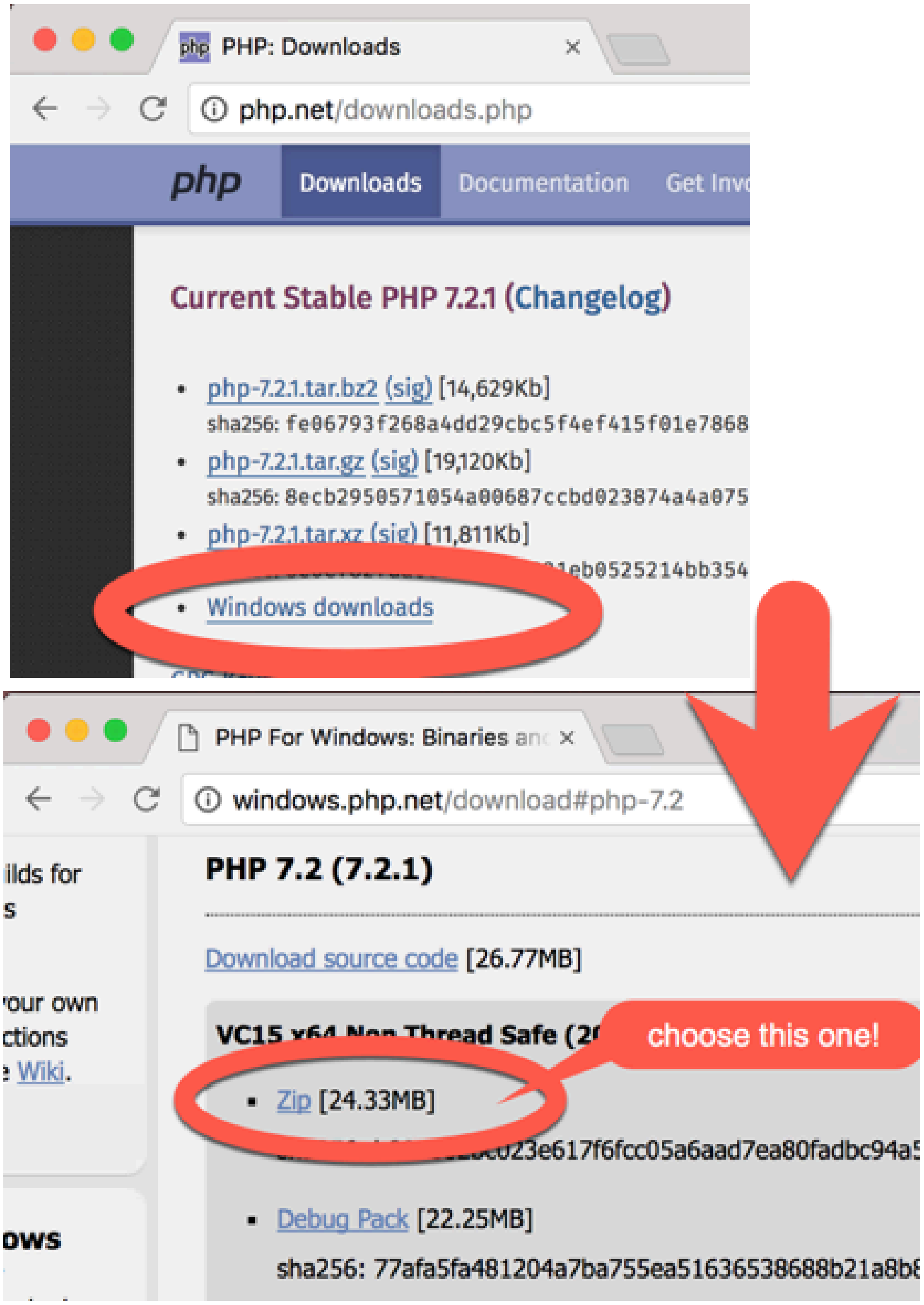
B.1.1 Download the latest version of PHP

Get the latest (7.4 at the time of writing) PHP Windows ZIP from:

- php.net click the **Windows Downloads** link

Figure B.1 shows a screenshot of the `php.net` general and Windows downloads page. The `ZIP` file to download (containing `php.exe` … don't download the source code version unless you want to build the file from code …):

Do the following:



PHP: Downloads

php.net/downloads.php

php Downloads Documentation Get Inv

Current Stable PHP 7.2.1 (Changelog)

- php-7.2.1.tar.bz2 (sig) [14,629Kb]
  sha256: fe06793f268a4dd29cbc5f4ef415f01e7868
- php-7.2.1.tar.gz (sig) [19,120Kb]
  sha256: 8ecb2950571054a00687ccbd023874a4a075
- php-7.2.1.tar.xz (sig) [11,811Kb]
  1eb0525214bb354
- Windows downloads

PHP For Windows: Binaries an

windows.php.net/download#php-7.2

PHP 7.2 (7.2.1)

Download source code [26.77MB]

VC15 x64 Non Thread Safe (2

- Zip [24.33MB]
  023e617f6fcc05a6aad7ea80fadbc94a5
- Debug Pack [22.25MB]
  sha256: 77afa5fa481204a7ba755ea51636538688b21a8b

Figure B.1: PHP.net / Windows ZIP download pages.

- unzip the PHP folder into: `C:\php`
- so you should now have a file `php.exe` inside `C:\php`, along with lots of other files
- make a copy the file `C:\php\php.ini-development`, naming the copy `C:\php\php.ini`
- open a new terminal CLI window (so new settings are loaded) and run `php --ini` to confirm the location of the `php.ini` file that you've just created. Note the following for a Mac - for Windows it should (hopefully) tell you it found the ini file in `c:\php\php.ini`:

```
$ php --ini
Configuration File (php.ini) Path: /Applications/MAMP/bin/php/php7.1.8/conf
Loaded Configuration File:         /Applications/MAMP/bin/php/php7.1.8/conf/php.ini
Scan for additional .ini files in: (none)
Additional .ini files parsed:      (none)
```

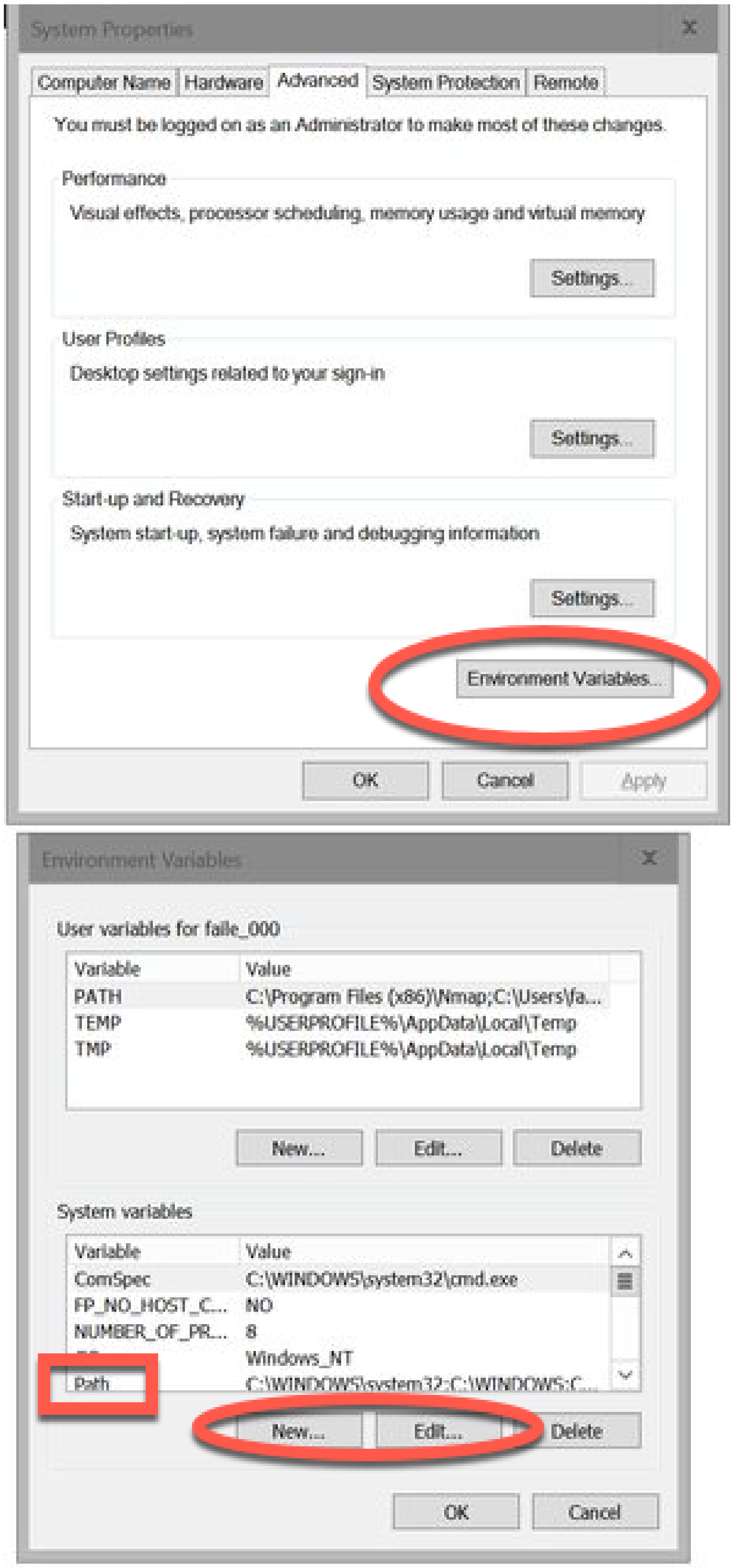
B.2 Add the path to `php.exe` to your System environment variables

Whenever you type a command at the CLI (Command Line Interface) Windows searches through all the directories in its `path` environment variable. In order to use PHP at the CLI we need to add `c:\php` to the `path` environment variable so the `php.exe` executable can be found.

Via the System Properties editor, open your Windows Evironment Variables editor. The **system** environment variablesa re in the lower half of the Environment Variables editor. If there is already a system variable named `Path`, then select it and click the **Edit** button. If none exists, then click the **New** button, naming the new variable **path**. Add a new value to the **path** variable with the value `c:\php`. Then click all the **Okay** buttons needed to close all these windows.

Now open a windows **Cmd** window and try the `php -v` - hopefully you'll see confirmation that your system now has PHP installed and in the **path** for CLI commands.

Figure B.2 shows a screenshot of the Windows system and environment variables editor.



You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage and virtual memory

User Profiles
Desktop settings related to your sign-in

Start-up and Recovery
System start-up, system failure and debugging information

User variables for faile_000

Variable	Value
PATH	C:\Program Files (x86)\Nmap;C:\Users\fa...
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_PR...	8
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;C...

Figure B.2: The Windows Environment Variables editor.

B.3 PHP Info & SQL driver test

For database work we need to enable the PDO[1] options for MySQL and SQLite (see later database exercises for how to do this)

Although PHP may have been installed, and its SQL drivers too, they may have not been enabled. For this module we'll be using the SQLite and MySQL drivers for PHP – to talk to databases. The function `phpinfo()` is very useful since it displays many of the settings of the PHP installation on your computer / website.

1. In the current (or a temporary) direcotry, create file `info.php` containing just the following 2 lines of code:

    ```
    <?php
    print phpinfo();
    ```

2. At the CLI run the built-in PHP web server to serve this page, and visit: `localhost:8000/info.php` in your web browser

    ```
    php -S localhost:8000
    ```

In the PDO section of the web page (`CTL-F` and search for `pdo` ...) we are looking for **mysql** and **sqlite**. If you see these then great!

Figure B.3 shows a screenshot the Windows system and environment variables editor.

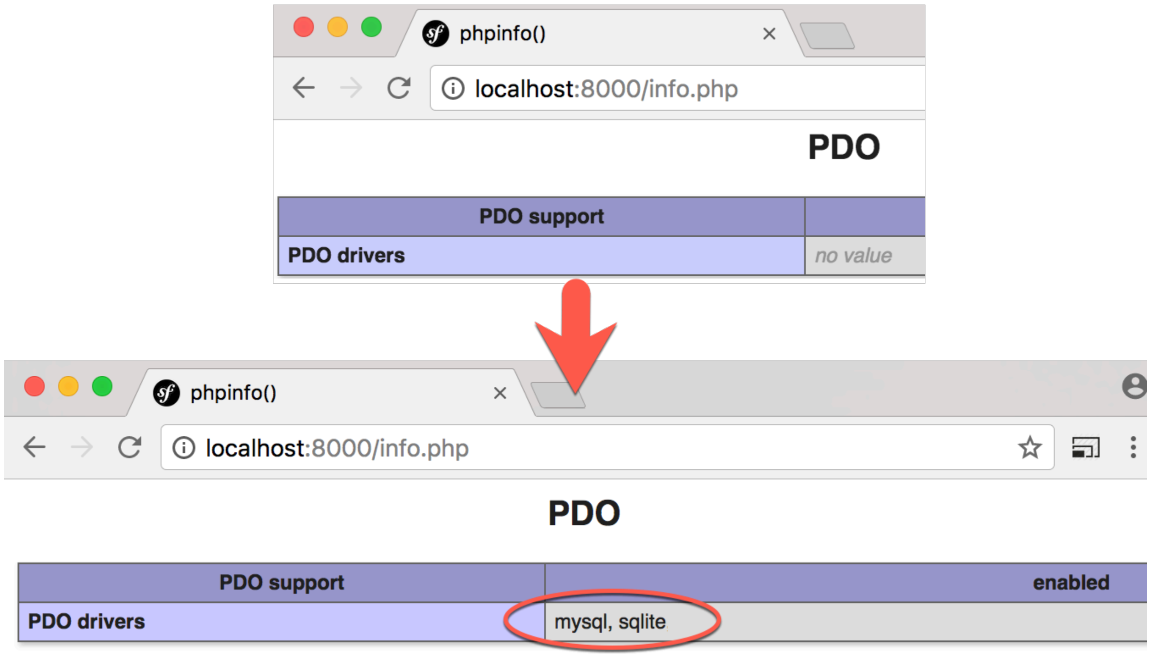
But, if you see "no value" under the PDO drivers section, then we'll need to edit file `c:\php\php.ini`:

1. In a text editor open file `c:\php\php.ini` and locate the "Dynamic Extensions" section in this file (e.g. use the editor Search feature - or you could just search for `pdo`)

2. Now remove the semi-colon `;` comment character at the beginning of the lines for the SQLite and MySQL DLLs to enable them as shown here:

    ```
    ;;;;;;;;;;;;;;;;;;;;;;
    ; Dynamic Extensions ;
    ;;;;;;;;;;;;;;;;;;;;;;

    .. other lines here ...
    extension=php_pdo_mysql.dll <<<<<<<<<< here is the PDO MYSQL driver line
    ;extension=php_pdo_oci.dll
    ;extension=php_pdo_odbc.dll
    ```

[1]PDO = PHP Database Objects, the modern library for managing PHP program communications with databases. Avoid using old libries like `mysql` (security issues) and even `mysqli` (just for MySQL). PDO offers an object-oriented, standardized way to communicate with many different database systems. So a project could change the databse management system (e.g. from Oracle to MySQL to SQLite), and only the database connetion optins need to change - all other PDO code will work with the new database system!



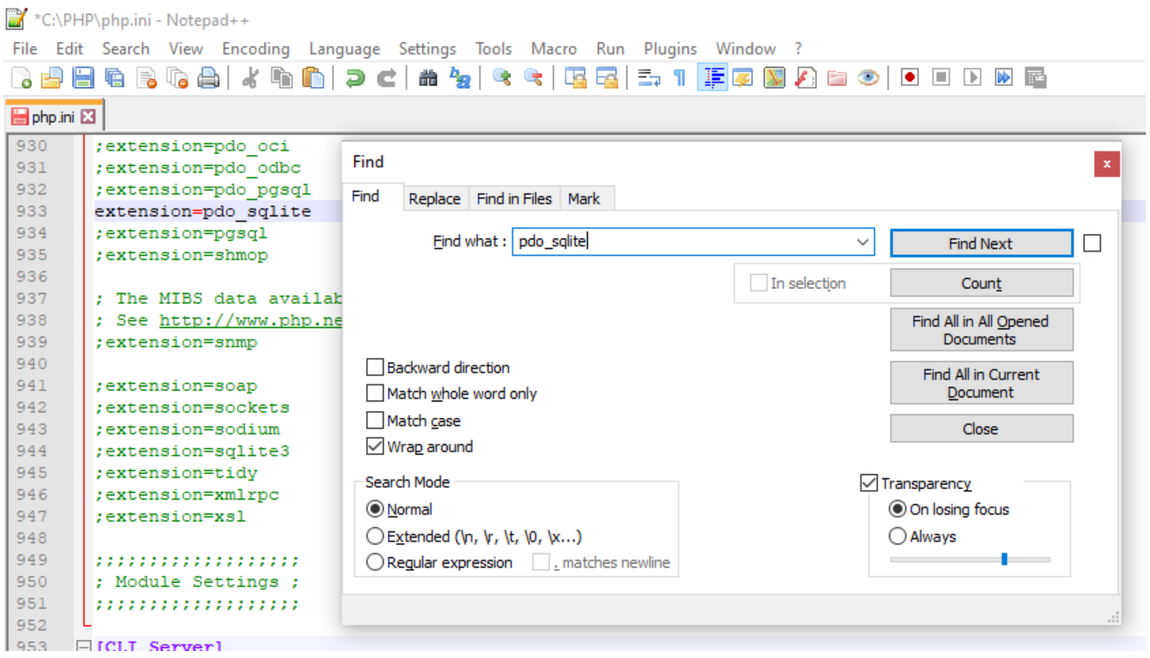
PDO

PDO support	
PDO drivers	no value

PDO

PDO support	enabled
PDO drivers	mysql, sqlite

Figure B.3: The PDO section of the `phpinfo()` information page.


*C:\PHP\php.ini - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
php.ini
Find
Find Replace Find in Files Mark
Find Next
Count
Find All in All Opened Documents
Find All in Current Document
Close
Search Mode
Transparency

```
;extension=pdo_oci
;extension=pdo_odbc
;extension=pdo_pgsql
extension=pdo_sqlite
;extension=pgsql
;extension=shmop

; The MIBS data availab
; See http://www.php.ne
;extension=snmp

;extension=soap
;extension=sockets
;extension=sodium
;extension=sqlite3
;extension=tidy
;extension=xmlrpc
;extension=xsl

;;;;;;;;;;;;;;;;;;;
; Module Settings ;
;;;;;;;;;;;;;;;;;;;

[CLI Server]
```

Find what : pdo_sqlite

☐ In selection

☐ Backward direction
☐ Match whole word only
☐ Match case
☑ Wrap around

Search Mode: [✓ Normal] Extended (\n, \r, \t, \0, \x...) Regular expression ☐ . matches newline

☑ Transparency: [✓ On losing focus] Always

Figure B.4: SQLite being enabled in php.ini in the Notepad++ editor.

```
;extension=php_pdo_pgsql.dll
extension=php_pdo_sqlite.dll <<<<<<<<  here is the PDO SQLITE driver line
```

3. Save the file. Close your Command Prompt, and re-open it (to ensure new settings are used).

   – Run the webserver again and visit: `localhost:8000/info.php` to check the PDO drivers.

NOTE: Knowing how to view `phpinfo()` is very handy when checking server features.

C

The Composer andd Symfonhy command line tools

C.1 Composer

Composer is a **fantastic** PHP tool for managing project dependencies (the libraries and class packages used by OO PHP projects).

1. ensure that Composer is up to date by running:

   ```
   composer self-update
   ```

2. enable the PDO options for MySQL and SQLite (see Appendix B for how to do this by editing ther `c:\php\php.ini` file …)

The Composer tool is actually a **PHAR** (PHP Archive) - i.e. a PHP application packaged into a single file. So ensure you have PHP installed and in your environment **path** before attempting to install or use Composer.

Ensure you have (or install) an up-to-date version of the Composer PHP package manager.

```
composer self-update
```

C.1.1 Windows Composer install

Get the latest version of Composer from

- getcomposer.org

- run the provided **Composer-Setup.exe** installer (just accept all the default options - do NOT tick the developer mode)
  - https://getcomposer.org/doc/00-intro.md#installation-windows ## The Composer PHP library tool

The Composer tool is actually a **PHAR** (PHP Archive) - i.e. a PHP application packaged into a single file. So ensure you have PHP installed and in your environment **path** before attempting to install or use Composer.

Ensure you have (or install) an up-to-date version of the Composer PHP package manager.

```
composer self-update
```

C.1.2 Windows Composer install

Get the latest version of Composer from

- getcomposer.org
- run the provided **Composer-Setup.exe** installer (just accept all the default options - do NOT tick the developer mode)
  - https://getcomposer.org/doc/00-intro.md#installation-windows

C.2 Symfony command line tool

D
Software tools

NOTE: All the following should already available on the college computers.

D.1 PHPStorm editor

Ensure you have your free education Jetbrains licence from:

- Students form: https://www.jetbrains.com/shop/eform/students (ensure you use your ITB student email address)

Downdload and install PHPStorm from:

- https://www.jetbrains.com/phpstorm/download/

To save lots of typing, try to install the following useful PHPStorm plugins:

- Twig
- Symfony
- Annotations

D.2 MySQL Workbench

While you can work with SQLite and other database management systems, many ITB modules use MySQLWorkbench for database work, and it's fine, so that's what we'll use (and, of course, it is already installed on the ITB windows computers …)

Download and install MySQL Workbench from:

- https://dev.mysql.com/downloads/workbench/

D.3 Git

Git is a fantastic (and free!) DVCS - Distributed Version Control System. It has free installers for Windows, Mac, Linus etc.

Check is Git in installed on your computer by typing `git` at the CLI terminal:

```
> git
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

...

collaborate (see also: git help workflows)
   fetch      Download objects and refs from another repository
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.

>
```

If you don't see a list of **Git** commands like the above, then you need to install Git on your computer.

D.4 Git Windows installation

Visit this page to run the Windows Git installer.

- https://git-scm.com/downloads

NOTE: Do **not** use a GUI-git client. Do all your Git work at the command line. It's the best way to learn, and it means you can work with Git on other computers, for remote terminal sessions (e.g. to work on remote web servers) and so on.