

2023

# Build a timekeeping system with



*vibe.d*  
www.vibe.d

BY LEARNING FROM A LEARNER

REY VALEZA

THE LOREM IPSUM COMPANY | Toronto, ON, Canada

Build web apps with



by learning from a learner

Rey Valeza

## Contents

Introduction.....	6
Configuring your system for Vibe.d .....	7
The default Hello, World! application .....	13
Using your own HTML page.....	19
Serving files other than HTML.....	24
Templates with Diet.....	29
A more interesting template page .....	34
Making use of your own functions .....	44
Using templates for ease of maintenance.....	47
Using include in templates.....	51
Layout without Bootstrap using CSS grid.....	53
A fixed navbar and a sticky footer without Bootstrap .....	60
Adding a modal dialogue to the menu .....	71
The web framework .....	75
A new project.....	76
Setting up the MySQL server and tools .....	81
The schema .....	89
Accessing a MySQL database .....	92
The EmployeeController class .....	95
The EmployeeModel class.....	98
The data-entry form for adding a new employee record.....	105
Saving data from the form into the database .....	111
Testing the whole thing.....	115
Listing all the employees .....	120

Retrieving and displaying an employee record for editing .....	134
Deleting records in the database .....	153
Finding an employee record by name .....	168
Capturing and displaying (some) error messages with _error .....	184
Logging in, authentication and authorization .....	206
Authenticating the user .....	208
Saving the login state to the session .....	211
Enforcing authorization through the session variable m_user .....	228
Logging out.....	236
All the sources so far.....	237
A new project.....	268
The schema .....	270
The base model .....	272
The employee model.....	273
The timecard model .....	283
The timesheet model .....	285
The administrators model .....	287
The base controller.....	289
The employee controller .....	292
The login system and a new menu .....	297
The index, time in and time out templates .....	313
The employee views .....	319
The home controller and app.d .....	326
Displaying the punch-in page .....	329
Adding employees .....	335

Punching in .....	343
Viewing the timecards .....	350
Punching out .....	359
Editing a timecard entry.....	367
Deleting a timecard .....	381
Generating the timesheet.....	395
The end.....	405
The main program.....	406
The base controller.....	407
The home controller .....	409
The employees controller.....	411
The employee model.....	416
The timecard controller.....	425
The timecard model .....	429
The timesheet controller .....	435
The timesheet model .....	437
The CSS files .....	440
The timecard views .....	456
The employee views .....	461
The layout views .....	468
The home view.....	471
The timesheet views.....	472
The project configuration file .....	474
That's all! Bye!	475



## Introduction

Vibe.d is an app development framework written in the D language. I am happy to note that an expert-level knowledge of D is *not* required to develop apps in Vibe.d, and I am no expert on either D or Vibe.d.

I wrote this tutorial because I want more people to be aware of Vibe.d, its simplicity and its productivity. I hanker for Vibe.d whenever I look at the complexity of Spring / Hibernate and the technologies needed just to build a simple application. Yes, Java is fantastic, fantastically complex. The Java language itself isn't that complex but it is the ecosystem built around it that made things complex. Most people cannot get that simplicity is the summit of sophistication: the creator(s) labored so hard to make things simple. And Vibe.d is the embodiment of simplicity.

I used PHP, Java/JSP/servlets, C#/ASP.Net, Ruby on Rails, Meteor, even Lotus Notes Domino for web development, but it is Vibe.d that I find the most straightforward and most productive.

While reviewing the earlier version of this tutorial, I felt I should expand more on the MySQL database side as there are so many ageing deployed web apps out there with a MySQL database backend that feel the need to port to a more modern framework, as well as a large number of small businesses with a huge pile of MySQL data who desire to port their apps to the web. This book is by no means comprehensive since the purpose is to offer a tutorial for learners of a new web framework.

I found there aren't any drastic changes since the last time I used Vibe.d and I'm glad. That is how it should be. That is why PHP is still so popular: simplicity and no drastic changes. Young people can pick up any old book or tutorial and find it isn't much different from the current PHP.

I see that there is a dearth of tutorials on Vibe.d. I developed an app a few years back. After realizing I have to learn Vibe.d again when I found out I forgot so much, I decided to write this book while I was re-learning it.

The Vibe.d framework was created by Sönke Ludwig (there are now a bunch of guys helping in the project) out of frustration with existing frameworks, notably Node.js. Vibe.d doesn't rely on hype, which plays against it, so virtually nobody knows about it. So I wrote this book.

## Configuring your system for Vibe.d

I am using Windows 10 but you should be fine with another OS you are familiar with, and I assume you know the equivalent commands in your system's shell terminal. We will be using the terminal or command window a lot. I wrote the code in my Linux box then ported them to Windows to write this book, as I know that majority of developers prefer Windows. I find that Linux users easily follow Windows tutorials anyway, which means Linux users are or were Windows users too.

D is a language that compiles to native code. The Vibe.d framework is essentially a set of libraries written in D, which means your code needs to be linked to those libraries and compiled into a native executable. To develop web apps in Vibe.d, you need at least a D compiler and a text editor.

Regarding compilers, there are three choices: DMD, GDC and LDC. It is advised that, for development purposes, we should use the DMD compiler for its speed of compilation, then, when we are ready to deploy the final production version, use one of the other compilers (GDC or LDC) for the runtime speed and optimizations of the binary output. This book uses the DMD compiler. I hope you do too, so download and install the compiler by heading to

<https://dlang.org/>

and clicking on the 'Download Windows Installer' button.

mentation ▾   Downloads   Packages   Community ▾   Resources ▾

Search   Entire Site  

[Report a bug](#) [Improve this page](#)

eneral-purpose programming language  
atic typing, systems-level access, and  
yntax. With the **D Programming  
ge**, write fast, read fast, and run fast.  
le, fast.

[Download Windows Installer](#)

Other Downloads

Latest version: 2.094.2 – [Changelog](#)

Support

ible through the hard work and dedication of many  
fit organization. You can help further the developm

[Donate](#)   [Leave a review](#)

Lots of ❤️ to go around

Installers

**Sort an Array at Compile-Time** [your code here](#)

```
void main()
{
    import std.algorithm, std.stdio, std.file, std.range;
    enum cols = 14;
    // Split file into 14-byte chunks per row
    thisExePath.File("rb").byChunk(cols).take(20).each!(chunk =>
        // Use range formatting to format the
        // hexadecimal part and align the text part
        writeln!("%(02X %)*s %s"(chunk,
            3 * (cols - chunk.length), "", // Padding
            chunk.map!(c => // Replace non-printable
                ...
            )
        )
    )
}
```

**Save As**

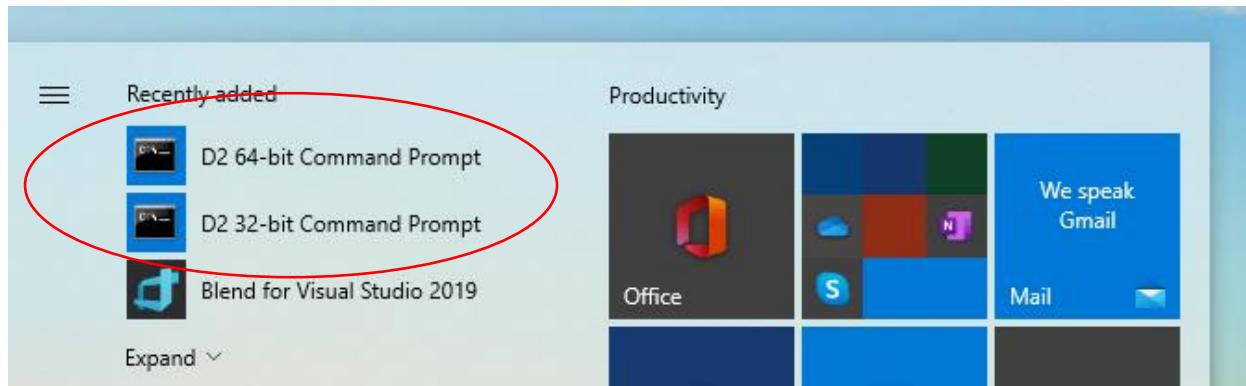
← → ↑ ↓ This PC > Downloads >

Name	Date modified	Type	Size
others	2020-12-04 1:56 PM	File folder	
vs_Community.exe	2020-12-03 5:21 PM	Application	1,376 KB
VisualD-v1.0.1-dmd-2.093.1-ldc2-1.23.0.e...	2020-12-03 5:21 PM	Application	86,394 KB
VSCodeUserSetup-ia32-1.51.1.exe	2020-12-03 5:19 PM	Application	59,356 KB
dmd-2.094.2fromlinux.exe	2020-12-03 5:18 PM	Application	31,150 KB

File name:  Save as type:

Save Cancel

After the installation in Windows 10, you should have new entries in your start menu.



But even if you don't, just open a terminal or command window.

To test the installation of the DMD compiler, type `dmd` on the terminal window. If the installation is successful, it will fill the window with help information.

```
C:\Users\Owner\Downloads>dmd
DMD32 D Compiler v2.100.2-dirty
Copyright (C) 1999-2022 by The D Language Foundation, All Rights Reserved written by Walter Bright
```

```
Documentation: https://dlang.org/
Config file: C:\D\dmd2\windows\bin\sc.ini
Usage:
  dmd [<option>...] <file>...
  dmd [<option>...] -run <file> [<arg>...]
```

...

And you will see a bunch of other messages that fill up and scroll down the screen.

Here are two DMD options:

dmd --help will display the same information.

dmd --version will show you just the DMD compiler version.

The DMD installer comes with the dub package manager. It is the default package manager for D projects and was written by the creator of the Vibe.d framework himself.

To test dub, simply type in your terminal

dub

If the compiler complains there is no project manifest, you are good.

Another way to test is to type

dub help

or

dub -h

or

dub --help

These will show the dub help information.

To show just the version number, type

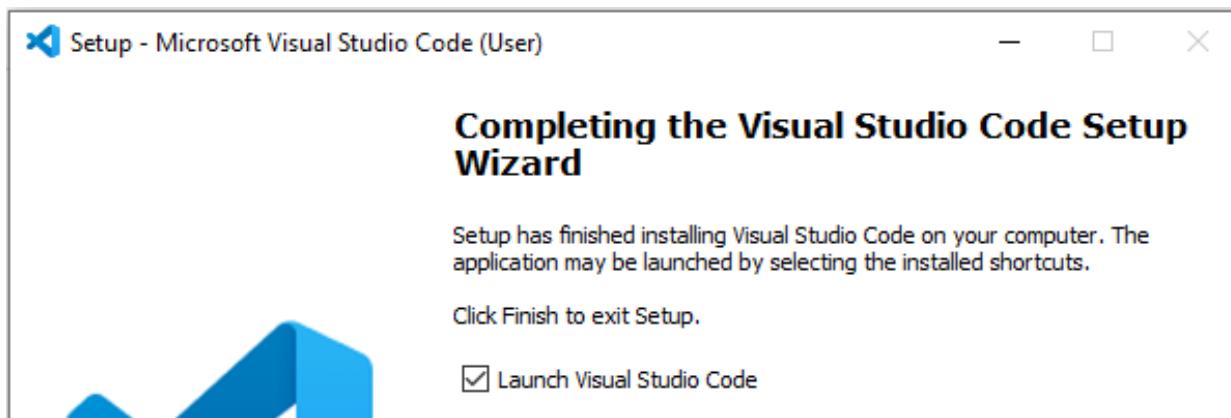
dub --version

After the compiler is installed, the next thing you need is a text editor or an IDE. You can stick with your favorite IDE/text editor while following this book but this book will use the ubiquitous and free Visual Studio Code, with versions for Windows, MacOS and Linux. Since you are using Windows, you must have this already, but to those who haven't yet, head to

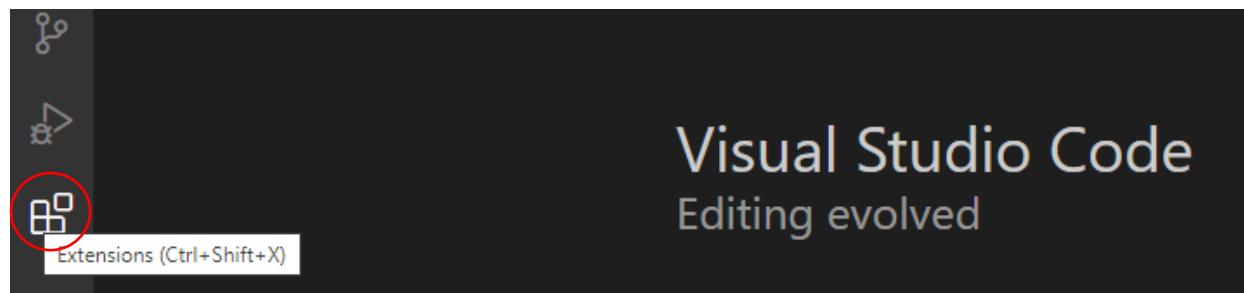
<https://code.visualstudio.com/download>

and choose the installer appropriate for your system.

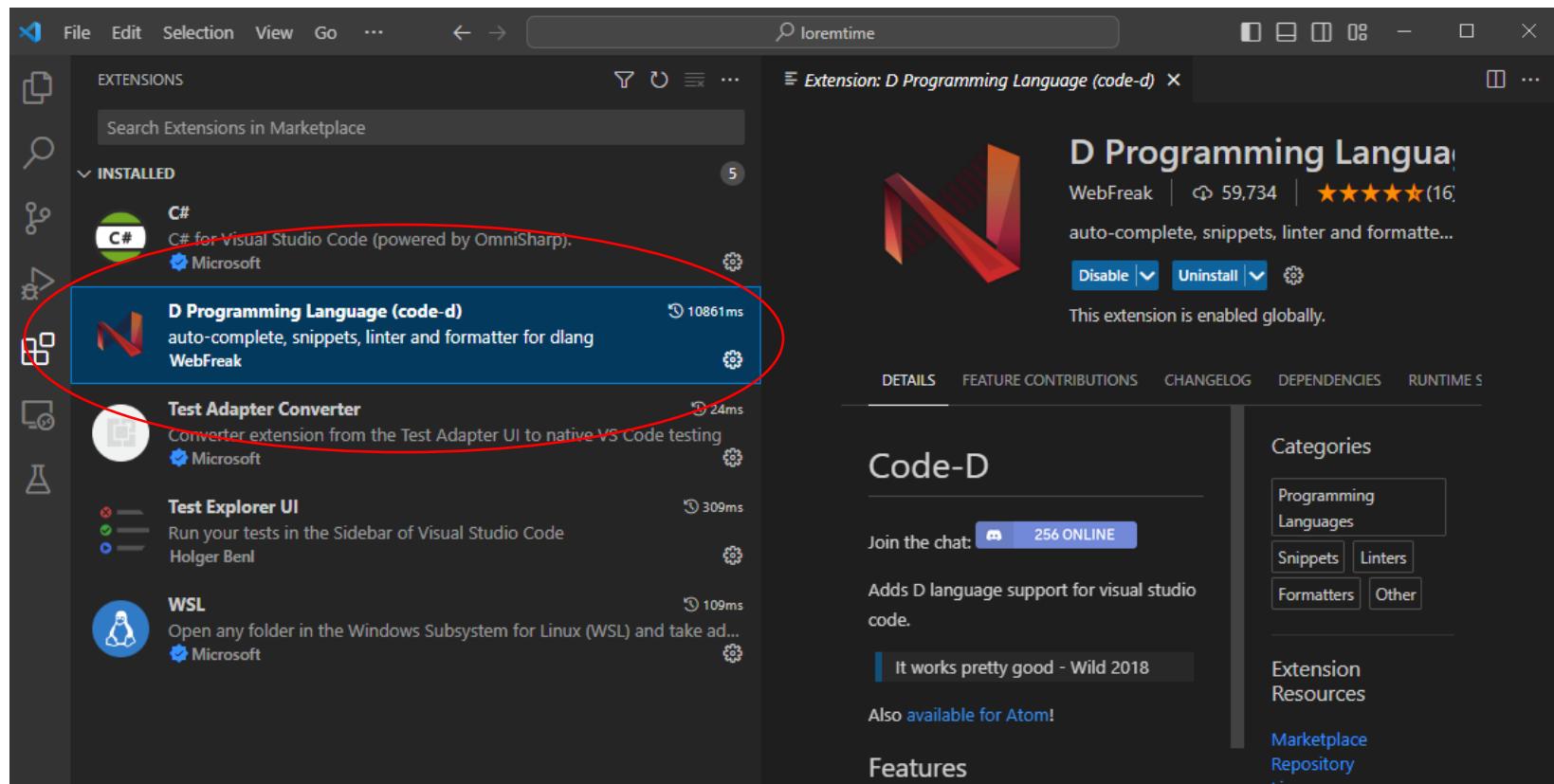
After downloading, simply double-click on the downloaded file and follow the instructions on the setup wizard.



After installation, add the D language extension so VS Code can recognize D language code.



Add the D Programming Language (code-d) extension pack. With this pack, you will have features like syntax highlighting, code-completion and some error flagging.



Now you are ready to build web apps in Vibe.d.

## The default Hello, World! application

In the terminal window, go to your folder of choice or create a new one, for example c:\vibeprojects. Inside that folder, type the command

```
dub init hello --type=vibe.d
```

or

```
dub init hello -t vibe.d
```

For now, press Enter at every prompt that follows to accept the defaults.

```
c:\vibeprojects> dub init hello -t vibe.d
Package recipe format (sdl/json) [json]:
Name [hello]:
Description [A simple vibe.d server application.]:
Author name [Owner]:
License [proprietary]:
Copyright string [Copyright © 2023, Owner]:
Add dependency (leave empty to skip) []:
    Success created empty project in c:\vibeprojects\hello
        Package successfully created in hello
```

Go inside the newly created hello project folder and explore it.

```
c:\vibeprojects>cd hello
c:\vibeprojects\hello>dir
Volume in drive C has no label.
Volume Serial Number is 4E0A-BFEF
```

```
Directory of c:\vibeprojects\hello
```

```
2023-01-04 01:30 PM <DIR> .
2023-01-04 01:30 PM <DIR> ..
2023-01-04 01:30 PM      131 .gitignore
2023-01-04 01:30 PM      215 dub.json
2023-01-04 01:30 PM <DIR>     public
2023-01-04 01:30 PM <DIR>     source
2023-01-04 01:30 PM <DIR>     views
2 File(s)      346 bytes
5 Dir(s) 134,991,044,608 bytes free
```

You are now inside the root directory of the hello application. To compile and run this application in one step (you haven't done anything yet!), type

```
dub
```

Since this is the first time you will be running this command, this will download all the dependencies, then compile, then link, and then run the application.

```
c:\vibeprojects\hello>dub
```

There will be a lot of messages before you see this at the tail end:

```
C:\Users\Owner\AppData\Local\dub\packages\vibe-core-1.22.5\vibe-core\source\vibe\internal\traits.d-mixin-429(429,48):
Deprecation: scope variable `'_param_1` assigned to non-scope parameter `dst` calling `read`
Linking hello
Copying files for vibe-d:tls...
Running hello.exe
[main(----) INF] Listening for requests on http://[::1]:8080/
[main(----) INF] Listening for requests on http://127.0.0.1:8080/
[main(----) INF] Please open http://127.0.0.1:8080/ in your browser.
```

The last three lines indicate that the application has successfully compiled and the server is now running.

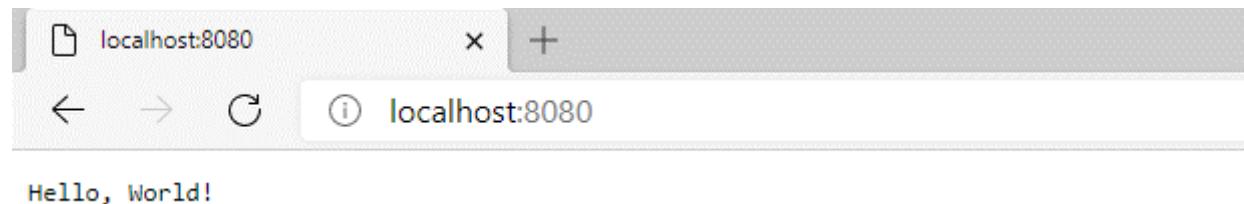
Open your browser and point it to localhost, port 8080.

<http://127.0.0.1:8080>

or

<http://localhost:8080>

and you will see the ‘Hello, World!’ message in your browser.



To stop the running application, press Ctrl-C (Control-C) on the terminal window. Sometimes you have to do it twice.

```
[00000000(----) INF] Received signal 2. Shutting down.  
[main(----) INF] Stopped to listen for HTTP requests on ::1:8080  
[main(----) INF] Stopped to listen for HTTP requests on 127.0.0.1:8080  
^C  
c:\vibeprojects\hello>
```

The common command options for dub are

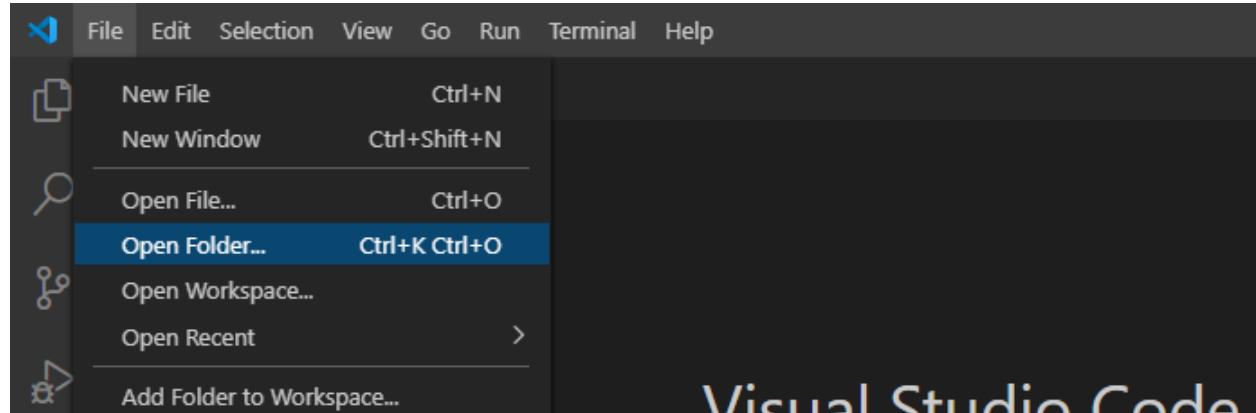
dub init - create a new project

dub run - build and run combined

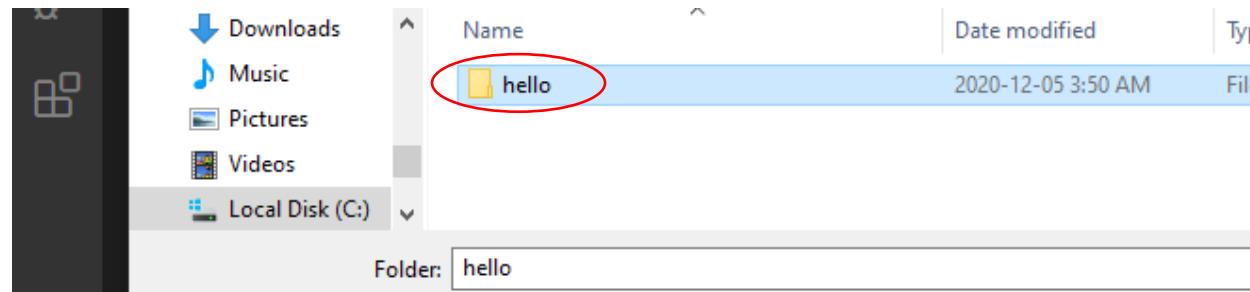
- dub - same as dub run
- dub build - compile and link the whole project and its dependencies but don't run
- dub test - runs the unit tests

Let us take a closer look at the default application.

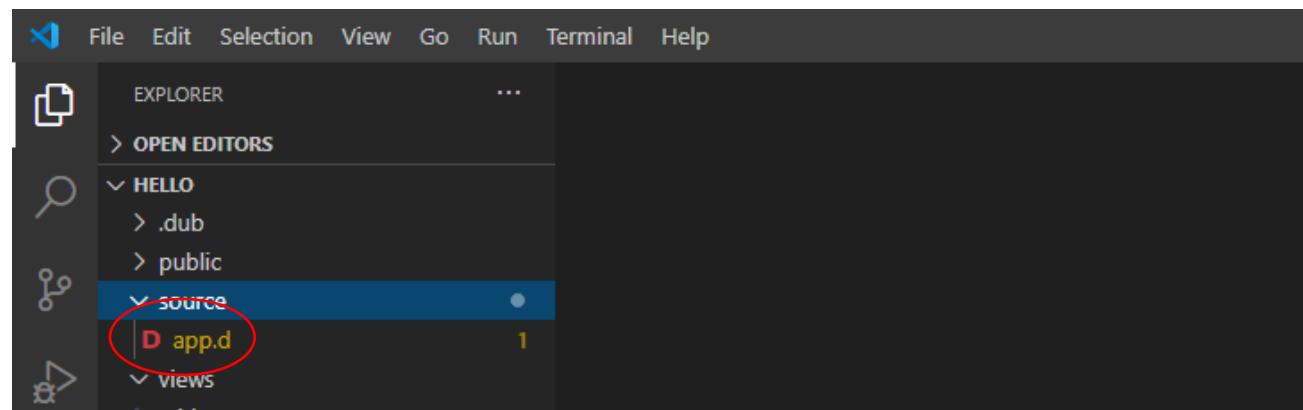
Open VS Code and go to File->Open folder...



Then open the new application folder hello.



In the Explorer window of VS Code, open source\app.d.



And you will see the automatically generated code.

```
import vibe.vibe;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];
    auto listener = listenHTTP(settings, &hello);
```

```
scope (exit)
{
    listener.stopListening();
}

logInfo("Please open http://127.0.0.1:8080/ in your browser.");
runApplication();
}

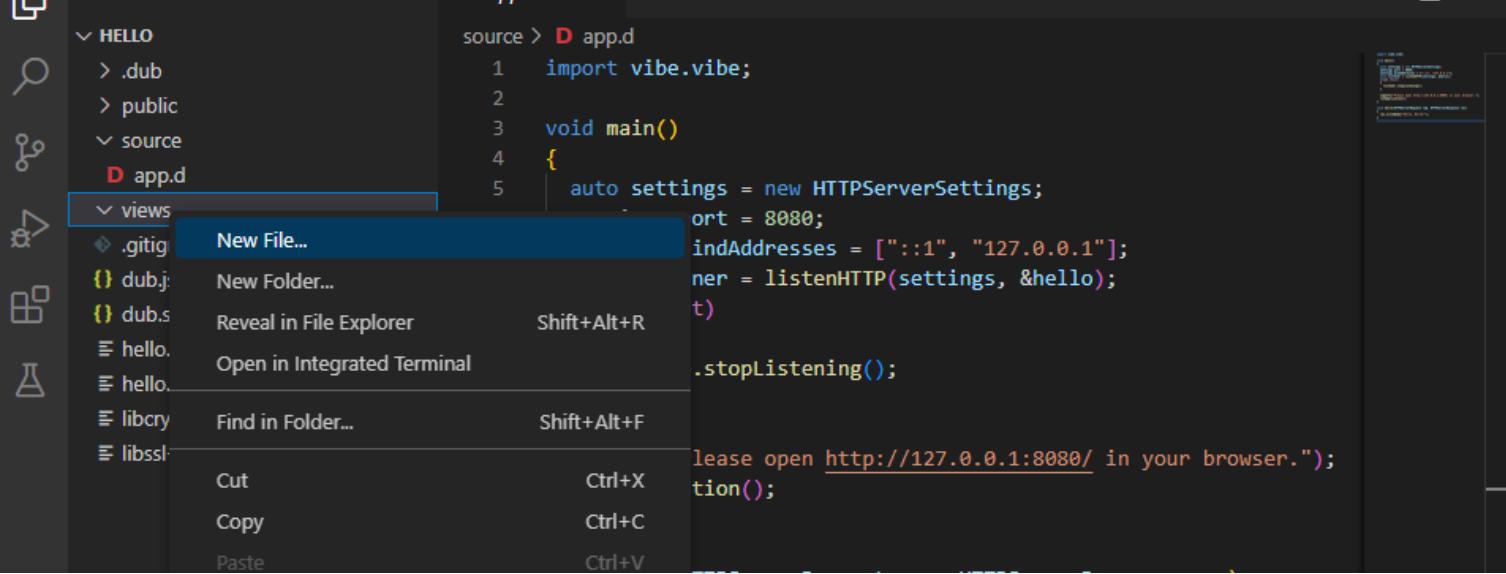
void hello(HTTPServerRequest req, HTTPServerResponse res)
{
    res.writeBody("Hello, World!");
}
```

In the main() method of source\app.d:

- a new HTTPServerSettings object is created
- the HTTP port number is set to 8080 (which you can change)
- the IP the server listens to is set to localhost in both IPv6 (“::1”) and IPv4 (“127.0.0.1”) format
- the listenHTTP(settings, &hello) call means ‘run the hello() function using these settings’
- the runApplication() call starts the ball rolling (starts the event loop)

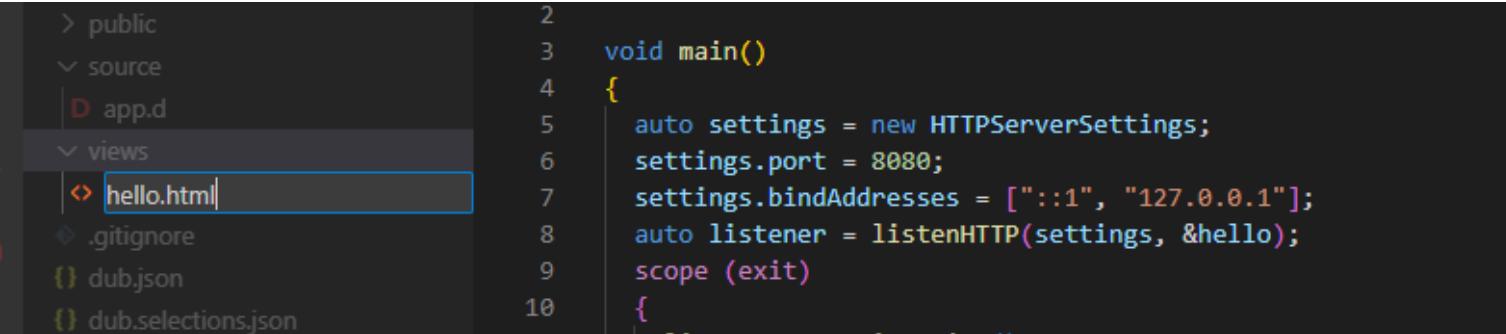
## Using your own HTML page

Inside the views folder, create a new HTML file named views\hello.html



The screenshot shows a code editor interface with a sidebar containing project files. In the sidebar, under the 'source' folder, there is a 'views' folder which is currently selected. A context menu is open over this 'views' folder, with the 'New File...' option highlighted.

```
source > D app.d
1 import vibe.vibe;
2
3 void main()
4 {
5     auto settings = new HTTPServerSettings();
6     port = 8080;
7     bindAddresses = [":1", "127.0.0.1"];
8     server = listenHTTP(settings, &hello);
9     t)
10
11     .stopListening();
12
13     lease open http://127.0.0.1:8080/ in your browser.");
14     tion();
```



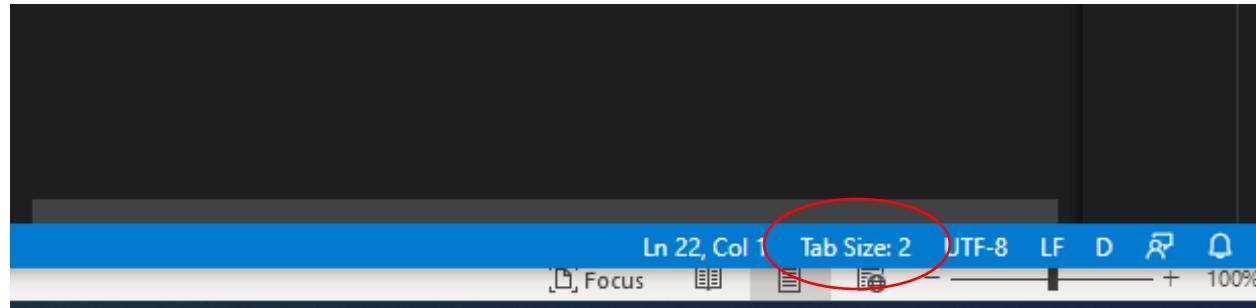
The screenshot shows the contents of the 'hello.html' file within the 'views' folder. The file contains a single line of text: 'lease open <http://127.0.0.1:8080/> in your browser.");'. The file path 'views\hello.html' is visible in the sidebar.

```
> public
< source
  D app.d
    < views
      hello.html
      .gitignore
      {} dub.json
      {} dub.selections.json
      2
      3 void main()
      4 {
      5     auto settings = new HTTPServerSettings();
      6     settings.port = 8080;
      7     settings.bindAddresses = [":1", "127.0.0.1"];
      8     auto listener = listenHTTP(settings, &hello);
      9     scope (exit)
      10
```

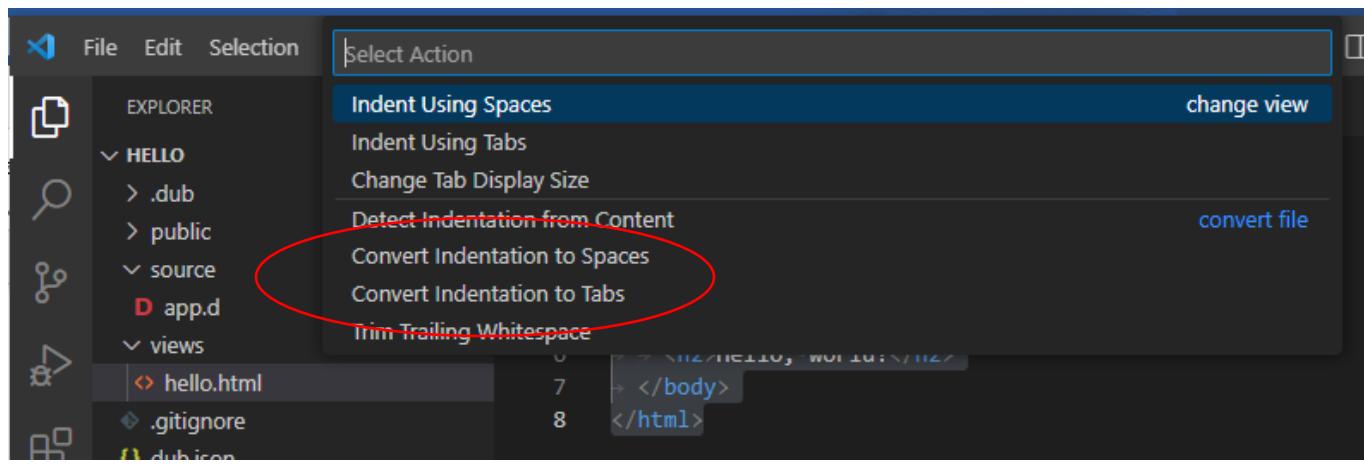
with the following contents:

```
<html>
  <head>
    <title>Hello world!</title>
  </head>
  <body>
    <h2>Hello, world!</h2>
  </body>
</html>
```

Vibe.d is a stickler to proper indentation so follow the indentation rules for HTML tags. Sometimes, your editor saves tabs as spaces so make sure the spacing in the file is consistent, either all tabs or all spaces, or you will see mysterious errors when compiling. If you encounter that, click on the indentation setting at the bottom of VS Code



And the drop-down indentation options will show at the top



Choose either ‘Convert indentations to Spaces’ or ‘Convert indentations to Tabs’, whichever is your preference as long as you are consistent, then save the file again.

Next, edit source\app.d to make it display the hello.html page.

```
import vibe.vibe;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];
    auto listener = listenHTTP(settings, staticTemplate!"hello.html");
    scope (exit) listener.stopListening();
    runApplication();
}

void hello(HTTPServerRequest req, HTTPServerResponse res)
{
    res.writeBody("Hello, World!");
}
```

{

And delete the hello() function at the bottom as it is no longer called.

Stop the application if it is still running by pressing Ctrl-C in the terminal window (maybe twice).

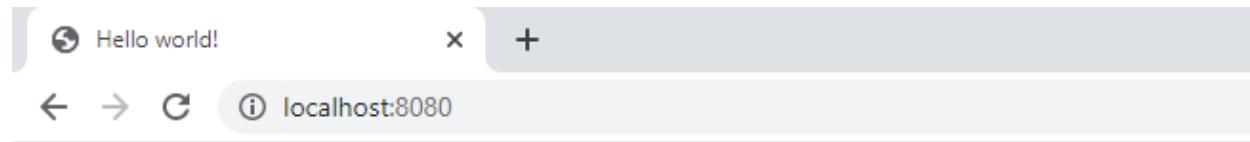
Compile and run the project with dub.

```
c:\vibeprojects\hello>dub
...
C:\Users\Owner\AppData\Local\dub\packages\vibe-d-0.9.5\vibe-d\stream\vibe\stream\wrapper.d(375,12):      Deprecation:
reference to local variable `chars` assigned to non-scope parameter `elems` calling `put`

    Linking hello
vibe-d_web.lib(common.obj) : warning LNK4255: library contain multiple objects of the same name; linking object as if no debug
info
diet-ng.lib(html.obj) : warning LNK4255: library contain multiple objects of the same name; linking object as if no debug info
eventcore.lib(driver.obj) : warning LNK4255: library contain multiple objects of the same name; linking object as if no debug info
eventcore.lib(driver.obj) : warning LNK4255: library contain multiple objects of the same name; linking object as if no debug info
eventcore.lib(core.obj) : warning LNK4255: library contain multiple objects of the same name; linking object as if no debug info

    Finished To force a rebuild of up-to-date targets, run again with --force
    Copying files for vibe-d:tls...
    Running hello.exe
[main(---) INF] Listening for requests on http://[::1]:8080/
[main(---) INF] Listening for requests on http://127.0.0.1:8080/
ser.
```

This time around, you notice that the required libraries were no longer downloaded so the compilation is a bit faster. Refresh your browser and you should see the new message.



## Hello, world!

To stop the running app, press Ctrl-C (maybe twice).

As you have seen, this is the conventional directory layout of a Vibe.d app:

\source - your D code should reside here

\views - the HTML or template pages stay here

\public - for other files such as raw CSS, scripts, images, videos, etc.

## Serving files other than HTML

What if an HTML page requires CSS and JavaScript files?

Create views\helloagain.html with the following code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My webpage!</title>
    <link rel="stylesheet" href="css/helloagain.css" />
    <script async src="js/helloagain.js"></script>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <h4 id='date'></h4>
    <div class="image-section">
      <div class="section-style">
        
        <p>A random image courtesy of unsplash.com.</p>
      </div>
      <div class="section-style">
        
        <p>A random image courtesy of unsplash.com.</p>
      </div>
    </div>
    <div class="image-section">
      <div class="section-style">
        
        <p>A random image courtesy of unsplash.com.</p>
      </div>
      <div class="section-style">
```

```

<p>A random image courtesy of unsplash.com.</p>
</div>
</div>
</body>
</html>
```

Actually, I got this from the wild wild web courtesy of Programming Liftoff  
(<https://dev.to/programliftoff/create-a-basic-webpage-with-css-and-javascript--104i>)

Next, create a css\ folder inside the public\ folder (you can right-click on it)

Then create the public\css\helloagain.css stylesheet file (also from Programming Liftoff) inside that new folder.

```
body
{
  text-align: center;
  background-color: #f0e8c5;
}

div
{
  margin-top: 15px;
}

.image-section
{
  display: flex;
  justify-content: center;
}

.section-style
```

```
{  
    margin-right: 25px;  
    margin-left: 25px;  
    background-color: white;  
}
```

Next, create the public\js folder.

Then create the public\js\helloagain.js with the following contents (also from Programming Liftoff)

```
document.getElementById('date').innerHTML = new Date().toLocaleString();
```

Then edit source\app.d to use a router

```
import vibe.vibe;  
  
void main()  
{  
    auto settings = new HTTPServerSettings;  
    settings.port = 8080;  
    settings.bindAddresses = ["::1", "127.0.0.1"];  
  
    auto router = new URLRouter;  
    router.get("/", staticTemplate!"helloagain.html");  
    router.get("*", serveStaticFiles("public/"));  
  
    auto listener = listenHTTP(settings, router);  
    scope (exit) listener.stopListening();  
    runApplication();  
}
```

The line

```
router.get("/", staticTemplate! "helloagain.html");
```

means display helloagain.html when the URL simply shows “/”. This means the helloagain.html is the index page or home page. The “/” URL expands to <http://localhost:8080/>, which is the root of the application.

The line

```
router.get("*", serveStaticFiles("public/"));
```

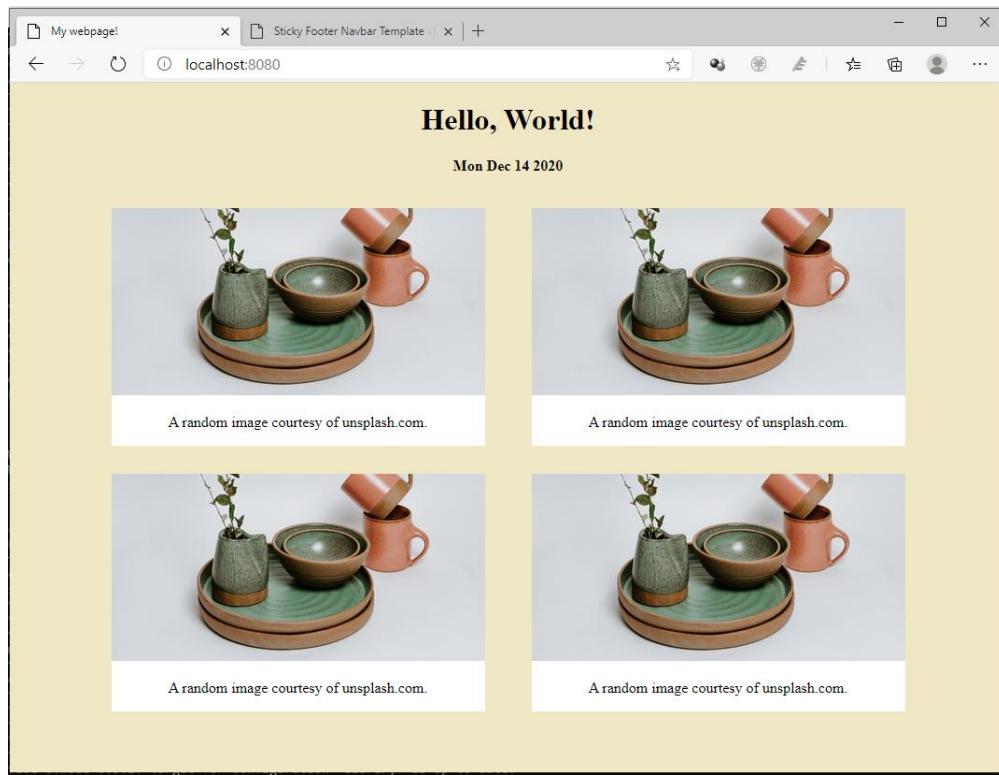
means for all other files not indicated by any route, treat them as static files and look for them inside the public/ folder.

We indicated in the helloagain.html page that the stylesheet and the javascript files are in /public/css/ and /public/js/ but without mentioning the public/ folder:

```
<link rel="stylesheet" href="css/helloagain.css" />
<script async src="js/helloagain.js"></script>
```

We do not include the public/ folder in the path.

After refreshing the browser, something similar is what you will see:



The images you see will most likely be different as they are randomly selected every time you refresh the page, so try to refresh the page a few times to see different pictures.

## Templates with Diet

Vibe.d comes with a templating engine called Diet.

To show how Diet templates work, create views\simplehello.html with the following code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simple hello!</title>
  </head>
  <body>
    <h1>Simple hello, World!</h1>
    <h2>How are you doing today?</h2>
  </body>
</html>
```

Edit the source\app.d file to make it display the views\simplehello.html file.

```
import vibe.vibe;

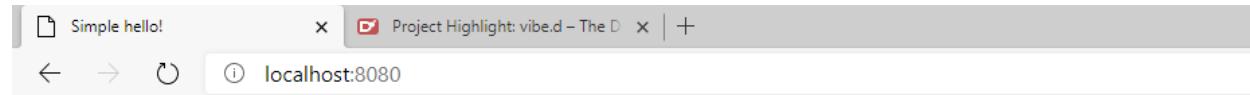
void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];

    auto router = new URLRouter;
    router.get("/", staticTemplate!"simplehello.html");
    router.get("*", serveStaticFiles("public/"));

    auto listener = listenHTTP(settings, router);
```

```
    scope (exit) listener.stopListening();
    runApplication();
}
```

Then build and run the project and refresh the browser to see the new message:



## Simple hello, World!

**How are you doing today?**

We just tested that the new HTML file is working. Now, let's convert the HTML file into a Diet template file.

Save the views\simplehello.html file as views\simplehello.dt with this contents:

```
doctype html
html
  head
    title Simple hello using a template
  body
    h1 Simple hello world from a template!
    h2 How are you doing today?
    | Please visit the home of <a href="https://vibed.org/">Vibe.d</a>
```

It feels weird at first, but then you realize there is less to type when using Diet templates.

Then edit source\app.d to make it display simplehello.dt instead:

```
import vibe.vibe;
```

```
void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];

    auto router = new URLRouter;
    router.get("/", staticTemplate!"simplehello.dt");
    router.get("*", serveStaticFiles("public/"));

    auto listener = listenHTTP(settings, router);
    scope(exit) listener.stopListening();
    runApplication();
}
```

In your terminal, stop the app if it is still running (Ctrl-C), then compile and run.

```
dub --force
```

We sometimes use the --force option to tell dub to recompile everything again, just in case it fails to see that some files were changed and needed to be recompiled or that there are updated libraries that needed to be downloaded again.

Refresh your browser to see the changes.



## Simple hello world from a template!

How are you doing today?

Please visit the home of [Vibe.d](#)

The syntax of Diet templates is based on Pug/Jade templates (<http://jade-lang.com/>).

Here are some of the basic rules:

The default Diet template file extension is .dt

The first word on a line is usually an HTML tag

```
span Hello, world!  
br
```

| (pipe) tag at the start of a line means the line is composed of plain text  
| Hello world

:javascript tag at the start of a line means JavaScript code on the next indented lines

```
:javascript  
    alert('Hello, world!');
```

:css tag at the start of a line means CSS code on the following indented lines

```
:css  
    .alert-message  
    {  
        font-size: 20px;  
        color: brown;  
    }
```

The attributes of a tag are comma-separated values inside parentheses

```
div(class="high-status", id="high-status-1");
```

A tag followed by a . (dot) followed by an identifier name means a CSS class name

```
div.high-status
```

A tag followed by a # (hash) followed by an identifier name means a CSS ID name

```
div#high-status-1
```

CSS class names and IDs can be combined

```
div.high-status#high-status-1
```

CSS classes can be strung together with dots

```
div.high-status.row-major.row-inner
```

Plain text following a tag should be separated by a space

```
div Hello world!  
div(class="high-status") Hello, world!
```

A tag ending in a . (dot) means multi-line plain text follows

```
div.  
    These lines are all simple text  
    but may contain HTML code  
    like <a href="/">this</a>
```

Plain text may contain HTML code

```
| Like <a href="/">this</a>.
```

Nesting of elements is done by adding indentation

```
div Like this  
    span this <span> is inside the div  
        div this <div> is inside the span  
            | and there is no need for end tags
```

## A more interesting template page

Let's have a more interesting example.

Create another file named views\notsimplehello.html with the following contents:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Demo site</title>
  </head>
  <body>
    <header>
      <h2>Welcome to our not so simple site</h2>
    </header>
    <nav>
      <ul>
        <li><a href="/">Home</a></li>
        <li><a href="/about">About us</a></li>
        <li><a href="/events">Events</a></li>
        <li><a href="/contact">Contact us</a></li>
      </ul>
    </nav>
    <article>
      <h1>Welcome to our not so simple site!</h1>
      <div>
        Lorem ipsum dolor sit amet, consectetur adipisci elit, sed
        eiusmod tempor incididunt ut labore et dolore magna aliqua.
        Ut enim ad minim veniam, quis nostrum exercitationem ullam
        corporis suscipit laboriosam, nisi ut aliquid ex ea commodi
        consequatur. Quis aute iure reprehenderit in voluptate velit
        esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
```

```
    obcaecat cupiditat non proident, sunt in culpa qui officia  
    deserunt mollit anim id est laborum.  
  </div>  
</article>  
<footer>  
  <p>Copyright 2021 Notsosimple Company</p>  
</footer>  
</body>  
</html>
```

Then edit source\app.d to use the new file:

```
import vibe.vibe;  
  
void main()  
{  
    auto settings = new HTTPServerSettings;  
    settings.port = 8080;  
    settings.bindAddresses = [":1", "127.0.0.1"];  
  
    auto router = new URLRouter;  
    router.get("/", staticTemplate!"notsimplehello.html");  
    router.get("*", serveStaticFiles("public/"));  
  
    auto listener = listenHTTP(settings, router);  
    scope (exit) listener.stopListening();  
    runApplication();  
}
```

Then compile, run and refresh your browser:



Welcome to our not so simple site!

  Lorem ipsum dolor sit amet, consectetur adipisci elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Copyright 2022 Notsosimple Company

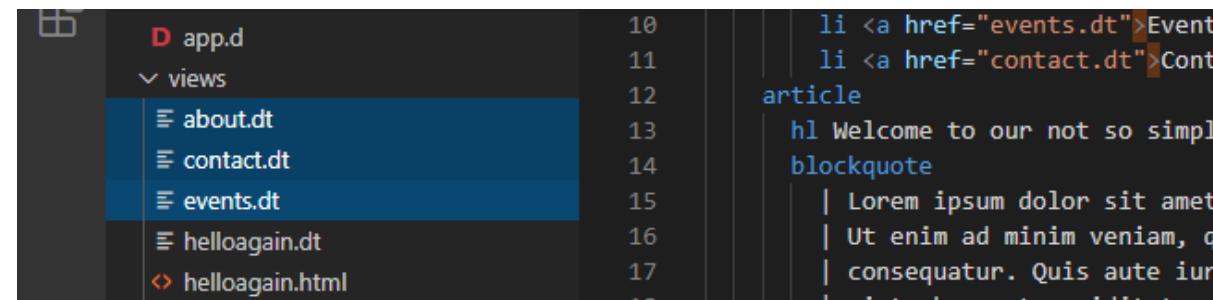
Now that we have shown the HTML page is working, let's convert it into a Diet template. Save notsimplehello.html as notsimplehello.dt with the following contents:

```
html
  head
    title Demo site
  body
    header
      h2 Welcome to our not so simple site
    nav
      ul
        li <a href="/">Home</a>
        li <a href="/about">About us</a>
        li <a href="/events">Events</a>
```

```
    li <a href="/contact">Contact us</a>
article
    h1 Welcome to our not so simple site!
    div.
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
        eiusmod tempor incididunt ut labore et dolore magna aliqua.
        Ut enim ad minim veniam, quis nostrum exercitationem ullam
        corporis suscipit laboriosam, nisi ut aliquid ex ea commodi
        consequatur. Quis aute iure reprehenderit in voluptate velit
        esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
        obcaecat cupiditat non proident, sunt in culpa qui officia
        deserunt mollit anim id est laborum.
footer
    p Copyright 2022 Notsosimple Company
```

There is really less to type when you use Diet templates.

Then copy notsimplehello.dt into about.dt, events.dt and contact.dt inside the views\ folder (or File->Save As three times)



The screenshot shows a code editor with a sidebar on the left displaying a file tree. The tree includes an 'app.d' folder and a 'views' folder containing files: 'about.dt', 'contact.dt', 'events.dt', 'hellogain.dt', and 'hellogain.html'. The 'about.dt' file is currently selected and its content is visible in the main pane. The content of 'about.dt' is a Diet template with the following structure:

```
10    li <a href="events.dt">Events</a>
11    li <a href="contact.dt">Contact</a>
12    article
13        h1 Welcome to our not so simple site!
14        blockquote
15            | Lorem ipsum dolor sit amet
16            | Ut enim ad minim veniam, quis nostrum exercitationem ullam
17            | corporis suscipit laboriosam, nisi ut aliquid ex ea commodi
18            | consequatur. Quis aute iure reprehenderit in voluptate velit
19            | esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
20            | obcaecat cupiditat non proident, sunt in culpa qui officia
21            | deserunt mollit anim id est laborum.
```

And make some changes to the three new files so they display a different message.

views\about.dt

```
html
  head
    title Demo site
  body
    header
      h2 Welcome to our not so simple site
    nav
      ul
        li <a href="/">Home</a>
        li <a href="/about">About us</a>
        li <a href="/events">Events</a>
        li <a href="/contact">Contact us</a>
    article
      h1 This is the About us page!
    footer
      p Copyright 2022 Notsosimple Company
```

views\contact.dt

```
html
  head
    title Demo site
  body
    header
      h2 Welcome to our not so simple site
    nav
      ul
        li <a href="/">Home</a>
        li <a href="/about">About us</a>
        li <a href="/events">Events</a>
        li <a href="/contact">Contact us</a>
```

```
article
  h1 This is the Contact us page!
footer
  p Copyright 2022 Notsosimple Company
```

views\events.dt

```
html
  head
    title Demo site
  body
    header
      h2 Welcome to our not so simple site
    nav
      ul
        li <a href="/">Home</a>
        li <a href="/about">About us</a>
        li <a href="/events">Events</a>
        li <a href="/contact">Contact us</a>
    article
      h1 This is the Events page!
    footer
      p Copyright 2022 Notsosimple Company
```

Next, edit source\app.d to use views\notsimplehello.dt

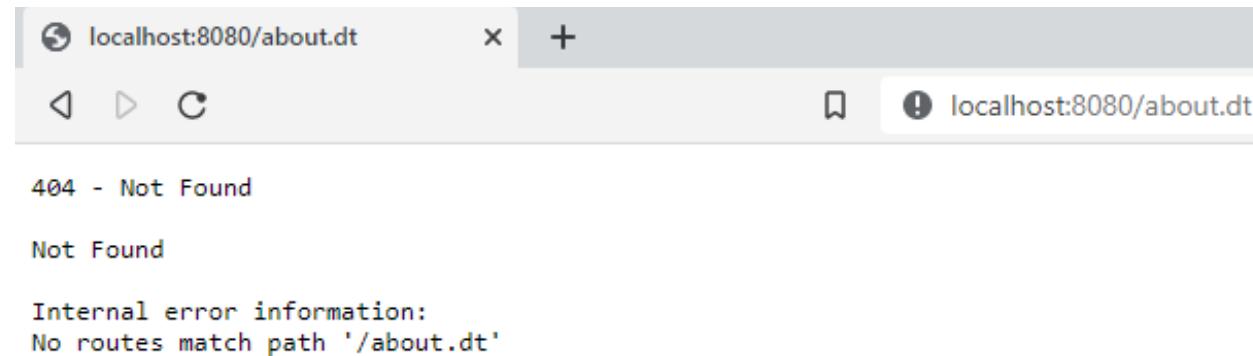
```
import vibe.vibe;

void main()
{
  auto settings = new HTTPServerSettings;
  settings.port = 8080;
```

```
settings.bindAddresses = [":1", "127.0.0.1"];\n\nauto router = new URLRouter;\nrouter.get("/", staticTemplate!"notsimplehello.dt");\nrouter.get("*", serveStaticFiles("public/")); \n\nauto listener = listenHTTP(settings, router);\nscope (exit) listener.stopListening();\nrunApplication();\n}
```

Then compile, run and refresh your browser. If you see no changes, that means you did good.

But when you click on the links, you get an error message like this:



That's because we did not indicate what page to load when the user clicks on a link. We need the `router` to display a certain page when given a certain URL. This matching of URLs to pages or actions is called *routing*.

Edit source\app.d to make the server display the appropriate page when clicking on a link.

```
import vibe.vibe;
```

```
void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];

    auto router = new URLRouter;
    router.get("/", staticTemplate!"notsimplehello.dt");
    router.get("/about", staticTemplate!"about.dt");
    router.get("/contact", staticTemplate!"contact.dt");
    router.get("/events", staticTemplate!"events.dt");
    router.get("*", serveStaticFiles("public/"));

    auto listener = listenHTTP(settings, router);
    scope (exit) listener.stopListening();
    runApplication();
}
```

Then compile, run and refresh your browser. Now the links should work.



## Welcome to our not so simple site

- [Home](#)
- [About us](#)
- [Events](#)
- [Contact us](#)

Welcome to our not so simple site!

The About us page:

A screenshot of a web browser window titled "Demo site". The address bar shows "localhost:8080/about". The page content includes a heading "Welcome to our not so simple site", a navigation menu with links to Home, About us, Events, and Contact us, and a message "This is the about us page!". A red oval highlights the URL in the address bar and the message on the page.

Welcome to our not so simple site

- [Home](#)
- [About us](#)
- [Events](#)
- [Contact us](#)

This is the about us page!

Copyright 2022 Notsosimple Company

The Events page:

A screenshot of a web browser window titled "Demo site". The address bar shows "localhost:8080/events". The page content includes a heading "Welcome to our not so simple site", a navigation menu with links to Home, About us, Events, and Contact us, and a message "This is the events page!". A red oval highlights the URL in the address bar and the message on the page.

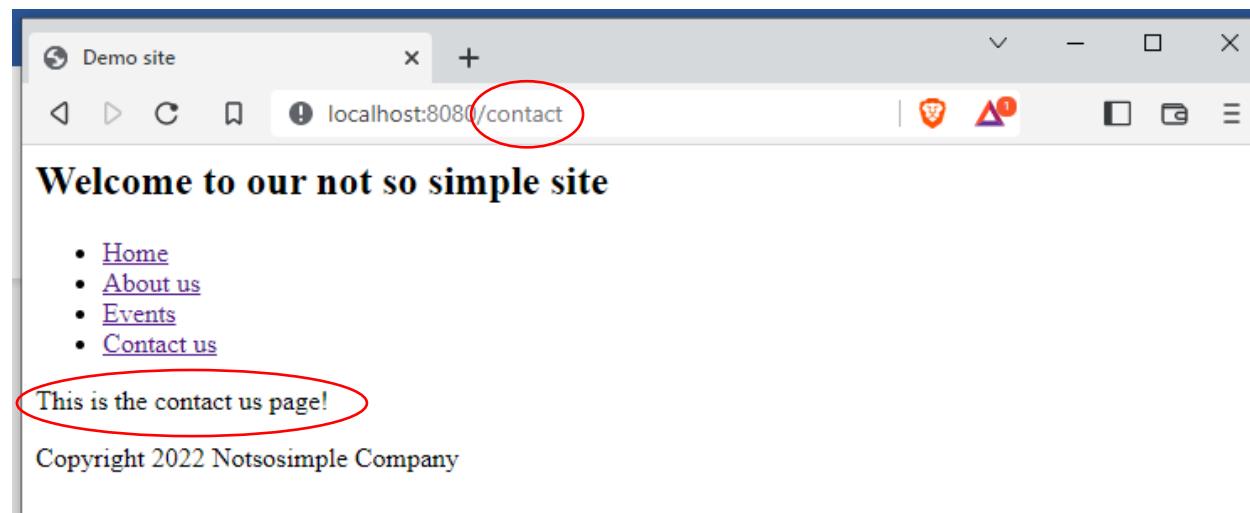
Welcome to our not so simple site

- [Home](#)
- [About us](#)
- [Events](#)
- [Contact us](#)

This is the events page!

Copyright 2022 Notsosimple Company

And the Contact us page:



## Making use of your own functions

You can also make calls to your own functions. Add the following link in notsimplehello.dt

```
html
  head
    title Demo site
  body
    header
      h2 Welcome to our not so simple site
    nav
      ul
        li <a href="/">Home</a>
        li <a href="/about">About us</a>
        li <a href="/events">Events</a>
        li <a href="/contact">Contact us</a>
        li <a href="/helloagain">Hello again!</a>
    article
      h1 Welcome to our not so simple site!
      div.
        Lorem ipsum dolor sit amet, consectetur adipisci elit, sed
        eiusmod tempor incididunt ut labore et dolore magna aliqua.
        Ut enim ad minim veniam, quis nostrum exercitationem ullam
        corporis suscipit laboriosam, nisi ut aliquid ex ea commodi
        consequatur. Quis aute iure reprehenderit in voluptate velit
        esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
        obcaecat cupiditat non proident, sunt in culpa qui officia
        deserunt mollit anim id est laborum.
    footer
      p Copyright 2023 Notsosimple Company
```

and edit app.d to call your own user-created function helloAgain()

```
import vibe.vibe;

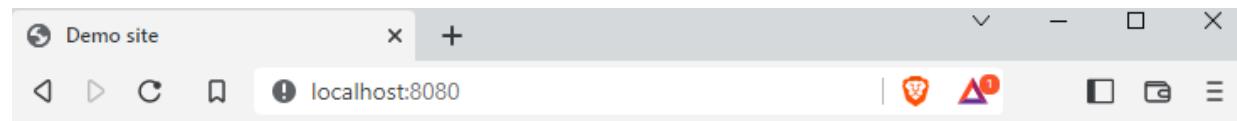
void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];

    auto router = new URLRouter;
    router.get("*", serveStaticFiles("public/"));
    router.get("/", staticTemplate!"notsimplehello.dt");
    router.get("/about", staticTemplate!"about.dt");
    router.get("/contact", staticTemplate!"contact.dt");
    router.get("/events", staticTemplate!"events.dt");
    router.get("/helloagain", &helloAgain);

    auto listener = listenHTTP(settings, router);
    scope (exit) listener.stopListening();
    runApplication();
}

void helloAgain(HTTPServerRequest req, HTTPServerResponse res)
{
    res.writeBody("Hello again, World!");
}
```

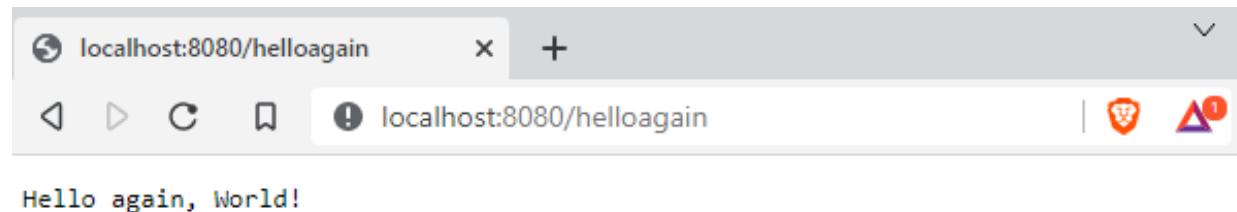
Compile and run then refresh your browser to localhost:8080/



## Welcome to our not so simple site

- [Home](#)
- [About us](#)
- [Events](#)
- [Contact us](#)
- [Hello again!](#)

and click on the Hello again! link and you will see that the new function gets called



## Using templates for ease of maintenance

In the last example, there were some code duplication. Each and every page has the same navigation links as well as a header and a footer. If we want to edit the links, we will have to go through each and every file and edit them. Although there is that super-convenient tool called copy-and-paste, doing manual changes to different files is fraught with errors. Although for now this looks trivial because our files are small, things can get more complicated as we add more files and the files become bigger.

To reduce code duplication and errors in maintenance, Diet templating provides a mechanism using inheritance with the keyword `extends`.

Go to File→Save As... and save views\notsimplehello.dt as views\mainlayout.dt. Then edit views\mainlayout.dt to make it look like this:

```
html
  head
    title Demo site
  body
    header
      h2 Welcome to our not so simple site
    nav
      ul
        li <a href="/">Home</a>
        li <a href="/about">About us</a>
        li <a href="/events">Events</a>
        li <a href="/contact">Contact us</a>
        li <a href="/helloagain">Hello again!</a>
    article
      block content
    footer
      p Copyright 2022 Notsosimple Company
```

Then rename views\notsimplehello.dt into views\index.dt with this contents:

```
extends mainlayout
block content
    h1 Welcome to our not so simple site!
    div.
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
        eiusmod tempor incididunt ut labore et dolore magna aliqua.
        Ut enim ad minim veniam, quis nostrum exercitationem ullam
        corporis suscipit laboriosam, nisi ut aliquid ex ea commodi
        consequatur. Quis aute iure reprehenderit in voluptate velit
        esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
        obcaecat cupiditat non proident, sunt in culpa qui officia
        deserunt mollit anim id est laborum.
```

Edit about.dt with the following contents

```
extends mainlayout
block content
    h1 This is the About us page
```

Edit events.dt with the following contents

```
extends mainlayout
block content
    h1 This is the Events page!
```

Edit contact.dt with the following contents

```
extends mainlayout
block content
    h1 This is the Contact us page!
```

Next, edit source\app.d to make it look like this:

```
import vibe.vibe;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];

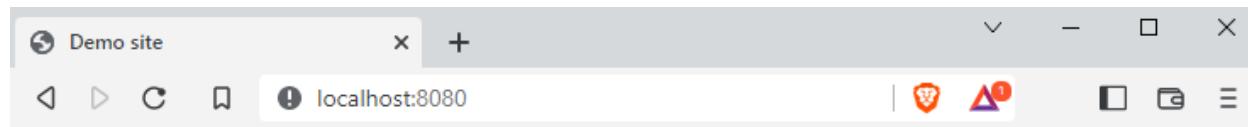
    auto router = new URLRouter;
    router.get("/", staticTemplate!"index.dt");
    router.get("/about", staticTemplate!"about.dt");
    router.get("/contact", staticTemplate!"contact.dt");
    router.get("/events", staticTemplate!"events.dt");
    router.get("/helloagain", &helloAgain);
    router.get("*", serveStaticFiles("public/"));

    auto listener = listenHTTP(settings, router);
    scope (exit) listener.stopListening();
    runApplication();
}

void helloAgain(HTTPServerRequest req, HTTPServerResponse res)
{
    res.writeBody("Hello again, World!");
}
```

Compile, run and refresh your browser.

If you find that nothing has changed in the browser, you did good.



## Welcome to our not so simple site

- [Home](#)
- [About us](#)
- [Events](#)
- [Contact us](#)
- [Hello again!](#)

By creating a layout template, we placed the repeating tags in the layout template (which we called mainlayout.dt) while leaving the other templates with only their specific contents. We named the part that changes as the content block and we indicated in the other pages that their contents will appear in that content block of the mother layout.

In the block content statement, we have named that part, or 'block', of the layout as 'content'. You can give the block any name as long as you give the same name to the other templates that inherit from that template.

## Using include in templates

Oftentimes we need to break up our layout into smaller chunks of code so that we can just include them to compose a page. The include directive is just for that purpose.

Create views\header.dt and extract from views\mainlayout.dt this content:

```
h2 Welcome to our not so simple site
```

Then create views\navbar.dt and extract from views\mainlayout.dt this content:

```
ul
li <a href="/">Home</a>
li <a href="/about">About us</a>
li <a href="/events">Events</a>
li <a href="/contact">Contact us</a>
li <a href="/helloagain">Hello again!</a>
```

Then create views\footer.dt and extract from views\mainlayout.dt with the following contents:

```
p Copyright 2022 Notsosimple Company
```

That leaves views\mainlayout.dt with this:

```
html
  head
    title Demo site
  body
    header
      include header.dt
    nav
      include navbar.dt
```

```
article
  block content
footer
  include footer.dt
```

Compile, run and refresh the browser. If you see no changes, you did good again.

## Layout without Bootstrap using CSS grid

CSS grid was designed to make it easier to lay out components on a page using standard CSS. It divides the page into a grid composed of rows and columns, like a table, which is similar to Bootstrap's row- and col- classes.

As an example, let's create an HTML page *but we are not going to compile*.

Let us create views\cssgrid.html:

```
<html>
  <head>
    <title>CSS Grid Sample</title>
    <style>
      .container
      {
        margin: 0 auto;
        width: 100%;
        height: 100%;
        display: grid;
        grid-template-columns: 150px 150px 150px;
        grid-template-rows: 200px 200px 200px;
      }
      .entry
      {
        color: white;
        text-align: center;
        display: table-cell;
        vertical-align: middle;
      }
      #first { background-color: red; }
      #second { background-color: green; }
      #third { background-color: blue; }
```

```
#fourth { background-color: teal; }
#fifth { background-color: grey; }
#sixth { background-color: brown; }
#seventh { background-color: silver; }
#eighth { background-color: maroon; }
#ninth { background-color: cyan; }

</style>
</head>
<body>
  <div class="container">
    <div class="entry" id="first">1st</div>
    <div class="entry" id="second">2nd</div>
    <div class="entry" id="third">3rd</div>
    <div class="entry" id="fourth">4th</div>
    <div class="entry" id="fifth">5th</div>
    <div class="entry" id="sixth">6th</div>
    <div class="entry" id="seventh">7th</div>
    <div class="entry" id="eighth">8th</div>
    <div class="entry" id="ninth">9th</div>
  </div>
</body>
</html>
```

The line

```
grid-template-columns: 150px 150px 150px;
```

means we are making three columns, with each column 150 pixels wide.

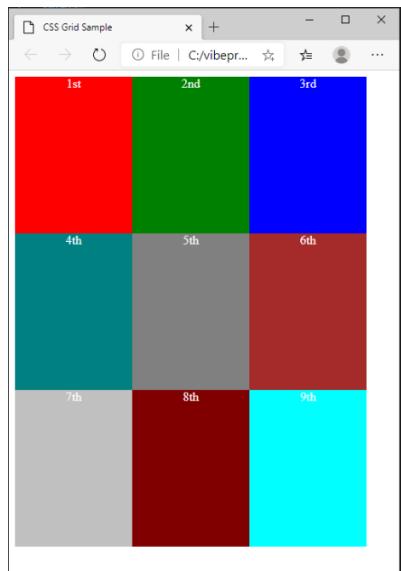
The line

```
grid-template-rows: 200px 200px 200px;
```

means we are declaring three rows, with each row having a height of 200 pixels.

This way, we defined a three-by-three grid, or three rows with three columns.

**Do not compile.** Simply open it in File Explorer and right-click the file to open it in the browser, then resize the browser to make it small:



Then expand the browser:

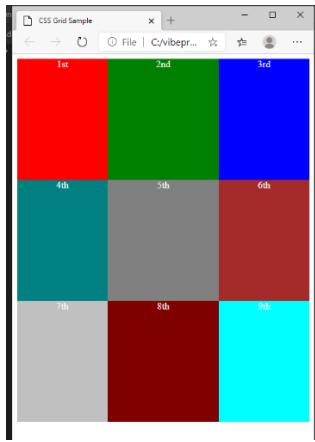


You see that when you resize the browser, the blocks remain the same sizes.

Now let's change the CSS to make the center column expandable.

```
.container
{
    margin: 0 auto;
    width: 100%;
    height: 100%;
    display: grid;
    grid-template-columns: 150px auto 150px;
    grid-template-rows: 200px 200px 200px;
}
```

Refresh the browser. This time, when you expand or contract the browser, the middle column resizes automatically.

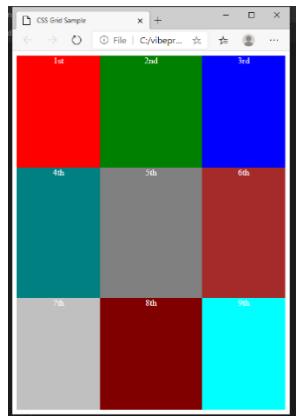


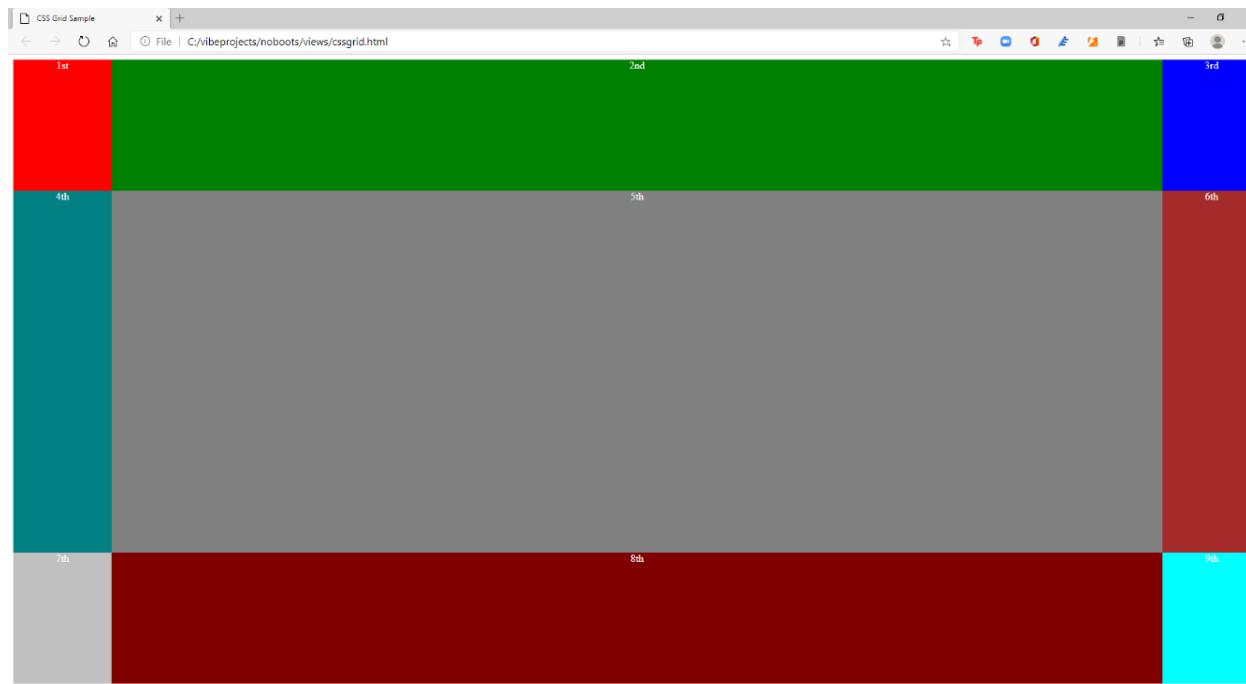
Now let's make the middle row expandable too.

```
.container
{
    margin: 0 auto;
```

```
width: 100%;  
height: 100%;  
display: grid;  
grid-template-columns: 150px auto 150px;  
grid-template-rows: 200px auto 200px;  
}
```

And resize the browser.





Now both the center column and the middle row are expandable.

We are just skimming the surface of CSS Grid here. There are so many variations in the uses of CSS grid, but for now let us settle for this rudimentary knowledge. At least we are armed with the knowledge to develop the fixed navbar and the sticky footer without Bootstrap.

## A fixed navbar and a sticky footer without Bootstrap

It is nice to work with Bootstrap. CSS seems less intimidating and more approachable, and they also keep improving Bootstrap every year. However, there are developments in standard CSS which we tend to ignore because of the familiarity and convenience of Bootstrap. And some dev outfits might not want its developers to use Bootstrap, so it's always a good practice to learn alternatives.

Let's create a new project with a fixed navbar and a sticky footer.

On a terminal window, create a project named no\_bs.

Stop the server if it is still running.

```
[00000000(----) INF] Received signal 2. Shutting down.  
[main(----) INF] Stopped to listen for HTTP requests on ::1:8080  
[main(----) INF] Stopped to listen for HTTP requests on 127.0.0.1^C:8080  
Warning  
: c:\vibeprojects\hello>3 socket handles leaked at driver shutdown.  
Warning: 3 socket handles leaked at driver shutdown.
```

```
c:\vibeprojects\hello>cd ..
```

```
c:\vibeprojects>dub init no_bs -t vibe.d  
Package recipe format (sdl/json) [json]:  
Name [no_bs]:  
Description [A simple vibe.d server application.]:  
Author name [Owner]:  
License [proprietary]:  
Copyright string [Copyright T- 2023, Owner]:  
Add dependency (leave empty to skip) []:  
Success created empty project in c:\vibeprojects\no_bs
```

Package successfully created in no\_bs

```
c:\vibeprojects>cd no_bs
```

```
c:\vibeprojects\no_bs>
```

Open the no\_bs folder in VS Code and edit source\app.d to make it look like this:

```
import vibe.vibe;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];

    auto router = new URLRouter;
    router.get("*", serveStaticFiles("public/"));
    router.get("/", staticTemplate!"index.dt");
    router.get("/about", staticTemplate!"about.dt");
    router.get("/contact", staticTemplate!"contact.dt");
    router.get("/login", staticTemplate!"login.dt");

    auto listener = listenHTTP(settings, router);
    scope (exit) listener.stopListening();

    runApplication();
}
```

Next, create views\layout.dt.

```
doctype 5
```

```
html
  head
    title Employee Timekeeping System
    include styles.dt
  body
    div.container
      div.menu
        include menu.dt
      div.content
        block maincontent
      div.footer
        include footer.dt
```

Then create views\styles.dt.

```
:css
body
{
  margin: 0;
  padding: 0;
}
.container
{
  display: grid;
  grid-template-rows: 40px auto 30px;
  height: 100%;
  width: 100%;
  margin: 0;
  padding: 0;
}
.menu
{
```

```
position: fixed;
top: 0;
width: 100%;
padding: 10px 0;
background-color: whitesmoke;
}

.menu-item
{
    display: inline;
}

.item-link
{
    margin-left: 30px;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: 18px;
    color: teal;
    text-decoration: none;
}

.item-link-right
{
    float: right;
    margin-right: 30px;
}

.modal-form
{
    position: fixed;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    background: rgba(0,0,0,0.5);
```

```
z-index: 99999;
opacity: 0;
pointer-events: none;
}
#login:target, #find_employee:target
{
    opacity: 1;
    pointer-events: auto;
}
.modal-form-grid
{
    display: grid;
    grid-template: 30px 30px 30px / 1fr 1fr;
    grid-gap: 10px;
}
.text-control
{
    grid-column: 1 / 3;
}
#but-reset
{
    grid-column: 1 / 2;
}
#but-submit
{
    grid-column: 2 / 3;
}
.modal-form-wrapper-login
{
    width: 250px;
    position: absolute;
    right: 0;
```

```
margin-top: 40px;
padding: 25px 20px;
border-radius: 10px;
text-align: center;
background-color: whitesmoke;
}
.modal-form-wrapper-find_employee
{
width: 250px;
position: relative;
left: 280px;
margin-top: 40px;
padding: 25px 20px;
border-radius: 10px;
text-align: center;
background-color: whitesmoke;
}
.content
{
padding: 40px 30px;
}
.footer
{
position: fixed;
bottom: 0;
width: 100%;
background-color: whitesmoke;
padding: 5px 0;
}
.copyright
{
font-family: Verdana, Geneva, Tahoma, sans-serif;
```

```
    font-size: 14px;
    text-align: center;
    color: teal;
}
.form-hidden
{
    padding: 0;
    margin: 0;
    display: inline;
}
```

Then create views\menu.dt.

```
div.menu-item
    a.item-link(href="/") Home
div.menu-item
    a.item-link(href="about") About us
div.menu-item
    a.item-link(href="contact") Contact us
div.menu-item
    a.item-link.item-link-right(href="login") Login
```

Then create views\index.dt.

```
extends layout
block maincontent
    h2 The Lorem Ipsum Company
    div.
        Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Nam nec urna arcu. Quisque eleifend posuere vestibulum.
        In sed magna mauris. Phasellus bibendum ligula et placerat
        vulputate. In non suscipit lectus, a laoreet odio. Donec
        at sapien eu nisi porta condimentum. Morbi non varius ex,
```

```
nec luctus nisl. Aenean varius dui quis arcu auctor luctus.  
Integer efficitur ornare massa, ac suscipit enim sagittis et.  
Proin vestibulum tellus in ipsum ultrices, sed imperdiet  
sapien euismod. Praesent vel facilisis mauris. Proin finibus  
congue tellus, non varius ante. Nullam tincidunt dolor felis.  
Pellentesque non luctus tellus. Curabitur et sapien at justo  
fringilla feugiat et a erat.  
<br /><br />
```

And make stub pages for views\about.dt,

```
extends layout  
block maincontent  
    h2 The About Us page  
    div.  
        <h3>This is the About Us page.</h3>
```

and views\contact.dt,

```
extends layout  
block maincontent  
    h2 The Contact Us page  
    div.  
        <h3>This is the Contact Us page.</h3>
```

and views\login.dt.

```
extends layout  
block maincontent  
    h2 The Login page  
    div.  
        <h3>This is the Login page.</h3>
```

After you compile and run, you should be able to see these pages:

A screenshot of a web browser window titled "Employee Timekeeping System". The address bar shows "localhost:8080". The page content includes a navigation bar with "Home", "About us", "Contact us", and "Login" links. Below the navigation bar is the heading "The Lorem Ipsum Company" followed by a large amount of placeholder text (Lorem ipsum). At the bottom of the page is a copyright notice: "Copyright © The Lorem Ipsum Company 2023".

## The Lorem Ipsum Company

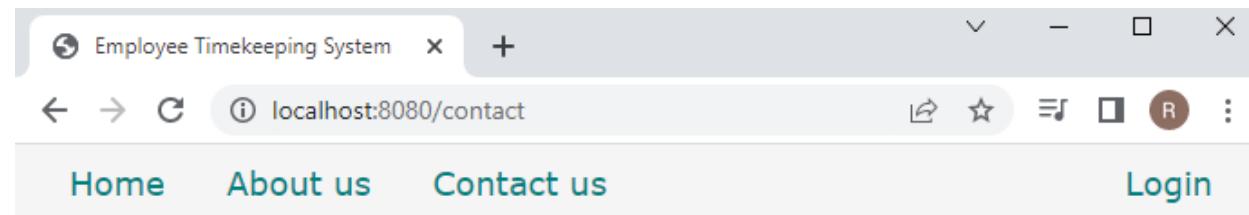
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam nec urna arcu. Quisque eleifend posuere vestibulum. In sed magna mauris. Phasellus bibendum ligula et placerat vulputate. In non suscipit lectus, a laoreet odio. Donec at sapien eu nisi porta condimentum. Morbi non varius ex, nec luctus nisl. Aenean varius dui quis arcu auctor luctus. Integer efficitur ornare massa, ac suscipit enim sagittis et. Proin vestibulum tellus in ipsum ultrices, sed imperdiet sapien euismod. Praesent vel facilisis mauris. Proin finibus congue tellus, non varius ante. Nullam tincidunt dolor felis. Pellentesque non luctus tellus. Curabitur et sapien at justo fringilla feugiat et a erat.

Copyright © The Lorem Ipsum Company 2023

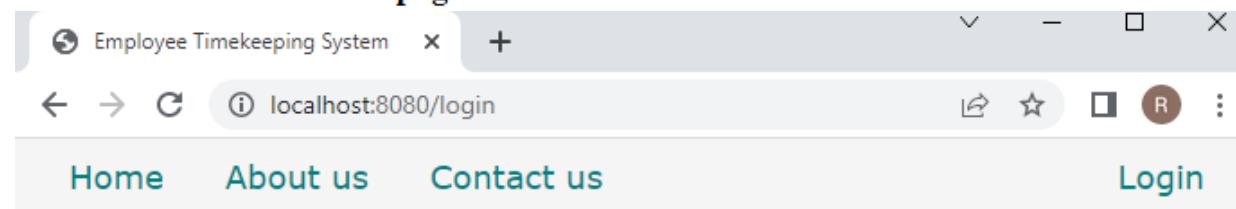
A screenshot of a web browser window titled "Employee Timekeeping System". The address bar shows "localhost:8080/about". The page content includes a navigation bar with "Home", "About us", "Contact us", and "Login" links. Below the navigation bar is the heading "The About Us page" followed by the text "This is the About Us page."

## The About Us page

This is the About Us page.



**This is the Contact Us page.**



**The Login page**

**This is the Login page.**

You just created a layout with a fixed navbar and a sticky footer without using Bootstrap. If you expand and contract the browser, you will see that the middle row expands and contracts accordingly, making the pages responsive.

In the `views\styles.dt` file, we declared this:

```
.container
{
  display: grid;
  grid-template-rows: 40px auto 30px;
  height: 100%;
  width: 100%;
  margin: 0;
```

```
padding: 0;  
}
```

We did not declare any columns, effectively making the whole page just one column with three rows. The menu bar is on the top row and the footer is on the bottom row, with the middle row expanding and contracting to fit the page height when the browser is resized, thus making it responsive.

With CSS Grid, you will notice that the layout terms are declared in the CSS file so the HTML file is cleaner compared to Bootstrap.

How about a modal dialogue without JavaScript?

## Adding a modal dialogue to the menu

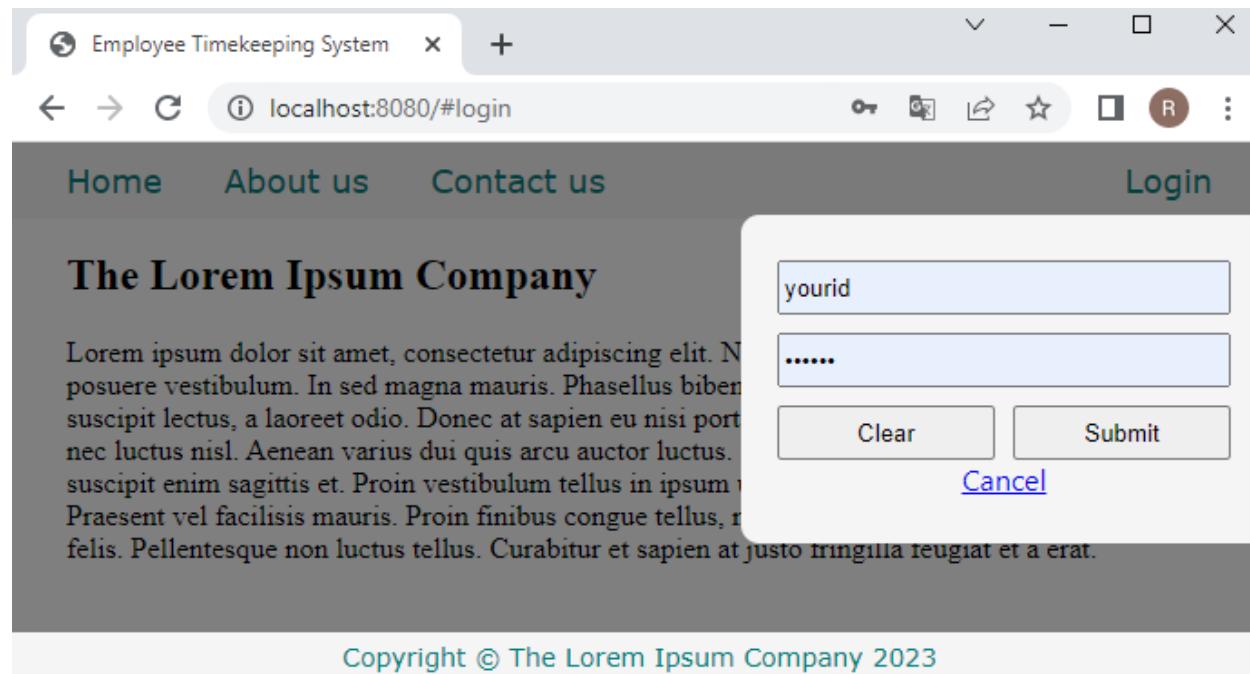
In CSS3, modal dialogues are actually part of the page but is hidden, then becomes visible and takes the focus after an event, like a click.

Let us implement the Login link with a modal form.

Edit views\menu.dt to add the modal form activation at the bottom:

```
div.menu-item
  a.item-link(href="/") Home
div.menu-item
  a.item-link(href="about") About us
div.menu-item
  a.item-link(href="contact") Contact us
div.menu-item
  a.item-link.item-link-right(href="#login") Login
div#login.modal-form
  div.modal-form-wrapper-login
    form.modal-form-grid(method="post", action="login")
      input.text-control(name="email", type="email", placeholder=" email address")
      input.text-control(name="password", type="password", placeholder=" password")
      input#but-reset(type="reset", value="Clear")
      input#but-submit(type="submit", value="Submit")
      a.close(href="#close") Cancel
```

We made a div with an id of “#login” so the link can point to it. After you compile and run, click on the Login link and see the modal form:



Let us edit the menu to make it more relevant to our final project.

Edit `views\menu.dt` to make it look like this:

```
div.menu-item
  a.item-link(href="/") Home
div.menu-item
  a.item-link(href="all_employees") All employees
div.menu-item
  a.item-link(href="#find_employee") Find employee
div.menu-item
  a.item-link(href="add_employee") Add employee
div.menu-item
  a.item-link.item-link-right(href="#login") Login
```

```


#login.modal-form
  div.modal-form-wrapper-login
    form.modal-form-grid(method="post", action="login")
      input.text-control(name="email", type="email", placeholder=" email address")
      input.text-control(name="password", type="password", placeholder=" password")
      input#but-reset(type="reset", value="Clear")
      input#but-submit(type="submit", value="Submit")
    a.close(href="#close") Cancel



Compile and run, then refresh the browser. Click on the Find employee link.



The screenshot shows a web page with a navigation bar at the top containing links for Home, All employees, Find employee, Add employee, and Login. Below the navigation bar, there is a heading "The Lorem Ipsum Comp..." followed by a large amount of placeholder text from the Lorem Ipsum generator. A modal dialog is overlaid on the page, centered over the "Find employee" link. The modal has a light gray background and contains the following elements:



- A title area with the truncated heading "The Lorem Ipsum Comp...".
- Two input fields stacked vertically, both labeled with placeholder text: "First name" and "Last name".
- Below the input fields are three buttons: "Clear", "Submit", and "Cancel".
- The "Cancel" button is positioned directly below the "Submit" button.


```

So now we have a menu system closer to what we need for our project.

## The web framework

The web framework makes routing simpler and more intuitive as it combines business logic and displaying web pages easily. Instead of declaring all the routes in the router itself, the web framework makes use of conventions based on REST (Representational State Transfer) to match URLs to actions or routes combined with the business logic.

The five actions / requests in REST are

GET - retrieve something from the server  
POST - create something new in the server  
PUT - update fully something in the server  
DELETE – delete something from the server  
PATCH – update partially something in the server

For our purposes, we will focus on the GET and POST operations. We will use GET for simple retrieval operations and use POST for the rest. For example, if the request is only asking for a page to be displayed, we usually use the GET method. If the request is asking for data to be added, retrieved, modified or deleted, we will use the POST method.

When a form is submitted, a GET method shows the input variables of the form at the end of the URL, called the query string. With a POST method, they are not displayed. Also, a query string has a limit in length while a POST method has no limit.

## A new project

We don't want to change or erase what we have done so far so we can preserve this point in time and can go back to this state at any time, so we are going to create a new project and simply copy all the \views\\* files in this project to that new project.

Create a new project named empapp.

Press Ctrl-C to stop the running program (twice if needed).

```
[main(----) INF] Listening for requests on http://[::1]:8080/  
[main(----) INF] Listening for requests on http://127.0.0.1:8080/  
[00000000(----) INF] Received signal 2. Shutting down.  
Warnin^g: C6 socket ha  
ndc:\vibeprojects\noboots>les leaked at driver shutdown.  
Warning: 6 socket handles leaked at driver shutdown.
```

```
c:\vibeprojects\no_bs>cd ..
```

```
c:\vibeprojects>dub init empapp -t vibe.d  
Package recipe format (sdl/json) [json]:  
Name [empapp]:  
Description [A simple vibe.d server application.]:  
Author name [Owner]:  
License [proprietary]:  
Copyright string [Copyright T- 2023, Owner]:  
Add dependency (leave empty to skip) []:  
Success created empty project in c:\vibeprojects\empapp  
    Package successfully created in empapp
```

```
c:\vibeprojects>cd empapp
```

```
c:\vibeprojects\empapp>
```

Simply copy the files from no\_bs\views\ into the current project empapp\views so we don't have to create them again.

Open the new empapp folder in VS Code and edit source\app.d to make use of the web framework.

```
import vibe.vibe;
import empcontrol;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];

    auto router = new URLRouter;
    router.get("*", serveStaticFiles("public/"));
    router.registerWebInterface(new EmployeeController);

    auto listener = listenHTTP(settings, router);
    scope (exit) listener.stopListening();

    runApplication();
}
```

The line

```
    router.registerWebInterface(new EmployeeController);
```

brings in the web framework into the class EmployeeController, which we have not created yet.

Let's rectify that. We indicated that we are importing the empcontrol module, so let's create that file.

Create source\empcontrol.d with this contents:

```
module empcontrol;

import vibe.vibe;

class EmployeeController
{
    void index()
    {
        render!"index.dt";
    }

    void getAllEmployees()
    {
        response.writeBody("This is all_employees!");
    }

    void getFindEmployee()
    {
        response.writeBody("This is find_employee!");
    }

    void getAddEmployee()
    {
        response.writeBody("This is add_employee!");
    }

    void postLogin()
    {
        response.writeBody("This is login!");
    }
}
```

```
}
```

So instead of creating scattered stand-alone functions, we can write them all inside one class as its methods. We don't have to state the `HTTPServerRequest` and the `HTTPServerResponse` objects as arguments for each and every method we define in this class because they are available to us as the variables `request` and `response`.

The `index()` method corresponds to the root URL

`http://localhost:8080/`

The `getFindEmployee()` method will correspond to the link `find_employee` with the URL

`http://localhost:8080/find_employee`

So if we make a `postLogin()` method, it will correspond to the URL extracted from the form

```
form.modal-form-grid(method="post", action="login")
```

the `HTTPMethod` of which is POST.

The methods we defined in the `EmployeeController` class (you can give the class any name), are merely stubs for now just to demonstrate the use of the web framework. We will remove them later.

The web framework has some conventions.

The `index()` method corresponds to the root URL, “`/`”.

The `getAllEmployees()` method corresponds to the “`/all_employees`” link, the `HTTPMethod` of which is GET.

The `getFindEmployee()` method corresponds to the “`/find_employee`” link, whose `HTTPMethod` is also GET.

The methods we defined in the `EmployeeController` class, except for the `index()` method, are merely stubs to demonstrate the use of the web interface.

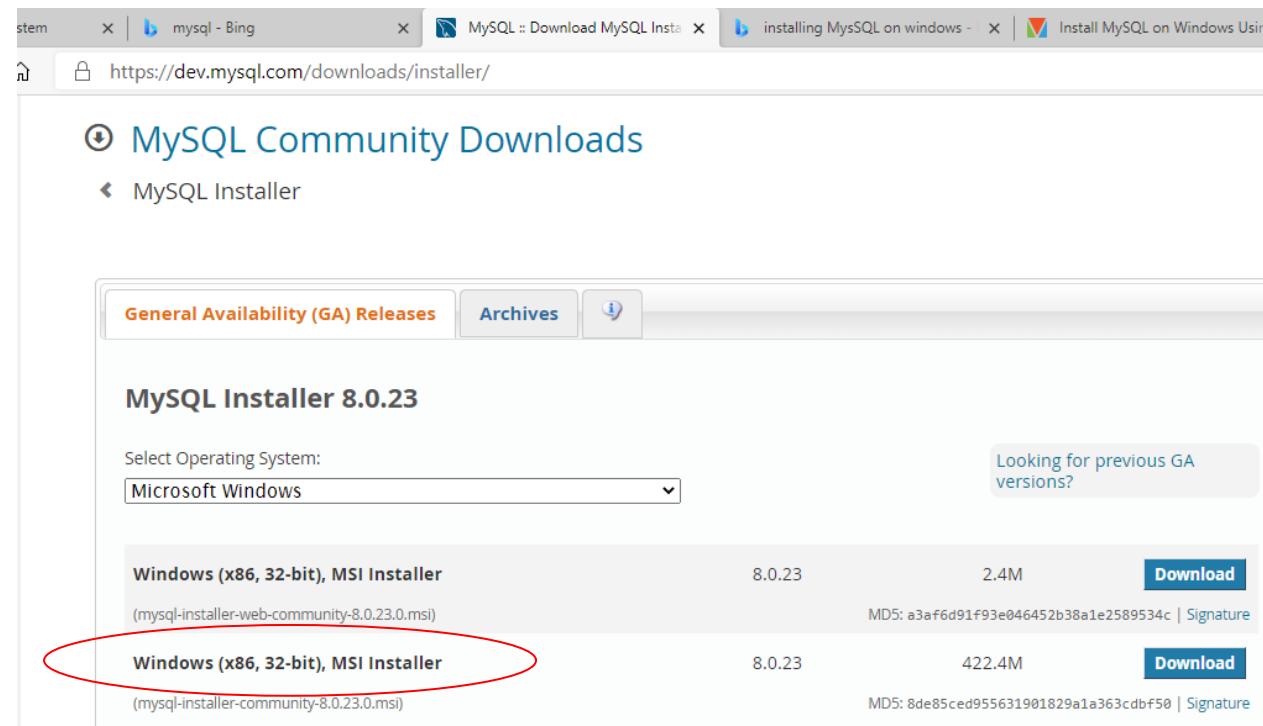
Using the web framework provides a way to combine business logic with routing, so developing a full application is easier. You can mix business logic with displaying a page, such as accessing a database to display a list of employees. That is what we will do next, but first we have to set up our data source.

## Setting up the MySQL server and tools

MySQL must be the most popular DBMS on the planet, although Gartner and other serious-looking humongous entities would like to disagree. Most sysadmins will also disagree since they are sure it is PostGreSQL, but the sheer number of small businesses that use MySQL and do not care to report it belies their claims. After MySQL's acquisition by Oracle, MariaDB was created and claims to be (almost) 100% compatible with MySQL. But let's not explore that path for now.

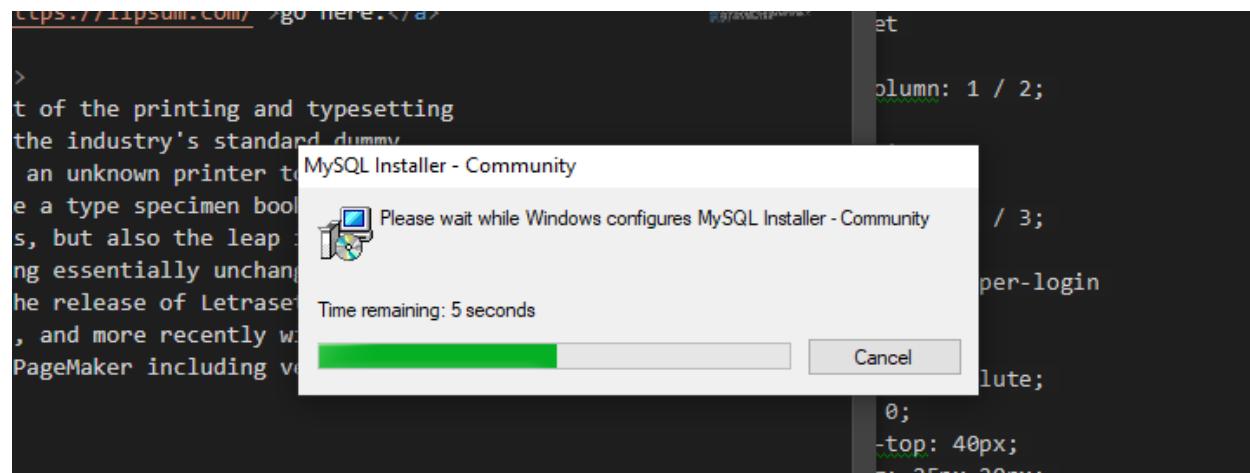
Download the MySQL community server here:

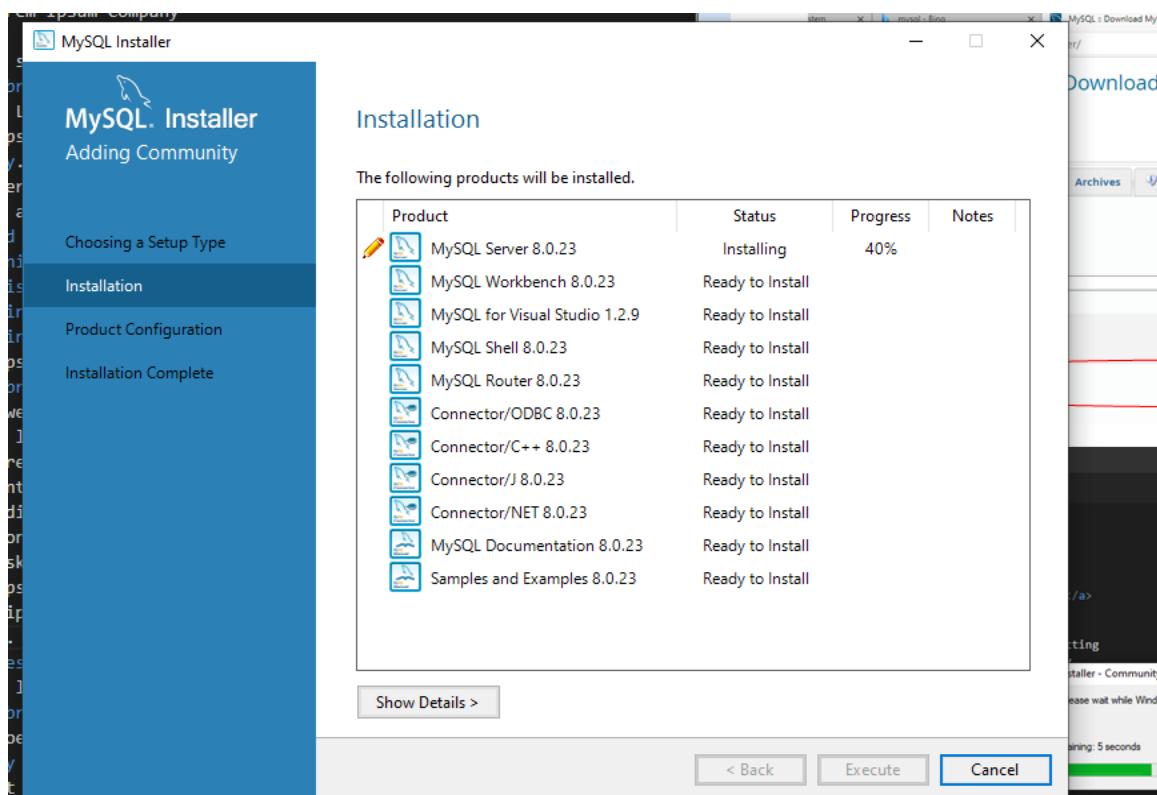
[MySQL :: Download MySQL Installer](https://dev.mysql.com/downloads/installer/)

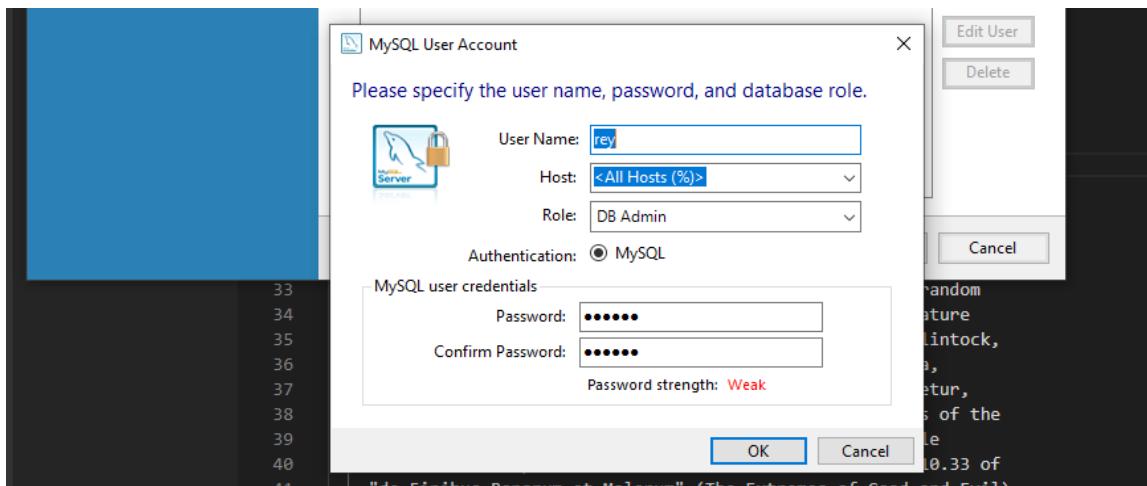


The screenshot shows a web browser window with several tabs open. The active tab is titled "MySQL :: Download MySQL Installer". The URL in the address bar is <https://dev.mysql.com/downloads/installer/>. The page content is titled "MySQL Community Downloads" and shows the "MySQL Installer 8.0.23" section. It includes a dropdown menu for "Select Operating System" set to "Microsoft Windows". Two download links are listed:

Download Link	Version	File Size	Action
<a href="#">Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.23.0.msi)</a>	8.0.23	2.4M	<a href="#">Download</a>
<a href="#">Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.23.0.msi)</a>	8.0.23	422.4M	<a href="#">Download</a>







**MySQL® Installer**  
MySQL Server 8.0.23

- Type and Networking
- Authentication Method
- Accounts and Roles
- Windows Service**
- Apply Configuration

### Windows Service

Configure MySQL Server as a Windows Service

#### Windows Service Details

Please specify a Windows Service name to be used for this MySQL Server instance.  
A unique name is required for each instance.

Windows Service Name:

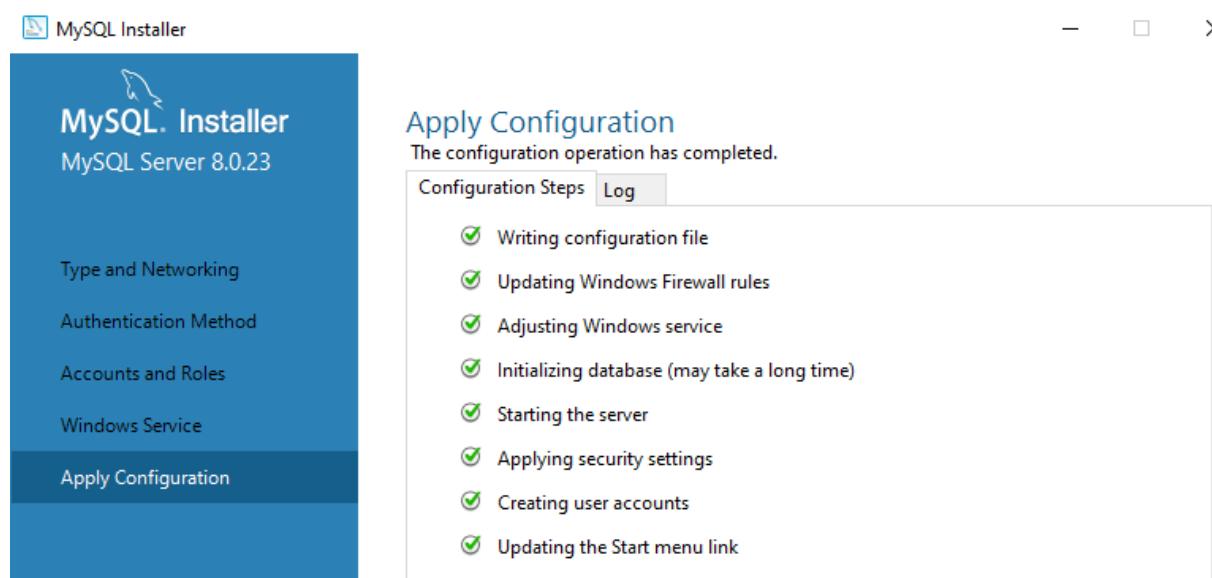
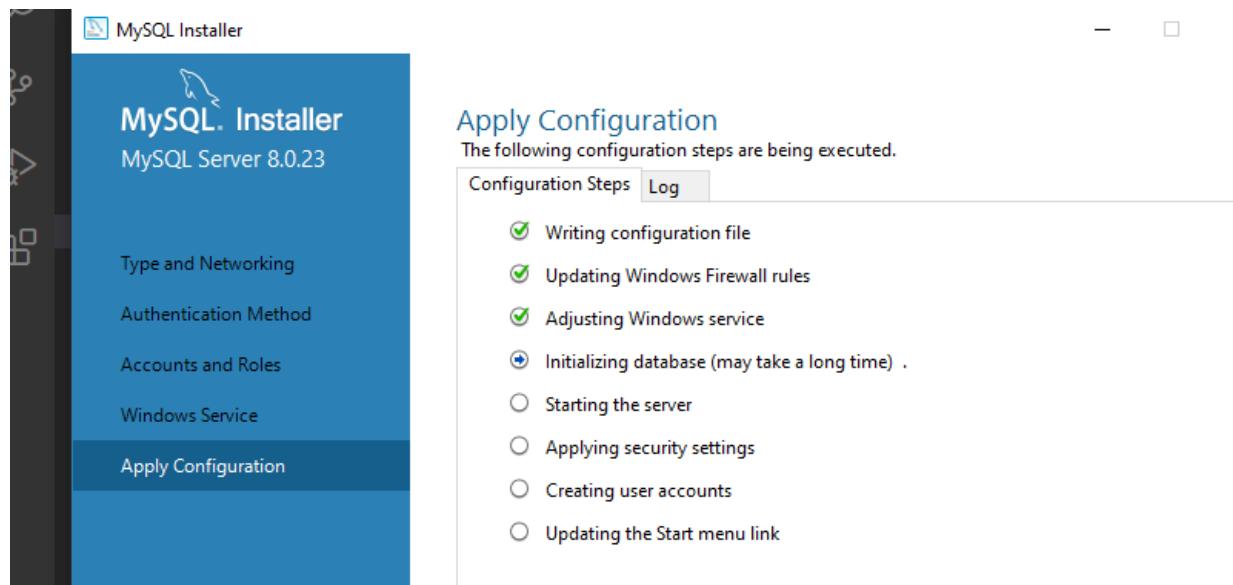
Start the MySQL Server at System Startup

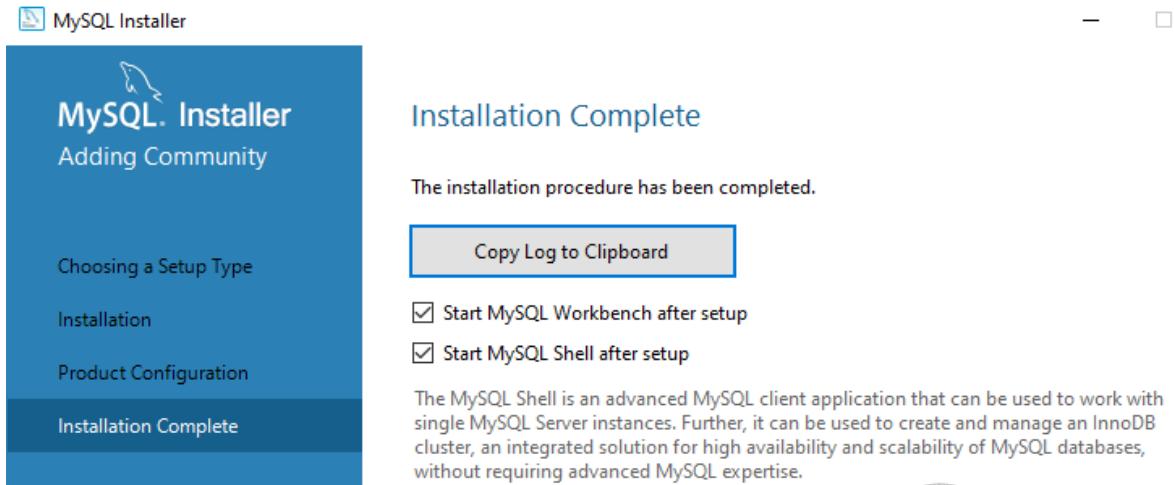
#### Run Windows Service as ...

The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

Standard System Account  
Recommended for most scenarios.

Custom User  
An existing user account can be selected for advanced scenarios.





The installation procedure has been completed.

[Copy Log to Clipboard](#)

Start MySQL Workbench after setup

Start MySQL Shell after setup

The MySQL Shell is an advanced MySQL client application that can be used to work with single MySQL Server instances. Further, it can be used to create and manage an InnoDB cluster, an integrated solution for high availability and scalability of MySQL databases, without requiring advanced MySQL expertise.

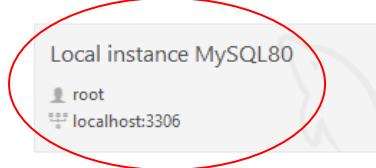
# Welcome to MySQL

MySQL Workbench is the official graphical user interface (GUI) tool for creating and browsing your database schemas, working with database objects, designing and running SQL queries to work with stored data. You can also manage database vendors to your MySQL data

[Browse Documentation >](#)

[Read the Blog >](#)

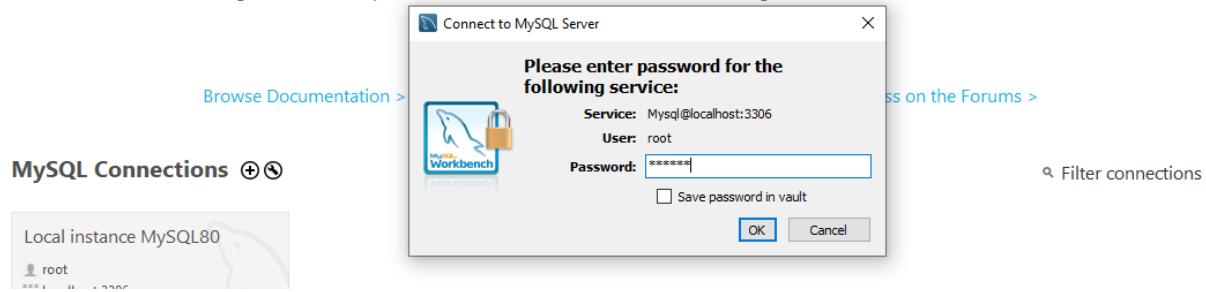
## MySQL Connections



Click on the Local instance MySQL80 shaded area to open the password dialog.

# Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other



Type the root password you set earlier and click OK. The MySQL Workbench front page opens.

Now let's talk about the schema we are going to use.

## The schema

At first, our app is going to be a simple employee records maintenance system of the imaginary Lorem Ipsum Company, where user admins can view, add, edit and delete employee records. These should cover the basic operations done on database tables and could easily be adapted and extended for use on products or services instead of personnel. But later, we will extend the project into a timekeeping system similar to a bundy clock system where employees clock in and clock out.

The database is named empdb. For now, the two tables we will create inside empdb will be the admins and employees tables.

The employees table – containing the employee records:

id	- the row id, the primary key incremented automatically, integer
empid	- the employee id number, string
dept	- department name, string
paygd	- salary grade, string
email	- email address, will also be used as username, string
pword	- password, string
fname	- firstname, string
lname	- lastname, string
phone	- phone number, string
photo	- full path to the employee photo file, string
street	- street address minus the city, string
city	- the city part of the address, string
province	- the province part of the address, string
postcode	- postal code of the address, string

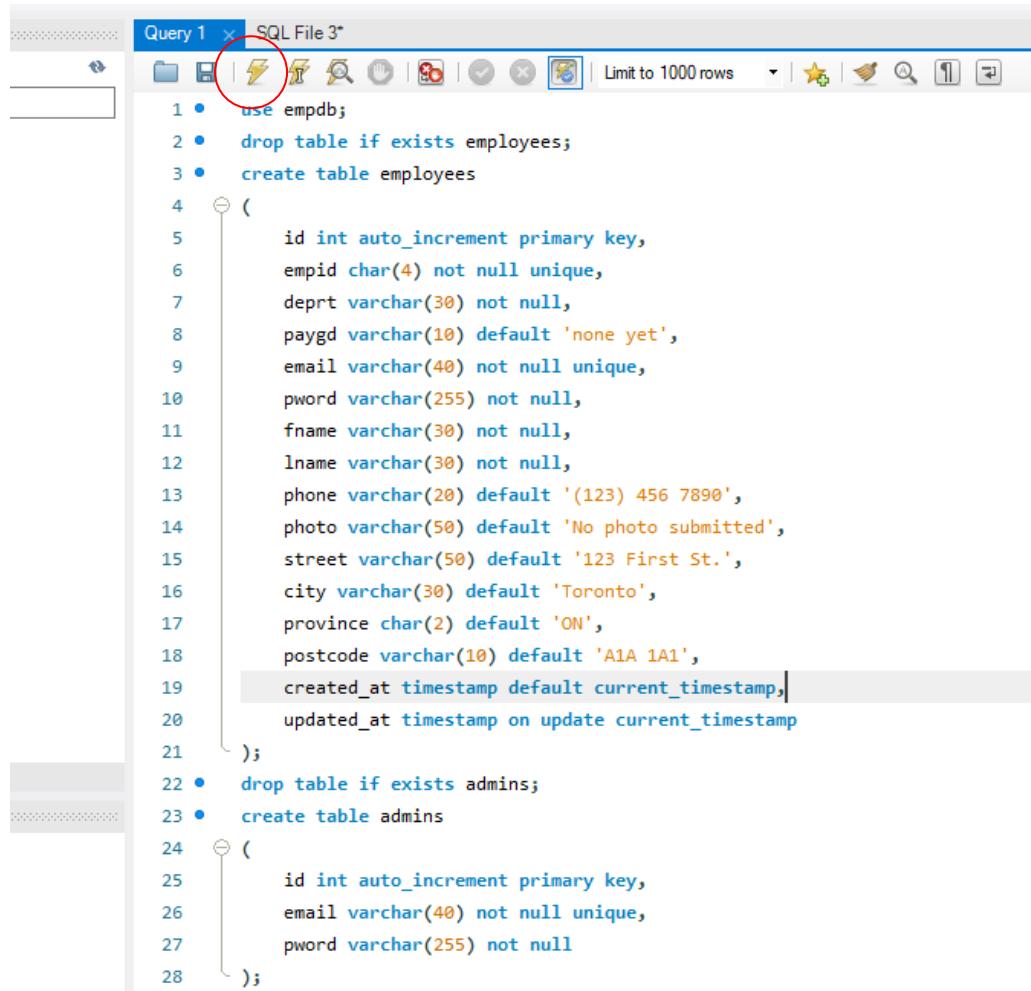
The admins table - the administrators of the employee records:

id	- the row id, the primary key incremented automatically, integer
email	- email address, which will serve as username, string
pword	- password, string

So here is the SQL we are going to write in the MySQL Workbench SQL script window:

```
drop database if exists empdb;
create database empdb;
use empdb;
drop table if exists employees;
create table employees
(
    id int auto_increment primary key,
    empid char(4) not null unique,
    dept varchar(30) not null,
    paygd varchar(10) default 'none yet',
    email varchar(40) not null unique,
    pword varchar(255) not null,
    fname varchar(30) not null,
    lname varchar(30) not null,
    phone varchar(20) default '(123) 456 7890',
    photo varchar(50) default 'No photo submitted',
    street varchar(50) default '123 First St.',
    city varchar(30) default 'Toronto',
    province char(2) default 'ON',
    postcode varchar(10) default 'A1A 1A1',
    created_at timestamp default current_timestamp,
    updated_at timestamp on update current_timestamp
);
drop table if exists admins;
create table admins
(
    id int auto_increment primary key,
    email varchar(40) not null unique,
    pword varchar(255) not null
);
```

After typing in these SQL statements, click on the lightning icon to execute.



```
Query 1 x SQL File 3*
use empdb;
drop table if exists employees;
create table employees
(
    id int auto_increment primary key,
    empid char(4) not null unique,
    dept varchar(30) not null,
    paygd varchar(10) default 'none yet',
    email varchar(40) not null unique,
    pword varchar(255) not null,
    fname varchar(30) not null,
    lname varchar(30) not null,
    phone varchar(20) default '(123) 456 7890',
    photo varchar(50) default 'No photo submitted',
    street varchar(50) default '123 First St.',
    city varchar(30) default 'Toronto',
    province char(2) default 'ON',
    postcode varchar(10) default 'A1A 1A1',
    created_at timestamp default current_timestamp,
    updated_at timestamp on update current_timestamp
);
drop table if exists admins;
create table admins
(
    id int auto_increment primary key,
    email varchar(40) not null unique,
    pword varchar(255) not null
);
```

After that, our MySQL database backend should be ready.

## Accessing a MySQL database

We will try to separate the database access code from the business logic so we can roughly follow the model-view-controller paradigm. To use MySQL in a Vibe.d app, we need to add an external library to our project, in this case, mysql-native (there is DDBC but DDBC doesn't work on my Windows machine).

Create a new project named lorem.

```
C:\vibeprojects\empapp>cd ..
```

```
C:\vibeprojects>dub init lorem -t vibe.d
```

```
C:\vibeprojects>cd lorem
```

```
C:\vibeprojects\lorem>
```

The easy, convenient and proper way to add the mysql-native library is to use dub.

Here is the present state of dub.json:

```
{
  "authors": [
    "Owner"
  ],
  "copyright": "Copyright © 2023, Owner",
  "dependencies": {
    "vibe-d": "~>0.9"
  },
  "description": "A simple vibe.d server application.",
  "license": "proprietary",
  "name": "empapp"
}
```

To add an external library or dependency, we type dub add

```
c:\vibeprojects\empapp>dub add mysql-native  
Adding dependency mysql-native ~>3.2.0
```

And here is the dub.json file after we added mysql-native:

```
{  
  "authors": [  
    "Owner"  
,  
  "copyright": "Copyright © 2023, Owner",  
  "dependencies": {  
    "mysql-native": "~>3.2.0",  
    "vibe-d": "~>0.9"  
,  
    "description": "A simple vibe.d server application.",  
    "license": "proprietary",  
    "name": "empapp"  
}
```

Now we are ready to use MySQL in our project.

Let's review source\app.d to make sure it uses the web framework.

```
import vibe.vibe;  
import empcontrol;  
  
void main()  
{  
  auto settings = new HTTPServerSettings;
```

```
settings.port = 8080;
settings.bindAddresses = [":1", "127.0.0.1"];

auto router = new URLRouter;
router.get("*", serveStaticFiles("public/"));
router.registerWebInterface(new EmployeeController);

auto listener = listenHTTP(settings, router);
scope (exit) listener.stopListening();

runApplication();
}
```

Here we indicated that we are instantiating an EmployeeController class.

## The EmployeeController class

The EmployeeController class will be the embodiment of the web framework. It will also be the controller for linking our views with the data (the models).

Edit or create source\empcontrol.d:

```
module empcontrol;

import vibe.vibe;
import empmodel;

class EmployeeController
{
    private EmployeeModel empModel;
    private string realm = "The Lorem Ipsum Company";

    this()
    {
        empModel = new EmployeeModel();
    }

    void index()
    {
        render!"index.dt";
    }

    void getAddEmployee()
    {
        render!("empadd.dt", departments, provinces, paygrades);
    }
}
```

In the constructor, we instantiated an empModel variable so we can start using the EmployeeModel class.

The method

```
render! "index.dt";
```

is a **variadic function template**. It means render!() can have a varying number of parameters (arguments) of different data types. We have seen this construct before with staticTemplate!(). If there is only one parameter, the parentheses are optional.

In this case,

```
render! ("empadd.dt", departments, provinces, paygrades);
```

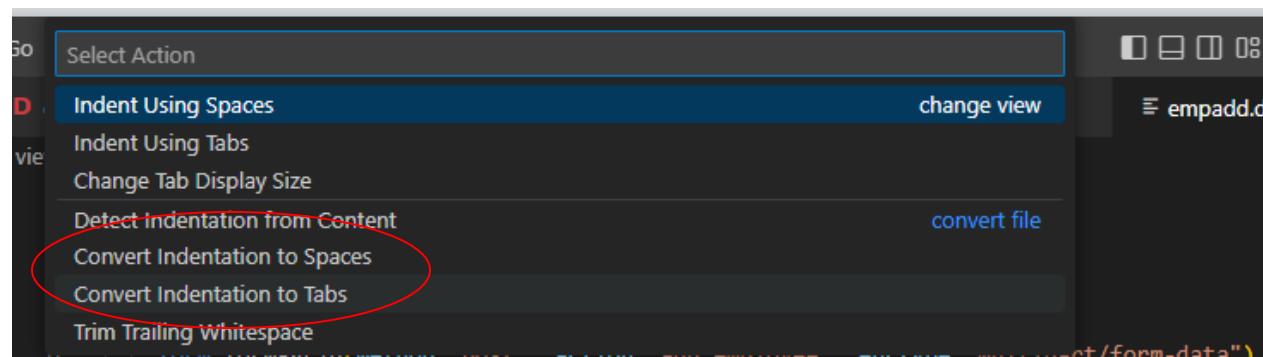
render!() has three arguments (or parameters), one string and three arrays, so we have to enclose them in parentheses.

Let us test by running the app.

Sometimes, instead of seeing a successful compilation, you might see several lines of errors instead. The usual culprit is the indentation. In that case, click on the ‘Tab Size.’ prompt at the bottom of VS Code for each file that cause an error:



And choose to either convert all tabs to spaces or all spaces to tabs from the choices at the top of VS Code to make indentations uniform and consistent and make sure all indentation rules are followed, then save the file.



The EmployeeController class is using an EmployeeModel class which we haven't created yet, so let's do that next.

## The EmployeeModel class

We are going to presume that this app will be deployed in an intranet setting and only a few people have access, and that there is a rare chance of multiple users using the app at the same time, so we won't have to talk about security, encryption, multiple users, etc.

A data-entry form in Vibe.d needs a struct data type from which it can extract the fields for saving into the database table. It makes things simpler if the database table, the struct and the form have a one-to-one correspondence in field names.

We will create a struct data structure that mimics an employee record. We will call that struct Employee.

Then we create a class that contains the methods to manipulate that struct, such as add, edit, get and delete, in short, all the CRUD operations (create, read, update, delete) on the MySQL table. It will be a model that represents the employees table on the MySQL server, so we will call that class EmployeeModel.

Create source\empmodel.d with this contents:

```
module empmodel;

import mysql;
import std.conv;
import std.array;

struct Employee
{
    int id; //row id or record id
    string empid; //employee number
    string dept; //department
    string paygd; //salary grade
    string email; //email address
    string pword; //password
    string fname; //first name
```

```
    string lname; //last name
    string phone; //phone number
    string photo; //ID photo
    string street; //street address
    string city; //city name
    string province; //province name
    string postcode; //postal code
}

struct Admin
{
    string email; //email address
    string pword; //password
}

string[] departments =
[
    "Management and Admin",
    "Accounting and Finance",
    "Production",
    "Maintenance",
    "Shipping and Receiving",
    "Purchasing and Supplies",
    "IT Services",
    "Human Resources",
    "Marketing"
];

string[][] provinces =
[
    ["AB", "Alberta"],
    ["BC", "British Columbia"],
```

```
[ "MB", "Manitoba"],
[ "NB", "New Brunswick"],
[ "NL", "Newfoundland and Labrador"],
[ "NS", "Nova Scotia"],
[ "NT", "Northwest Territories"],
[ "NU", "Nunavut"],
[ "ON", "Ontario"],
[ "PE", "Prince Edward Island"],
[ "QC", "Quebec"],
[ "SK", "Saskatchewan"],
[ "YT", "Yukon Territory"]
];

string[] paygrades =
[
    "A100", "A200", "A300", "A400",
    "B100", "B200", "B300", "B400",
    "C100", "C200", "C300", "C400",
    "D100", "D200", "D300", "D400",
    "E100", "E200", "E300", "E400",
    "F100", "F200", "F300", "F400"
];

class EmployeeModel
{
    Connection conn;

    this()
    {
        string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";
        conn = new Connection(url);
    }
}
```

```
    scope(exit) conn.close;
    insertIntoAdmins() // be sure to remove this after the first run!
}

void insertIntoAdmins()
{
    import vibe.http.auth.digest_auth;

    string email1 = "admin1@lorem.com";
    string email2 = "admin2@lorem.com";
    string email3 = "admin3@lorem.com";
    string realm = "The Lorem Ipsum Company";
    string pass1 = createDigestPassword(realm, email1, "secret");
    string pass2 = createDigestPassword(realm, email2, "secret");
    string pass3 = createDigestPassword(realm, email3, "secret");
    string sql = "insert into admins(email, pword)
        values ('" ~ email1 ~ "','" ~ pass1 ~ "')";
    conn.exec(sql);
    sql = "insert into admins(email, pword)
        values ('" ~ email2 ~ "','" ~ pass2 ~ "')";
    conn.exec(sql);
    sql = "insert into admins(email, pword)
        values ('" ~ email3 ~ "','" ~ pass3 ~ "')";
    conn.exec(sql);
}

ulong addEmployee(Employee e)
{
    string sql =
        "insert into employees
        (
            empid,
```

```

        dept, paygd, email, pword,
        fname, lname, phone, photo,
        street, city, province, postcode
    )
    values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
Prepared pstmt = conn.prepare(sql);
pstmt.setArgs
(
    to!string(e.empid),
    to!string(e.dept),
    to!string(e.paygd),
    to!string(e.email),
    to!string(e.pword),
    to!string(e.fname),
    to!string(e.lname),
    to!string(e.phone),
    to!string(e.photo),
    to!string(e.street),
    to!string(e.city),
    to!string(e.province),
    to!string(e.postcode)
);
    return conn.exec(pstmt);
}
}

```

We created the departments, the provinces and the paygrades array variables for forms that require drop-down options for departments, provinces and salary grades.

Also, there is an insertIntoAdmins() method that adds three records to the admins table as our test administrators. **Be sure to comment this out or erase these lines after the first compilation.**

In the `this()` constructor of the `EmployeeModel` class, we get a connection to the MySQL server running at `localhost (127.0.0.1)`, meaning, our own local machine where we installed MySQL.

The connection string we use for connecting to the MySQL server is

```
string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";
```

Of course, replace the user and the password fields with your own username and password for MySQL.

After that, we can connect to the `empdb` database through the `conn` variable.

As an aside, D constructors are written as `this()`, so we write what needs to be initialized when the class is instantiated in the constructor. Destructors are written as `~this()`.

The line

```
scope(exit) conn.close;
```

means that, when the `conn` variable gets out of scope, its `close()` method should be called before the garbage collector of the runtime erases it from memory (or the garbage collector recovers that part of memory occupied by the `conn` variable).

The `addEmployee()` method shows how to insert a record in mysql-native syntax.

We need to import `std.conv`

```
import std.conv;
```

so we can use the conversion functions of the D standard library, such as

```
to!string(e.deprt),
```

which converts the data in `e.deprt` (which we will get from the web form, which we haven't created yet) into a regular string before we save it to the database.

We need to add new records to the empty employees table, so we should create an employee data-entry form. Let's do that next.

## The data-entry form for adding a new employee record

Create views\empadd.dt for our data entry form.

```
extends layout
block maincontent
  include cssformgrid.dt
  div.form-grid-wrapper
    h2.center-align New employee details
    form.form-grid(method="post", action="add_employee", enctype="multipart/form-data")
      label.form-grid-label Employee number
      input.form-grid-input(type="empid", name="e.empid", placeholder="employee number", required)
      label.form-grid-label Department
      select#dept.form-grid-input(name="e.deprt")
        -foreach(dep; departments)
          option(value="#{dep}") #{dep}
      label.form-grid-label Salary grade
      select#dept.form-grid-input(name="e.paygd")
        -foreach(pay; paygrades)
          option(value="#{pay}") #{pay}
      label.form-grid-label Email address
      input.form-grid-input(type="email", name="e_email", placeholder="email@company.com", required)
      label.form-grid-label Password
      input.form-grid-input(type="password", name="e.pword", placeholder="password", required)
      label.form-grid-label First name
      input.form-grid-input(type="text", name="e.fname", placeholder="First name", required)
      label.form-grid-label Last name
      input.form-grid-input(type="text", name="e.lname", placeholder="Last name", required)
      label.form-grid-label Phone
      input.form-grid-input(type="text", name="e.phone", placeholder="Phone number")
      label.form-grid-label Street address (no city)
      input.form-grid-input(type="text", name="e.street", placeholder="Street address", required)
```

```

label.form-grid-label City
input.form-grid-input(type="text", name="e_city", placeholder="City", required)
label.form-grid-label Province
select#province.form-grid-input(name="e_province")
-foreach(prov; provinces)
    option(value="#{prov[0]}") #{prov[1]}
label.form-grid-label Postal code
input.form-grid-input(type="text", name="e_postcode", placeholder="A1A 1A1")
label.form-grid-label ID Picture
input.form-grid-input(type="file", name="picture")
input(type="hidden", name="e_photo")
input(type="hidden", name="e_id", value="1")
div
div
    input.form-grid-button(type="reset", value="Clear the form")
    input.form-grid-button(type="submit", value="Submit")

```

The line

```
form.form-grid(method="post",action="add_employee",enctype="multipart/form-data")
```

will call the postAddEmployee() method of the EmployeeController class, which we haven't created yet.

The part of the form declaration

```
form.form-grid(method="post",action="add_employee",enctype="multipart/form-data")
```

means the form will upload a file, which in this case is a photo.

We can mix D code inside a Diet template file with a – (hyphen). Thus the line

```
-foreach(dep; departments)
```

and

```
-foreach(prov; provinces)
```

and

```
-foreach(pay; paygrades)
```

mean iterate (or loop) through the departments or provinces or paygrades array to access each item.

The form includes views\cssformgrid.dt for its CSS formatting.

Here is the views\cssformgrid.dt.

```
:css
.form-grid-wrapper
{
    width: 700px;
    height: auto;
    margin: 20px auto;
    padding: 1px 20px 20px 0;
    border-radius: 20px;
    background-color: #eef;
}
.form-grid
{
    display: grid;
    grid-template-columns: 1fr 3fr;
    gap: 5px;
}
.center-align
{
```

```
    text-align: center;
}
.form-grid-label
{
    width: 100%;
    font-size: 16px;
    text-align: right;
    padding: 5px 0;
}
.form-grid-field
{
    width: 100%;
    font-size: 16px;
    border-bottom: 1px solid black;
    padding: 5px 0;
}
.form-grid-button
{
    width: 100%;
    font-size: 16px;
    height: 40px;
    margin-top: 10px;
}
```

Now let's try to run the app.

```
c:\vibeprojects\lorem>dub
Starting Performing "debug" build using C:\D\dmd2\windows\bin\dmd.exe for x86_64.
...
Finished To force a rebuild of up-to-date targets, run again with --force
Copying files for vibe-d:tls...
Running lorem.exe
```

```
[main(---) INF] Listening for requests on http://[::1]:8080/  
[main(---) INF] Listening for requests on http://127.0.0.1:8080/
```

Click on the ‘Add employee’ link and you should see this:

[Home](#)   [All employees](#)   [Find employee](#)   [Add employee](#)   [Login](#)

### New employee details

Department ▼

Salary grade ▼

Email address

Password

First name

Last name

Phone

Street address (no city)

City

Province ▼

Postal code

ID Picture  No file chosen

But after you fill out the form and click Submit, you get this error:

```
localhost:8080/add_employee
404 - Not Found
Not Found
Internal error information:
No routes match path '/add_employee'
```

That's because we haven't written the postAddEmployee() method yet. Take note that we wrote this in the empadd.dt form:

```
form.form-grid(method="post",action="add_employee",enctype="multipart/form-data")
```

which means the server will call the postAddEmployee() method, which we haven't written yet. Let's rectify that.

## Saving data from the form into the database

Append this code to source\empcontrol.d

```
void postAddEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string photopath = "none yet";
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
            try moveFile(pic.tempPath, NativePath(fullpath));
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
        }
        e.photo = photopath;
    }
    if(e.phone.length == 0) e.phone = "(123) 456 7890";
    if(e.paygd.length == 0) e.paygd = "none yet";
    if(e.postcode.length == 0) e.postcode = "A1A 1A1";
    e.pword = createDigestPassword(realm, e.email, e.pword);
```

```
    empModel.addEmployee(e);
    redirect("all_employees");
}
```

The photo being uploaded is in `request.files`. Remember that the `HTTPServerRequest` and the `HTTPServerResponse` are automatically available to us as the `request` and `response` variables.

So the line

```
auto pic = "picture" in request.files;
```

means we are extracting the uploaded file from `request.files` into the `pic` variable.

The word “picture” in that line corresponds to the name we gave to the button field in the `views\empadd.dt` form, which becomes the `picture` variable.

```
input.form-grid-input(type="file", name="picture")
```

The data-entry form requires a photo image file to be uploaded. Uploaded image files are usually not saved into the database as that would easily make the database bloated, so we need to save them into a folder which, if not already existing, we should create. Hence, we need to import some library modules related to file operations

```
import std.file;
import std.path;
import std.algorithm;
```

so we can call these file operation functions:

```
string ext = extension(pic.filename.name);
string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
if(canFind(exts, ext))
{
    photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
```

```
    string dir = "./public/uploads/photos/";
    mkdirRecurse(dir);
    string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
    try moveFile(pic.tempPath, NativePath(fullpath));
    catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
}
```

The uploaded photos will be saved in the \public\uploads\photos\ folder and the filename will be changed to the employee's name (firstname\_lastname.ext).

The passwords need to be encrypted before they are saved so even if bad guys gained access to the data, the passwords are still safely unreadable. The employees may need to have access to their own data in the future, such as access to time worked and salary for the period, so we need to keep their passwords safe.

So we need to import the relevant library for encryption:

```
import vibe.http.auth.digest_auth;
```

so we can call this function:

```
e.pword = createDigestPassword(realm, e.email, e.pword);
```

We created the realm variable near the top of the class:

```
private string realm = "The Lorem Ipsum Company";
```

as it is needed by the createDigestPassword() function.

Vibe.d sometimes generates errors when importing blank data from the database. If some of the non-required fields are blank, we need to fill them with dummy data:

```
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
```

```
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
```

before saving the record to the database.

The code

```
empModel.addEmployee(e);
```

does the actual saving into the database.

Now we can test the whole thing.

## Testing the whole thing

Let's look at source\app.d first

```
import vibe.vibe;
import empcontrol;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];

    auto router = new URLRouter;
    router.get("*", serveStaticFiles("public/"));
    router.registerWebInterface(new EmployeeController);

    auto listener = listenHTTP(settings, router);
    scope(exit) listener.stopListening();

    runApplication();
}
```

Then try to compile. If you get errors (the views\empadd.dt is a big file), they may be just simple typographical errors, like this:

```
500 - Internal Server Error

Internal Server Error

Internal error information:
object.Exception@C:\Users\owner\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-
d\data\vibe\data\serialization.d(918): Missing non-optional field 'id' of type 'Employee'
(DefaultPolicy(T)) .
-----
```

```
0x00007FF611495B23 in std.exception.bailOut! (object.Exception).bailOut at
C:\D\dmdd2\windows\bin\..\..\src\phobos\std\exception.d(516)
0x00007FF611495A29 in std.exception.enforce!().enforce!bool.enforce at
C:\D\dmdd2\windows\bin\..\..\src\phobos\std\exception.d(437)
0x00007FF6114FF20A in vibe.data.serialization.deserializeValueImpl! (vibe.data bson.BsonSerializer,
DefaultPolicy).deserializeValueDeduced! (empmongo.Employee).deserializeValueDeduced at
C:\D\dmdd2\windows\bin\..\..\src\phobos\std\exception.d(434)
0x00007FF6114FF0FD in vibe.data.serialization.deserializeValueImpl! (vibe.data bson.BsonSerializer,
DefaultPolicy).deserializeValue! (empmongo.Employee).deserializeValue at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\data\vibe\data\serialization.d(691)
0x00007FF6114FF0B5 in vibe.data.serialization.deserializeWithPolicy! (vibe.data bson.BsonSerializer,
DefaultPolicy, empmongo.Employee, vibe.data bson.Bson).deserializeWithPolicy at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\data\vibe\data\serialization.d(304)
0x00007FF6114FF031 in vibe.data.serialization.deserialize! (vibe.data bson.BsonSerializer,
empmongo.Employee, vibe.data bson.Bson).deserialize at C:\Users\rey\AppData\Local\dub\packages\vibe-
d-0.9.3\vibe-d\data\vibe\data\serialization.d(270)
0x00007FF6114FEFE1 in vibe.data bson.deserializeBson! (empmongo.Employee).deserializeBson at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\data\vibe\data\bson.d(1217)
0x00007FF61150ADB2 in empmongo.EmployeeModel.getEmployees at
C:\vibeprojects\lorem\source\empmongo.d(80)
0x00007FF611507728 in loremService.LoremInterface.getListEmployees at
C:\vibeprojects\lorem\source\loremservice.d(166)
0x00007FF6114D7FFF in vibe.web.web.handleRequest! ("getListEmployees", getListEmployees,
loremService.LoremInterface).handleRequest at C:\Users\rey\AppData\Local\dub\packages\vibe-d-
0.9.3\vibe-d\web\vibe\web.d(1043)
0x00007FF61149FFDD in vibe.web.web.registerWebInterface! (loremService.LoremInterface,
MethodStyle.lowerUnderScored).registerWebInterface._lambda20 at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\web\vibe\web.d(214)
0x00007FF6115CB2F4 in vibe.http.router.URLRouter.handleRequest._lambda4 at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\http\vibe\http\router.d(218)
0x00007FF6115E294B in vibe.http.router.MatchTree! (vibe.http.router.Route).MatchTree.doMatch at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\http\vibe\http\router.d(674)
0x00007FF6115E1C46 in vibe.http.router.MatchTree! (vibe.http.router.Route).MatchTree.match at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\http\vibe\http\router.d(607)
0x00007FF6115CAC5D in vibe.http.router.URLRouter.handleRequest at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\http\vibe\http\router.d(211)
0x00007FF61164B50A in vibe.http.server.handleRequest at C:\Users\rey\AppData\Local\dub\packages\vibe-
d-0.9.3\vibe-d\http\vibe\http\server.d(2292)
0x00007FF61160CD25 in vibe.http.server.handleHTTPConnection._lambda4 at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\http\vibe\http\server.d(253)
```

```
0x00007FF61160C587 in vibe.http.server.handleHTTPConnection at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\http\vibe\http\server.d(254)
0x00007FF6115E0279 in vibe.http.server.listenHTTPPlain.doListen._lambda6 at
C:\Users\rey\AppData\Local\dub\packages\vibe-d-0.9.3\vibe-d\http\vibe\http\server.d(2048)
0x00007FF61188D255 in void vibe.core.task.TaskFuncInfo.set!(void
delegate(vibe.core.net.TCPConnection) @safe, vibe.core.net.TCPConnection).set(ref void
delegate(vibe.core.net.TCPConnection) @safe, ref vibe.core.net.TCPConnection).callDelegate(ref
vibe.core.task.TaskFuncInfo) at C:\Users\rey\AppData\Local\dub\packages\vibe-core-1.13.0\vibe-
core\source\vibe\core\task.d-mixin-712(712)
0x00007FF611856596 in vibe.core.task.TaskFuncInfo.call at
C:\Users\rey\AppData\Local\dub\packages\vibe-core-1.13.0\vibe-core\source\vibe\core\task.d(730)
0x00007FF611836AE6 in vibe.core.task.TaskFiber.run at C:\Users\rey\AppData\Local\dub\packages\vibe-
core-1.13.0\vibe-core\source\vibe\core\task.d(439)
0x00007FF6119B64CF in void core.thread.context.Callable.opCall()
0x00007FF6119B8BA7 in fiber_entryPoint
0x00007FF61198C989 in pure nothrow @nogc void core.thread.fiber.Fiber.initStack().trampoline()
```

It is a very long error message but the culprit is just a typo in views\empadd.dt, which is e\_\_id (double underscore) instead of e\_id.

Once you resolved the errors, compile, run and refresh your browser.

Then click on Add employee and enter sample data to test the new data-entry system.

Employee Timekeeping System

localhost:8080/add\_employee

Home All employees Find employee Add employee (circled)

Login

### New employee details

Department	Accounting and Finance
Salary grade	B300
Email address	you@mail.com
Password	.....
First name	You
Last name	MySurname
Phone	0987654321
Street address (no city)	101 First St.
City	Toronto
Province	Ontario
Postal code	A1A 1A1
ID Picture	<input type="button" value="Choose File"/> GettyImages-1173175729-1000x667.jpg

Copyright © The Lorem Ipsum Company 2023

And click Submit, which will return this error:

404 - Not Found

Not Found

```
Internal error information:  
No routes match path '/all_employees'
```

In the class EmployeeController, we have this line at the end of postAddEmployee() method:

```
redirect("all_employees");
```

which will become the URL [http://localhost:8080/all\\_employees](http://localhost:8080/all_employees) which corresponds to the method getAllEmployees(), which we haven't written yet, so let's write it.

## Listing all the employees

Edit source\empcontrol.d and add this code:

```
void getAllEmployees()
{
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps);
}
```

Now, that's a very simple method, just two lines. But it is calling the empModel.getEmployees() method, which we haven't written yet, and also needs the emplistall.dt file, which we haven't created yet, so let's resolve that.

First, let's define the getEmployees() method at the end of source\empmodel.d:

```
Employee[] getEmployees()
{
    Employee[] emps;
    string sql = "select * from employees";
    Row[] rows = conn.query(sql).array;
    if(rows.length == 0) return emps;
    return prepareEmployees(rows);
}
```

This method returns all the records in the table. But this method calls the prepareEmployees() method, so let's define it.

```
Employee[] prepareEmployees(Row[] rows)
{
    Employee[] emps;
    foreach(row; rows)
    {
        Employee e = prepareEmployee(row);
```

```
        emps ~= e;
    }
    return emps;
}
```

This method in turn calls the `prepareEmployee()` (singular, no ‘s’) method, so let’s define that too.

```
Employee prepareEmployee(List<String> row)
{
    Employee e;
    e.id = to!int(to!string(row[0]));
    e.empid = to!string(row[1]);
    e.deprt = to!string(row[2]);
    e.paygd = to!string(row[3]);
    e.email = to!string(row[4]);
    e.pword = to!string(row[5]);
    e.fname = to!string(row[6]);
    e.lname = to!string(row[7]);
    e.phone = to!string(row[8]);
    e.photo = to!string(row[9]);
    e.street = to!string(row[10]);
    e.city = to!string(row[11]);
    e.province = to!string(row[12]);
    e.postcode = to!string(row[13]);
    return e;
}
```

You should make sure the order of fields in `prepareEmployee()` follows the order of fields in the MySQL employees table, for sometimes MySQL does not save the fields in the same order as in your SQL CREATE TABLE statement. Check the Schemas view of MySQL Browser.

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the database schema. Under the 'SCHEMAS' section, 'empdb' is selected, revealing its tables: 'admins' and 'employees'. The 'employees' table is expanded to show its columns: id, empid, dept, paygd, email, pword, fname, lname, phone, photo, street, city, province, postcode, created\_at, and updated\_at. On the right, the 'Query 1' tab is active, showing an SQL script for creating the 'employees' table:

```

1 • use empdb;
2 • drop table if exists employees;
3 • create table employees
4 ( (
5     id int auto_increment primary key,
6     empid char(4) not null unique,
7     dept varchar(30) not null,
8     paygd varchar(10) default 'none ye'
9     email varchar(40) not null unique,
10    pword varchar(255) not null,
11    fname varchar(30) not null,
12    lname varchar(30) not null,
13    phone varchar(20) default '(123) 4'
14    photo varchar(50) default 'No photo'
15    street varchar(50) default '123 Fin'
16    city varchar(30) default 'Toronto',
17    province char(2) default 'ON',
18    postcode varchar(10) default 'A1A '

```

Now let's create views\emplistall.dt.

```

extends layout
block maincontent
  include csstable.dt
  div.table-wrapper
    table
      tr
        th Employee Id

```

```
th First name
th Last name
th Department
th Phone number
th Email address
th Street address
th City
th Province
th PostCode
th Action
-foreach(e; emps)
tr
td #{e.empid}
td #{e.fname}
td #{e.lname}
td #{e.deprt}
td #{e.phone}
td #{e.email}
td #{e.street}
td #{e.city}
td #{e.province}
td #{e.postcode}
td &nbsp;
form.form-hidden(method="get", action="edit_employee")
input(type="hidden", name="id", value="#{e.id}")
input(type="image", src="images/pencil.ico", height="15px")
| &nbsp;
form.form-hidden(method="get", action="delete_employee")
input(type="hidden", name="id", value="#{e.id}")
input(type="image", src="images/trash.ico", height="15px")
| &nbsp;
```

As mentioned before, we can mix D code inside a Diet template file with – (hyphen)

Remember that we passed an Employee array emps to the template views\emplistall.dt:

```
void getListEmployees()
{
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps);
}
```

Hence, we can inject this code:

```
-foreach(e; emps)
```

which will iterate through all of emps, which is an array of Employees, with e representing the current Employee record being processed inside the loop.

To display the value of a variable passed to the template, we use the construct #{variable}. This is equivalent to JSP's <%= variable %> construct.

Hence the line

```
td #{e.deprt}
```

means display the value of e.deprt inside that table cell.

And here is views\csstable.dt which is needed by emplistall.dt.

```
:css
.table-wrapper
{
    margin: 20px auto;
    border-radius: 20px;
```

```
}

table
{
    margin: 0 auto;
    padding: 20px 0;
    border-collapse: collapse;
    background-color: #eff;
}

table td, table th
{
    border: 1px solid black;
    margin: 0;
    padding: 0 5px;
}

.no-border
{
    border: 0;
}
```

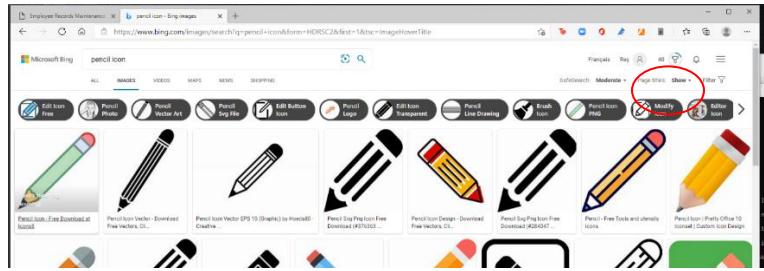
Compile and run the app and refresh the browser. Click on New employee again and add another employee, then click the Submit button.

And you should see this:

First name	Last name	Department	Salary grade	Phone number	Email address	Street address	City	Province	PostCode	Action
The	Boss	Management and Admin	A100	12345678	theboss@mail.com	101 First St.	Toronto	ON	A1A 1A1	
Gal	Gadot	Accounting and Finance	A300	0987654321	gal@gadot.com	102 First St.	Toronto	ON	A1A 1A2	
Eugene	Bou	Management and Admin	A200	31247425645	ebouchard@mail.com	103 First St.	Toronto	ON	A1A 1A3	

Copyright © The Lorem Ipsum Company 2023

I got the icons for the pencil and the trash bin by simply googling for ‘pencil icon’ and ‘trash bin icon’ and saving the icons into the \public\images\ folder.



In the emplistall.dt file, we have these lines

```
input(type="image", src="images/pencil.ico", height="15px")
```

```
input(type="image", src="images/trash.ico", height="15px")
```

so save and name the icon files in the \public\images\ folder with the same names.

Add more sample records by clicking on the Add employee link after each submission.

Here is the complete source\emppmodel.d so far:

```
module emppmodel;

import mysql;
import std.conv;
import std.array;

struct Employee
{
    int id; //row id or record id
    string empid; //employee number
```

```
    string dept; //department
    string paygd; //salary grade
    string email; //email address
    string pword; //password
    string fname; //first name
    string lname; //last name
    string phone; //phone number
    string photo; //ID photo
    string street; //street address
    string city; //city name
    string province; //province name
    string postcode; //postal code
}

struct Admin
{
    string email; //email address
    string pword; //password
}

string[] departments =
[
    "Management and Admin",
    "Accounting and Finance",
    "Production",
    "Maintenance",
    "Shipping and Receiving",
    "Purchasing and Supplies",
    "IT Services",
    "Human Resources",
    "Marketing"
];
```

```
string[][] provinces =
[
    ["AB", "Alberta"],
    ["BC", "British Columbia"],
    ["MB", "Manitoba"],
    ["NB", "New Brunswick"],
    ["NL", "Newfoundland and Labrador"],
    ["NS", "Nova Scotia"],
    ["NT", "Northwest Territories"],
    ["NU", "Nunavut"],
    ["ON", "Ontario"],
    ["PE", "Prince Edward Island"],
    ["QC", "Quebec"],
    ["SK", "Saskatchewan"],
    ["YT", "Yukon Territory"]
];

string[] paygrades =
[
    "A100", "A200", "A300", "A400",
    "B100", "B200", "B300", "B400",
    "C100", "C200", "C300", "C400",
    "D100", "D200", "D300", "D400",
    "E100", "E200", "E300", "E400",
    "F100", "F200", "F300", "F400"
];

class EmployeeModel
{
    Connection conn;
```

```
this()
{
    string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";
    conn = new Connection(url);
    scope(exit) conn.close;
}

ulong addEmployee(Employee e)
{
    string sql =
        "insert into employees
        (
            empid,
            dept, paygd, email, pword,
            fname, lname, phone, photo,
            street, city, province, postcode
        )
        values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs
    (
        to!string(e.empid),
        to!string(e.deprt),
        to!string(e.paygd),
        to!string(e.email),
        to!string(e.pword),
        to!string(e.fname),
        to!string(e.lname),
        to!string(e.phone),
        to!string(e.photo),
        to!string(e.street),
        to!string(e.province),
        to!string(e.postcode)
    );
}
```

```
        to!string(e.city),
        to!string(e.province),
        to!string(e.postcode)
    );
    return conn.exec(pstmt);
}

Employee[] getEmployees()
{
    Employee[] emps;
    string sql = "select * from employees";
    Row[] rows = conn.query(sql).array;
    if(rows.length == 0) return emps;
    return prepareEmployees(rows);
}

Employee[] prepareEmployees(Row[] rows)
{
    Employee[] emps;
    foreach(row; rows)
    {
        Employee e = prepareEmployee(row);
        emps ~= e;
    }
    return emps;
}

Employee prepareEmployee(Row row)
{
    Employee e;
    e.id = to!int(to!string(row[0]));
    e.empid = to!string(row[1]);
}
```

```
        e.deprt = to!string(row[2]);
        e.paygd = to!string(row[3]);
        e.email = to!string(row[4]);
        e.pword = to!string(row[5]);
        e.fname = to!string(row[6]);
        e.lname = to!string(row[7]);
        e.phone = to!string(row[8]);
        e.photo = to!string(row[9]);
        e.street = to!string(row[10]);
        e.city = to!string(row[11]);
        e.province = to!string(row[12]);
        e.postcode = to!string(row[13]);
        return e;
    }
}
```

And here is the full source\empcontrol.d so far:

```
module empcontrol;

import vibe.vibe;
import empmodel;

class EmployeeController
{
    private EmployeeModel empModel;
    private string realm = "The Lorem Ipsum Company";

    this()
    {
        empModel = new EmployeeModel;
    }
```

```
void index()
{
    render!("index.dt");
}

void getAddEmployee()
{
    render!("empadd.dt", departments, paygrades, provinces);
}

void postAddEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string photopath = "none yet";
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
            try moveFile(pic.tempPath, NativePath(fullpath));
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
        }
    }
}
```

```
    }
    e.photo = photopath;
}
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
e.pword = createDigestPassword(realm, e.email, e.pword);
empModel.addEmployee(e);
redirect("all_employees");
}

void getAllEmployees()
{
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps);
}
}
```

Now let's implement the editing of records to make the pencil icon functional.

## Retrieving and displaying an employee record for editing

First, we need to retrieve the record to be edited. We defined an id field in the employees table that has a unique number for each row, and this field is used to identify the row (being the primary key).

We included a hidden field in the emplistall.dt file: the id field, which has the value of the id field in the employees table, and which we can use as a key to retrieve the record so we can populate the form with the employee data.

```
input(type="hidden", name="id", value="#{e.id}")
```

Edit source\empcontrol.d and add this method at the end:

```
void getEditEmployee(int id)
{
    Employee e = empModel.getEmployee(id);
    render!("empedit.dt", e, departments, paygrades, provinces);
}
```

This method calls the empModel.getEmployee() method with the id as the argument, then receives the returned Employee with the needed data, which is used to populate the form empedit.dt, which we should create afterwards.

Edit source\empmodel.d and add this method at the end:

```
Employee getEmployee(int id)
{
    string sql = "select * from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Employee e;
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return e;
    return prepareEmployee(rows[0]);
```

```
}
```

Create views\empedit.dt.

```
extends layout
block maincontent
  include cssformgrid.dt
  div.form-grid-wrapper
    h2.center-align Edit employee details
    form.form-grid(method="post",action="edit_employee",enctype="multipart/form-data")
      label.form-grid-label Department
      select#dept.form-grid-input(name="e_dept", value="#{e.deprt}")
        -foreach(dep; departments)
          -if(dep == e.deprt)
            option(value="#{dep}", selected) #{dep}
          -else
            option(value="#{dep}") #{dep}
      label.form-grid-label Salary grade
      select#paygd.form-grid-input(name="e_paygd", value="#{e.paygd}")
        -foreach(pay; paygrades)
          -if(pay == e.paygd)
            option(value="#{pay}", selected) #{pay}
          -else
            option(value="#{pay}") #{pay}
      label.form-grid-label Email address
      input.form-grid-input(type="email", name="e_email", value="#{e.email}")
      label.form-grid-label Password
      input.form-grid-input(type="password", name="e_pword", value="#{e.pword}")
      label.form-grid-label First name
      input.form-grid-input(type="text", name="e_fname", value="#{e.fname}")
      label.form-grid-label Last name
      input.form-grid-input(type="text", name="e_lname", value="#{e.lname}")
```

```

label.form-grid-label Phone
input.form-grid-input(type="text", name="e_phone", value="#{e.phone}")
label.form-grid-label Street address (no city)
input.form-grid-input(type="text", name="e_street", value="#{e.street}")
label.form-grid-label City
input.form-grid-input(type="text", name="e_city", value="#{e.city}")
label.form-grid-label Province
select#province.form-grid-input(name="e_province", value="#{e.province}")
-foreach(prov; provinces)
-if(prov[0] == e.province)
    option(value="#{prov[0]}", selected) #{prov[1]}
-else
    option(value="#{prov[0]}") #{prov[1]}
label.form-grid-label Postal code
input.form-grid-input(type="text", name="e_postcode", value="#{e.postcode}")
label.form-grid-label ID Picture
input.form-grid-input(type="file", name="picture")
input(type="hidden", name="e_photo", value="#{e.photo}")
input(type="hidden", name="e_id", value="#{e.id}")
div
div
a(href="all_employees")
    button.form-grid-button(type="button") Cancel
    input.form-grid-button(type="submit", value="Submit")

```

The form is displayed using the Employee data that was passed to it.

Compile, run and refresh the browser and click on a pencil icon to edit a record.

Timekeeping System

localhost:8080/edit\_employee?id=1&x=108&y=3

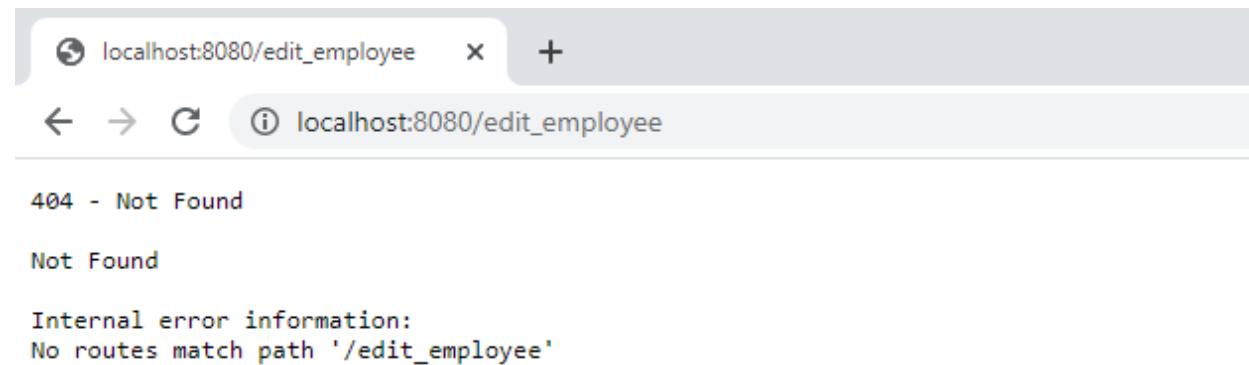
All employees Find employee Add employee

### Edit employee details

Department	Shipping and Receiving
Salary grade	A100
Email address	me@mail.com
Password	.....
First name	Me
Last name	Myfamily
Phone	12345678
Street address (no city)	101 First St.
City	Toronto
Province	Ontario
Postal code	A1A 1A1
ID Picture	<input type="button" value="Choose File"/> No file chosen

Great, a record was opened for editing.

But after clicking Submit, we get this error:



It means we haven't defined the methods to save the changes to the database. Let's do that.

## Saving the form changes to the database

Append this code to source\empcontrol.d.

```
void postEditEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    string photopath = e.photo;
    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
            try moveFile(pic.tempPath, NativePath(fullpath));
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
        }
    }
    e.photo = photopath;
    if(e.phone.length == 0) e.phone = "(123) 456 7890";
    if(e.paygd.length == 0) e.paygd = "none yet";
    if(e.postcode.length == 0) e.postcode = "A1A 1A1";
    e.pword = createDigestPassword(realm, e.email, e.pword);
```

```
    empModel.editEmployee(e);
    redirect("all_employees");
}
```

The only differences between this method and the postAddEmployee() method are the fifth line and the second to the last line, which is the call to empModel.editEmployee() method, which we haven't defined yet.

Let's add the method at the end of source\empmodel.d.

```
ulong editEmployee(Employee e)
{
    string sql = "update employees set empid=?,
        dept=? , paygd=? , email=? , pword=? ,
        fname=? , lname=? , phone=? , photo=? ,
        street=? , city=? , province=? , postcode=? ,
        where id=?";
    Prepared pstmt = conn.prepareStatement(sql);
    pstmt.setArgs
    (
        to!string(e.empid),
        to!string(e.deprt),
        to!string(e.paygd),
        to!string(e.email),
        to!string(e.pword),
        to!string(e.fname),
        to!string(e.lname),
        to!string(e.phone),
        to!string(e.photo),
        to!string(e.street),
        to!string(e.city),
        to!string(e.province),
        to!string(e.postcode),
```

```
    to!int(to!string(e.id))
);
return conn.exec(stmt);
}
```

The SQL statement basically says ‘get that record with this id then update that record with the following data’.

Compile, run and refresh the browser, then click on a pencil icon to display that record for editing.

[Employees](#)[Find employee](#)[Add employee](#)

## Edit employee details

Employee number Department Salary grade Email address Password First name Last name Phone Street address (no city) City Province Postal code 

ID Picture

 No file chosen

Make some changes to the record and press Submit. You should be able to see your changes saved.

So here is the full source\empcontrol.d so far.

```
module empcontrol;

import vibe.vibe;
import empmodel;

class EmployeeController
{
    EmployeeModel empModel;
    string realm = "The Lorem Ipsum Company";

    this()
    {
        empModel = new EmployeeModel();
    }

    void index()
    {
        render!"index.dt";
    }

    void getAddEmployee()
    {
        render!("empadd.dt", departments, paygrades, provinces);
    }

    void postAddEmployee(Employee e)
    {
        import std.file;
        import std.path;
        import std.algorithm;
        import vibe.http.auth.digest_auth;
```

```
auto pic = "picture" in request.files;
if(pic !is null)
{
    string photopath = "none yet";
    string ext = extension(pic.filename.name);
    string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
    if(canFind(exts, ext))
    {
        photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
        string dir = "./public/uploads/photos/";
        mkdirRecurse(dir);
        string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
        try moveFile(pic.tempPath, NativePath(fullpath));
        catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
    }
    e.photo = photopath;
}
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
e.pword = createDigestPassword(realm, e.email, e.pword);
empModel.addEmployee(e);
redirect("all_employees");
}

void getAllEmployees()
{
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps);
}
```

```
void getEditEmployee(int id)
{
    Employee e = empModel.getEmployee(id);
    render!("empedit.dt", e, departments, paygrades, provinces);
}

void postEditEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    string photopath = e.photo;
    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
            try moveFile(pic.tempPath, NativePath(fullpath));
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
        }
    }
    e.photo = photopath;
    if(e.phone.length == 0) e.phone = "(123) 456 7890";
    if(e.paygd.length == 0) e.paygd = "none yet";
```

```
    if(e.postcode.length == 0) e.postcode = "A1A 1A1";
    e.pword = createDigestPassword(realm, e.email, e.pword);
    empModel.editEmployee(e);
    redirect("all_employees");
}
}
```

And here is the full source\empmodel.d so far.

```
module empmodel;

import mysql;
import std.conv;
import std.array;

struct Employee
{
    int id; //row id or record id
    string empid; //employee number
    string dept; //department
    string paygd; //salary grade
    string email; //email address
    string pword; //password
    string fname; //first name
    string lname; //last name
    string phone; //phone number
    string photo; //ID photo
    string street; //street address
    string city; //city name
    string province; //province name
    string postcode; //postal code
}
```

```
struct Admin
{
    string email; //email address
    string pword; //password
}

string[] departments =
[
    "Management and Admin",
    "Accounting and Finance",
    "Production",
    "Maintenance",
    "Shipping and Receiving",
    "Purchasing and Supplies",
    "IT Services",
    "Human Resources",
    "Marketing"
];

string[][] provinces =
[
    ["AB", "Alberta"],
    ["BC", "British Columbia"],
    ["MB", "Manitoba"],
    ["NB", "New Brunswick"],
    ["NL", "Newfoundland and Labrador"],
    ["NS", "Nova Scotia"],
    ["NT", "Northwest Territories"],
    ["NU", "Nunavut"],
    ["ON", "Ontario"],
    ["PE", "Prince Edward Island"],
];
```

```
    ["QC", "Quebec"],
    ["SK", "Saskatchewan"],
    ["YT", "Yukon Territory"]
];

string[] paygrades =
[
    "A100", "A200", "A300", "A400",
    "B100", "B200", "B300", "B400",
    "C100", "C200", "C300", "C400",
    "D100", "D200", "D300", "D400",
    "E100", "E200", "E300", "E400",
    "F100", "F200", "F300", "F400"
];

class EmployeeModel
{
    Connection conn;

    this()
    {
        string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";
        conn = new Connection(url);
        scope(exit) conn.close;
    }

    ulong addEmployee(Employee e)
    {
        string sql =
            "insert into employees
            (

```

```
        empid,
        dept, paygd, email, pword,
        fname, lname, phone, photo,
        street, city, province, postcode
    )
    values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
Prepared pstmt = conn.prepare(sql);
pstmt.setArgs
(
    to!string(e.empid),
    to!string(e.deprt),
    to!string(e.paygd),
    to!string(e.email),
    to!string(e.pword),
    to!string(e.fname),
    to!string(e.lname),
    to!string(e.phone),
    to!string(e.photo),
    to!string(e.street),
    to!string(e.city),
    to!string(e.province),
    to!string(e.postcode)
);
return conn.exec(pstmt);
}

Employee[] getEmployees()
{
    Employee[] emps;
    string sql = "select * from employees";
    Row[] rows = conn.query(sql).array;
    if(rows.length == 0) return emps;
```

```
    return prepareEmployees(rows);
}

Employee[ ] prepareEmployees(Row[ ] rows)
{
    Employee[ ] emps;
    foreach(row; rows)
    {
        Employee e = prepareEmployee(row);
        emps ~= e;
    }
    return emps;
}

Employee prepareEmployee(Row row)
{
    Employee e;
    e.id = to!int(to!string(row[0]));
    e.empid = to!string(row[1]);
    e.deprt = to!string(row[2]);
    e.paygd = to!string(row[3]);
    e.email = to!string(row[4]);
    e.pword = to!string(row[5]);
    e.fname = to!string(row[6]);
    e.lname = to!string(row[7]);
    e.phone = to!string(row[8]);
    e.photo = to!string(row[9]);
    e.street = to!string(row[10]);
    e.city = to!string(row[11]);
    e.province = to!string(row[12]);
    e.postcode = to!string(row[13]);
    return e;
}
```

```
}

Employee getEmployee(int id)
{
    string sql = "select * from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Employee e;
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return e;
    return prepareEmployee(rows[0]);
}

ulong editEmployee(Employee e)
{
    string sql = "update employees set empid=?,
        dept=? , paygd=? , email=? , pword=? ,
        fname=? , lname=? , phone=? , photo=? ,
        street=? , city=? , province=? , postcode=?
        where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs
    (
        to!string(e.empid),
        to!string(e.dept),
        to!string(e.paygd),
        to!string(e.email),
        to!string(e.pword),
        to!string(e.fname),
        to!string(e.lname),
        to!string(e.phone),
        to!string(e.photo),
```

```
    to!string(e.street),
    to!string(e.city),
    to!string(e.province),
    to!string(e.postcode),
    to!int(to!string(e.id))
);
return conn.exec(pstmt);
}
}
```

Time to make the trash bin icon functional.

## Deleting records in the database

When a user clicks on the trash bin icon, it should not immediately delete the corresponding record. The user should be given the chance to recover if the user made a mistake. We can simply display a page showing the employee details first, then ask the user for confirmation.

Edit source\empcontrol.d and add this code.

```
void getDeleteEmployee(int id)
{
    Employee e = empModel.getEmployee(id);
    render!("empdelete.dt", e);
}
```

We retrieve the employee record and display that record in empdelete.dt.

So create views\empdelete.dt.

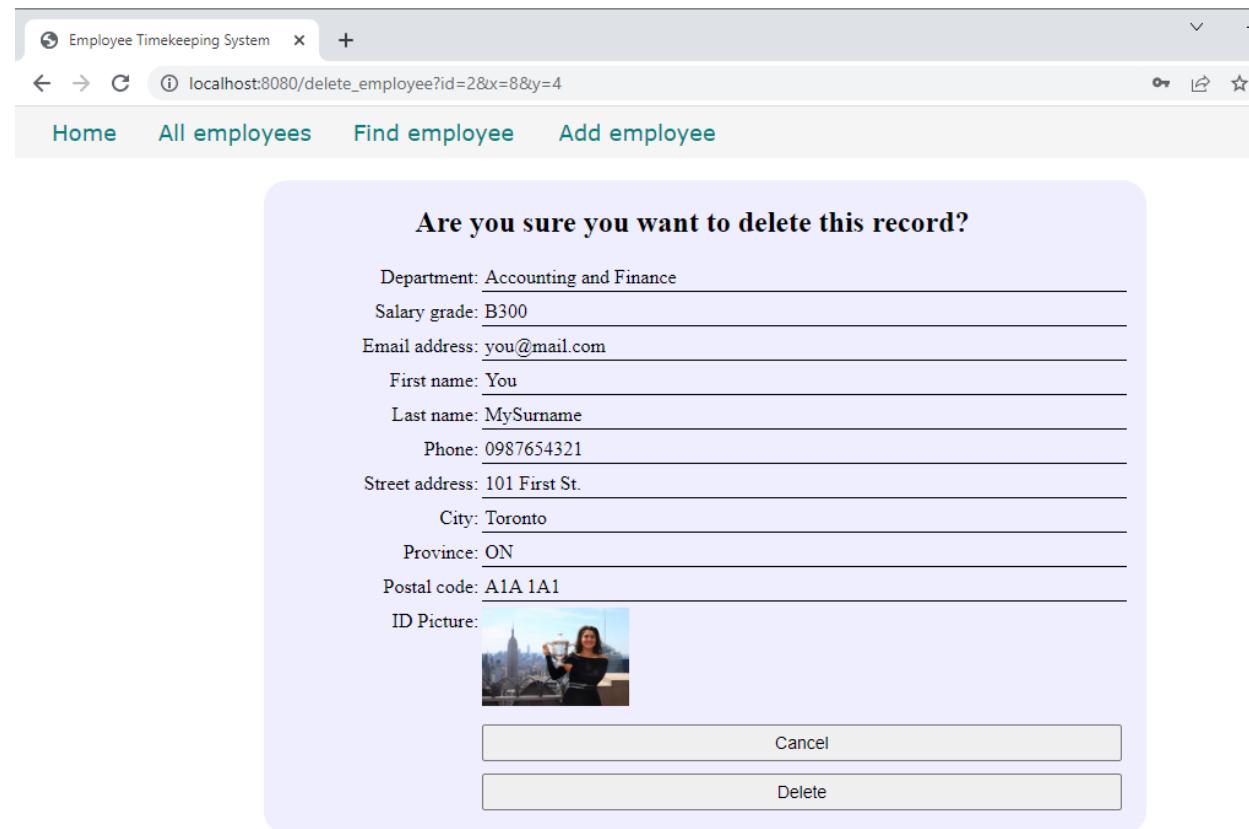
```
extends layout
block maincontent
    include cssformgrid.dt
    div.form-grid-wrapper
        h2.center-align Are you sure you want to delete this record?
        form.form-grid(method="post", action="delete_employee")
            span.form-grid-label Employee number:
            span.form-grid-field #{e.depid}
            span.form-grid-label Department:
            span.form-grid-field #{e.deprt}
            span.form-grid-label Salary grade:
            span.form-grid-field #{e.paygd}
            span.form-grid-label Email address:
            span.form-grid-field #{e.email}
```

```
span.form-grid-label First name:  
span.form-grid-field #{e.fname}  
span.form-grid-label Last name:  
span.form-grid-field #{e.lname}  
span.form-grid-label Phone:  
span.form-grid-field #{e.phone}  
span.form-grid-label Street address:  
span.form-grid-field #{e.street}  
span.form-grid-label City:  
span.form-grid-field #{e.city}  
span.form-grid-label Province:  
span.form-grid-field #{e.province}  
span.form-grid-label Postal code:  
span.form-grid-field #{e.postcode}  
span.form-grid-label ID Picture:  
img(src="#{e.photo}", height="80px")  
input(type="hidden", name="id", value="#{e.id}")  
div  
div  
a(href="all_employees")  
button.form-grid-button(type="button") Cancel  
input.form-grid-button(type="submit", value="Delete")
```

Here we show the employee details and asking the user for confirmation.

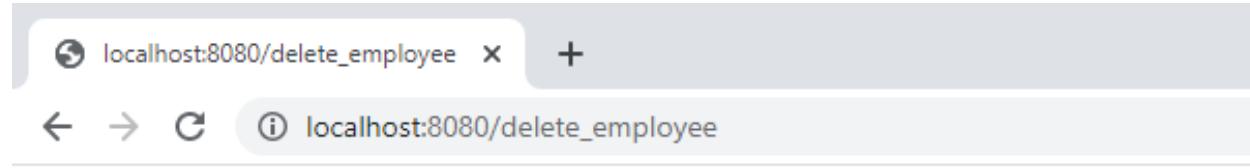
Compile, run and refresh the browser. Show the list of employees again and click on a trash bin icon. The confirmation page should show.

Click Cancel for now just to test if the Cancel button works. Then click the trash bin icon of another record.



This time click the Delete button.

And you get this error:



404 - Not Found

Not Found

Internal error information:  
No routes match path '/delete\_employee'

So let's implement postDeleteEmployee() inside the EmployeeController class.

Edit source\empcontrol.d and append this code.

```
void postDeleteEmployee(int id)
{
    empModel.deleteEmployee(id);
    redirect("all_employees");
}
```

We are calling empModel.deleteEmployee() here, so let's implement that.

Edit source\empmodel.d and append this code.

```
void deleteEmployee(int id)
{
    string sql = "delete from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    conn.exec(pstmt);
}
```

Here we finally delete the record from the database table.

Compile, run and refresh the browser to the list of employees. Click on a trash bin icon, click Delete on the next screen, and you will be redirected to the list of employees again.

This time you should see that the record you selected is no longer listed.

Here is the full source\empmodel.d so far.

```
module empmodel;

import mysql;
import std.conv;
import std.array;

struct Employee
{
    int id; //row id or record id
    string empid; //employee number
    string dept; //department
    string paygd; //salary grade
    string email; //email address
    string pword; //password
    string fname; //first name
    string lname; //last name
    string phone; //phone number
    string photo; //ID photo
    string street; //street address
    string city; //city name
    string province; //province name
    string postcode; //postal code
}
```

```
struct Admin
{
    string email; //email address
    string pword; //password
}

string[] departments =
[
    "Management and Admin",
    "Accounting and Finance",
    "Production",
    "Maintenance",
    "Shipping and Receiving",
    "Purchasing and Supplies",
    "IT Services",
    "Human Resources",
    "Marketing"
];

string[][] provinces =
[
    ["AB", "Alberta"],
    ["BC", "British Columbia"],
    ["MB", "Manitoba"],
    ["NB", "New Brunswick"],
    ["NL", "Newfoundland and Labrador"],
    ["NS", "Nova Scotia"],
    ["NT", "Northwest Territories"],
    ["NU", "Nunavut"],
    ["ON", "Ontario"],
    ["PE", "Prince Edward Island"],
];
```

```
    ["QC", "Quebec"],
    ["SK", "Saskatchewan"],
    ["YT", "Yukon Territory"]
];

string[] paygrades =
[
    "A100", "A200", "A300", "A400",
    "B100", "B200", "B300", "B400",
    "C100", "C200", "C300", "C400",
    "D100", "D200", "D300", "D400",
    "E100", "E200", "E300", "E400",
    "F100", "F200", "F300", "F400"
];

class EmployeeModel
{
    Connection conn;

    this()
    {
        string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";
        conn = new Connection(url);
        scope(exit) conn.close;
    }

    ulong addEmployee(Employee e)
    {
        string sql =
            "insert into employees
            (

```

```
        empid,
        dept, paygd, email, pword,
        fname, lname, phone, photo,
        street, city, province, postcode
    )
    values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
Prepared pstmt = conn.prepare(sql);
pstmt.setArgs
(
    to!string(e.empid),
    to!string(e.deprt),
    to!string(e.paygd),
    to!string(e.email),
    to!string(e.pword),
    to!string(e.fname),
    to!string(e.lname),
    to!string(e.phone),
    to!string(e.photo),
    to!string(e.street),
    to!string(e.city),
    to!string(e.province),
    to!string(e.postcode)
);
return conn.exec(pstmt);
}

Employee[] getEmployees()
{
    import std.array;

    Employee[] emps;
    string sql = "select * from employees";
```

```
Row[] rows = conn.query(sql).array;
if(rows.length == 0) return emps;
return prepareEmployees(rows);
}

Employee[] prepareEmployees(Row[] rows)
{
    Employee[] emps;
    foreach(row; rows)
    {
        Employee e = prepareEmployee(row);
        emps ~= e;
    }
    return emps;
}

Employee prepareEmployee(Row row)
{
    Employee e;
    e.id = to!int(to!string(row[0]));
    e.empid = to!string(row[1]);
    e.deprt = to!string(row[2]);
    e.paygd = to!string(row[3]);
    e.email = to!string(row[4]);
    e.pword = to!string(row[5]);
    e.fname = to!string(row[6]);
    e.lname = to!string(row[7]);
    e.phone = to!string(row[8]);
    e.photo = to!string(row[9]);
    e.street = to!string(row[10]);
    e.city = to!string(row[11]);
    e.province = to!string(row[12]);
```

```
e.postcode = to!string(row[13]);
return e;
}

Employee getEmployee(int id)
{
    import std.array;

    string sql = "select * from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Employee e;
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return e;
    return prepareEmployee(rows[0]);
}

ulong editEmployee(Employee e)
{
    string sql = "update employees set empid=?,
        dept=? , paygd=? , email=? , pword=? ,
        fname=? , lname=? , phone=? , photo=? ,
        street=? , city=? , province=? , postcode=?
        where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs
    (
        to!string(e.empid),
        to!string(e.deprt),
        to!string(e.paygd),
        to!string(e.email),
```

```

        to!string(e.pword),
        to!string(e.fname),
        to!string(e.lname),
        to!string(e.phone),
        to!string(e.photo),
        to!string(e.street),
        to!string(e.city),
        to!string(e.province),
        to!string(e.postcode),
        to!int(to!string(e.id))
    );
    return conn.exec(pstmt);
}

void deleteEmployee(int id)
{
    string sql = "delete from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    conn.exec(pstmt);
}
}

```

And here is the full source\empcontrol.d so far.

```

module empcontrol;

import vibe.vibe;
import empmodel;

class EmployeeController
{

```

```
EmployeeModel empModel;
private string realm = "The Lorem Ipsum Company";

this()
{
    empModel = new EmployeeModel;
}

void index()
{
    render! "index.dt";
}

void getAddEmployee()
{
    render! ("empadd.dt", departments, paygrades, provinces);
}

void postAddEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string photopath = "none yet";
        string ext = extension(pic.filename.name);
        string[] exts = [ ".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
    }
}
```

```

{
    photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
    string dir = "./public/uploads/photos/";
    mkdirRecurse(dir);
    string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
    try moveFile(pic.tempPath, NativePath(fullpath));
    catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
}
e.photo = photopath;
}

if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
e.pword = createDigestPassword(realm, e.email, e.pword);
empModel.addEmployee(e);
redirect("all_employees");
}

void getAllEmployees()
{
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps);
}

void getEditEmployee(int id)
{
    Employee e = empModel.getEmployee(id);
    render!("empedit.dt", e, departments, paygrades, provinces);
}

void postEditEmployee(Employee e)
{

```

```
import std.file;
import std.path;
import std.algorithm;
import vibe.http.auth.digest_auth;

string photopath = e.photo;
auto pic = "picture" in request.files;
if(pic !is null)
{
    string ext = extension(pic.filename.name);
    string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
    if(canFind(exts, ext))
    {
        photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
        string dir = "./public/uploads/photos/";
        mkdirRecurse(dir);
        string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
        try moveFile(pic.tempPath, NativePath(fullpath));
        catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
    }
}
e.photo = photopath;
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
e.pword = createDigestPassword(realm, e.email, e.pword);
empModel.editEmployee(e);
redirect("all_employees");
}

void getDeleteEmployee(int id)
{
```

```
Employee e = empModel.getEmployee(id);
render!("empdelete.dt", e);
}

void postDeleteEmployee(int id)
{
    empModel.deleteEmployee(id);
    redirect("all_employees");
}
```

Now, let's implement that Find employee link on the menu.

## Finding an employee record by name

We have been finding and displaying an employee record already, so this should be trivial to us by now. However, we have always used the id field. This time, we are going to use the name of the employee as the key.

Click on the Find employee menu item to test it. This time, we will find an employee by the first name and last name.

The screenshot shows a web browser window with the following details:

- Address bar: `http://localhost:8080/find_employee#find_employee`
- Content area:
  - Employee navigation: "Employees", "Find employee", "Add employee"
  - Search dialog box:
    - First name: `The`
    - Last name: `Boss`
    - Buttons: "Clear", "Find", "Cancel"
  - Background content:
    - Employee details:
      - First name: `The`
      - Last name: `Boss`
      - Phone: `12345678`
      - Street address: `101 First St`

But when you click on submit, you get this error.



404 - Not Found

Not Found

Internal error information:  
No routes match path '/find\_employee'

It is looking for the postFindEmployee() method, so let's create it.

Edit source\empcontrol.d and append this code.

```
void postFindEmployee(string fname, string lname)
{
    import std.uni; //so we can use toUpper() function

    string first = toUpper(fname);
    string last = toUpper(lname);
    Employee e = empModel.findEmployee(first, last);
    if(e != Employee.init) render!("employee.dt", e);
    else redirect("all_employees");
}
```

We are converting the names to uppercase because we are not sure how the names were inputted and saved in the database.

Edit source\empmodel.d and define this method at the end.

```
Employee findEmployee(string first, string last)
{
    Employee e;
```

```
string sql = "select * from employees where upper(fname)=? and upper(lname)=?";  
Prepared pstmt = conn.prepareStatement(sql);  
pstmt.setArgs(first, last);  
Row[] rows = conn.query(pstmt).array;  
if(rows.length == 0) return e;  
return prepareEmployee(rows[0]);  
}
```

In the SQL statement, we are also converting the names to uppercase before accessing the data. Instead of looking for the row ID, this time the method is looking for the first name and last name converted to uppercase.

Create views\employee.dt.

```
extends layout  
block maincontent  
    include cssformgrid.dt  
    div.form-grid-wrapper  
        h2.center-align Employee details  
        div.form-grid  
            span.form-grid-label Employee number:  
            span.form-grid-field #{e.empid}  
            span.form-grid-label Department:  
            span.form-grid-field #{e.deprt}  
            span.form-grid-label Salary grade:  
            span.form-grid-field #{e.paygd}  
            span.form-grid-label Email address:  
            span.form-grid-field #{e.email}  
            span.form-grid-label First name:  
            span.form-grid-field #{e.fname}  
            span.form-grid-label Last name:  
            span.form-grid-field #{e.lname}  
            span.form-grid-label Phone:
```

```
span.form-grid-field #{e.phone}
span.form-grid-label Street address:
span.form-grid-field #{e.street}
span.form-grid-label City:
span.form-grid-field #{e.city}
span.form-grid-label Province:
span.form-grid-field #{e.province}
span.form-grid-label Postal code:
span.form-grid-field #{e.postcode}
span.form-grid-label ID Picture:

div
div
a\(href="all\_employees"\)
button.form-grid-button(type="button") Close
```

Then compile, run and refresh the browser. Look for an employee and click Find. If the employee was not found, it simply shows the home page.

But if the employee was found, the record will be displayed.

## Employee details

Department: IT Services

Salary grade: A100

Email address: nat@port.com

First name: Nat

Last name: Port

Phone: 3124742

Street address: 102 First St.

City: Toronto

Province: ON

Postal code: A1A 1A2

ID Picture:



[Close](#)

So here is the full source\empcontrol.d so far.

```
module empcontrol;

import vibe.vibe;
import empmodel;

class EmployeeController
{
    EmployeeModel empModel;
    private string realm = "The Lorem Ipsum Company";

    this()
    {
```

```
    empModel = new EmployeeModel;
}

void index()
{
    render!"index.dt";
}

void getAddEmployee()
{
    render!("empadd.dt", departments, paygrades, provinces);
}

void postAddEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string photopath = "none yet";
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
        }
    }
}
```

```
        try moveFile(pic.tempPath, NativePath(fullpath));
        catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
    }
    e.photo = photopath;
}
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
e.pword = createDigestPassword(realm, e.email, e.pword);
empModel.addEmployee(e);
redirect("all_employees");
}

void getAllEmployees()
{
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps);
}

void getEditEmployee(int id)
{
    Employee e = empModel.getEmployee(id);
    render!("empedit.dt", e, departments, paygrades, provinces);
}

void postEditEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;
```

```
string photopath = e.photo;
auto pic = "picture" in request.files;
if(pic !is null)
{
    string ext = extension(pic.filename.name);
    string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
    if(canFind(exts, ext))
    {
        photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
        string dir = "./public/uploads/photos/";
        mkdirRecurse(dir);
        string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
        try moveFile(pic.tempPath, NativePath(fullpath));
        catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
    }
}
e.photo = photopath;
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
e.pword = createDigestPassword(realm, e.email, e.pword);
empModel.editEmployee(e);
redirect("all_employees");
}

void getDeleteEmployee(int id)
{
    Employee e = empModel.getEmployee(id);
    render!("empdelete.dt", e);
}

void postDeleteEmployee(int id)
```

```

{
    empModel.deleteEmployee(id);
    redirect("all_employees");
}

void postFindEmployee(string fname, string lname)
{
    import std.uni; //so we can use the toUpper() or toLower() function

    string first = toUpper(fname);
    string last = toUpper(lname);
    Employee e = empModel.findEmployee(first, last);
    if(e != Employee.init) render!("employee.dt", e);
    else redirect("all_employees");
}
}

```

And here is the full source\empmodel.d so far.

```

module empmodel;

import mysql;
import std.conv;
import std.array;

struct Employee
{
    int id; //row id or record id
    string empid; //employee number
    string dept; //department
    string paygd; //salary grade
    string email; //email address
}

```

```
    string pword; //password
    string fname; //first name
    string lname; //last name
    string phone; //phone number
    string photo; //ID photo
    string street; //street address
    string city; //city name
    string province; //province name
    string postcode; //postal code
}

struct Admin
{
    string email; //email address
    string pword; //password
}

string[] departments =
[
    "Management and Admin",
    "Accounting and Finance",
    "Production",
    "Maintenance",
    "Shipping and Receiving",
    "Purchasing and Supplies",
    "IT Services",
    "Human Resources",
    "Marketing"
];

string[][] provinces =
[
```

```
["AB", "Alberta"],
["BC", "British Columbia"],
["MB", "Manitoba"],
["NB", "New Brunswick"],
["NL", "Newfoundland and Labrador"],
["NS", "Nova Scotia"],
["NT", "Northwest Territories"],
["NU", "Nunavut"],
["ON", "Ontario"],
["PE", "Prince Edward Island"],
["QC", "Quebec"],
["SK", "Saskatchewan"],
["YT", "Yukon Territory"]
];

string[] paygrades =
[
    "A100", "A200", "A300", "A400",
    "B100", "B200", "B300", "B400",
    "C100", "C200", "C300", "C400",
    "D100", "D200", "D300", "D400",
    "E100", "E200", "E300", "E400",
    "F100", "F200", "F300", "F400"
];

class EmployeeModel
{
    Connection conn;

    this()
    {
```

```
string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";
conn = new Connection(url);
scope(exit) conn.close;
}

ulong addEmployee(Employee e)
{
    string sql =
        "insert into employees
        (
            empid,
            dept, paygd, email, pword,
            fname, lname, phone, photo,
            street, city, province, postcode
        )
        values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs
    (
        to!string(e.empid),
        to!string(e.dept),
        to!string(e.paygd),
        to!string(e.email),
        to!string(e.pword),
        to!string(e.fname),
        to!string(e.lname),
        to!string(e.phone),
        to!string(e.photo),
        to!string(e.street),
        to!string(e.city),
        to!string(e.province),
        to!string(e.postcode)
```

```
    );
    return conn.exec(pstmt);
}

Employee[] getEmployees()
{
    Employee[] emps;
    String sql = "select * from employees";
    Row[] rows = conn.query(sql).array;
    if(rows.length == 0) return emps;
    return prepareEmployees(rows);
}

Employee[] prepareEmployees(Row[] rows)
{
    Employee[] emps;
    foreach(row; rows)
    {
        Employee e = prepareEmployee(row);
        emps ~= e;
    }
    return emps;
}

Employee prepareEmployee(Row row)
{
    Employee e;
    e.id = to!int(to!string(row[0]));
    e.empid = to!string(row[1]);
    e.deprt = to!string(row[2]);
    e.paygd = to!string(row[3]);
    e.email = to!string(row[4]);
}
```

```
e.pword = to!string(row[5]);
e.fname = to!string(row[6]);
e.lname = to!string(row[7]);
e.phone = to!string(row[8]);
e.photo = to!string(row[9]);
e.street = to!string(row[10]);
e.city = to!string(row[11]);
e.province = to!string(row[12]);
e.postcode = to!string(row[13]);
return e;
}

Employee getEmployee(int id)
{
    string sql = "select * from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Employee e;
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return e;
    return prepareEmployee(rows[0]);
}

ulong editEmployee(Employee e)
{
    string sql = "update employees set empid=?,
        deprt=?, paygd=?, email=?, pword=?,
        fname=?, lname=?, phone=?, photo=?,
        street=?, city=?, province=?, postcode=?
        where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs
```

```
(  
    to!string(e.empid),  
    to!string(e.deprt),  
    to!string(e.paygd),  
    to!string(e.email),  
    to!string(e.pword),  
    to!string(e.fname),  
    to!string(e.lname),  
    to!string(e.phone),  
    to!string(e.photo),  
    to!string(e.street),  
    to!string(e.city),  
    to!string(e.province),  
    to!string(e.postcode),  
    to!int(to!string(e.id))  
);  
    return conn.exec(pstmt);  
}  
  
void deleteEmployee(int id)  
{  
    string sql = "delete from employees where id=?";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs(id);  
    conn.exec(pstmt);  
}  
  
Employee findEmployee(string first, string last)  
{  
    Employee e;  
    string sql = "select * from employees where upper(fname)=? and upper(lname)=?";  
    Prepared pstmt = conn.prepare(sql);
```

```
        pstmt.setArgs(first, last);
        Row[] rows = conn.query(pstmt).array;
        if(rows.length == 0) return e;
        return prepareEmployee(rows[0]);
    }
}
```

Now we have gone through the adding, viewing, editing and deleting of records. Meaning, we have completed the CRUD (create, read, update, delete) operations.

Now let's talk about error trapping.

## Capturing and displaying (some) error messages with \_error

When a new employee record failed to be added to the database because of some error, that error should be shown to the user to let the user know why the insertion failed so the user can try to resolve the problem.

When an update attempt failed, the error should also be shown to let the user know the state of things.

After a failed search when the employee was not found, the message should be also be displayed.

When a login attempt failed, the login page should display again with an error message telling the user why the login attempt failed.

When a method in EmployeeController generates an error, the error message is captured in the `_error` variable which is generated automatically, so we can use it to display the error. The facility to display this `_error` message is provided by the `@errorDisplay` annotation. The `@errorDisplay` indicates which method and page will handle the error.

As an example, let us edit the `postFindEmployee()` method of EmployeeController.

```
@errorDisplay!getAllEmployees
void postFindEmployee(string fname, string lname)
{
    import std.uni; //so we can use the toUpper() function

    string first = toUpper(fname);
    string last = toUpper(lname);
    Employee e = empModel.findEmployee(first, last);
    enforce(e != Employee.init, fname ~ " " ~ lname ~ " not found.");
    render!("employee.dt", e);
}
```

Here, we indicated that the getAllEmployees() method will handle the error. We used the enforce() function to generate an error if the condition was not met. In this case, the getAllEmployees() method will be called with a new parameter named \_error, which was generated automatically, and which it can pass to the emplistall.dt page for display.

So here is the edited getAllEmployees() method.

```
void getAllEmployees(string _error = null)
{
    string error = _error;
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps, error);
}
```

We assigned the \_error parameter with null as default in case there is no error message.

Edit the views\emplistall.dt page so it can display the error, if there is an error.

```
extends layout
block maincontent
    include csstable.dt
    -if(error)
        div.error-div
            span.error-message #{error}
    div.table-wrapper
        table
            tr
                th Employee Id
                th First name
                th Last name
                th Department
                th Phone number
                th Email address
```

```

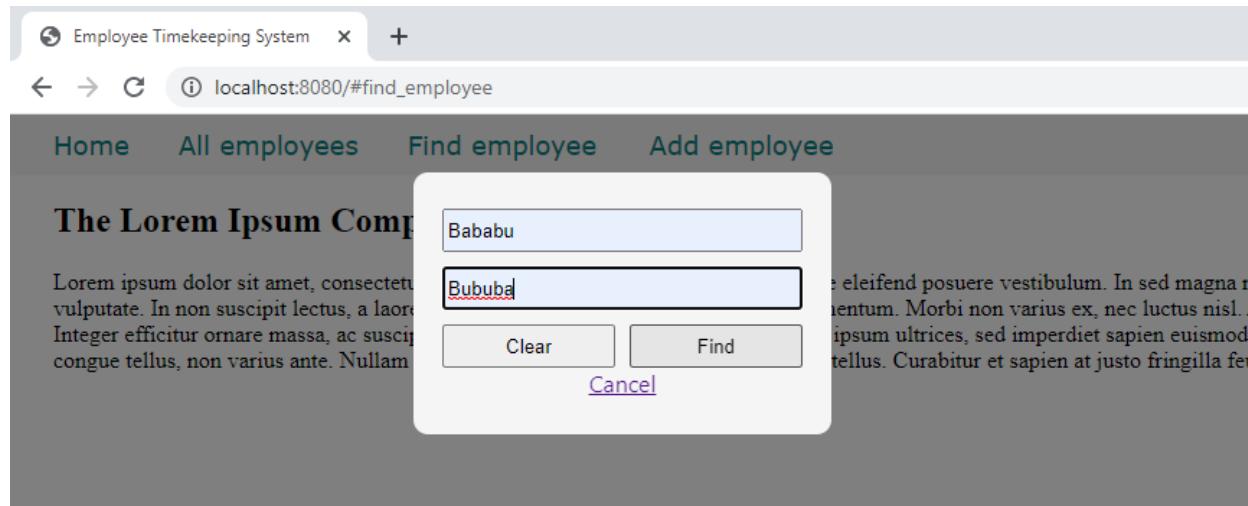
th Street address
th City
th Province
th PostCode
th Action
-foreach(e; emps)
tr
td #{e.empid}
td #{e.fname}
td #{e.lname}
td #{e.deprt}
td #{e.phone}
td #{e.email}
td #{e.street}
td #{e.city}
td #{e.province}
td #{e.postcode}
td &nbs;
form.form-hidden(method="get", action="edit_employee")
input(type="hidden", name="id", value="#{e.id}")
input(type="image", src="images/pencil.ico", height="15px")
| &nbs;
form.form-hidden(method="get", action="delete_employee")
input(type="hidden", name="id", value="#{e.id}")
input(type="image", src="images/trash.ico", height="15px")
| &nbs;
```

And here is our addition to the views\styles.dt file.

```
.error-div
{
    width: 90%;
```

```
    margin: 20px auto;
}
.error-message
{
    color: brown;
    font-weight: bold;
}
```

Compile, run and refresh your browser to test the Find employee link and look for a non-existent record.



After clicking Find, you should get this:

<a href="#">Home</a>	<a href="#">All employees</a>	<a href="#">Find employee</a>	<a href="#">Add employee</a>
<b>Bababu Bububa not found.</b>			
Employee Id	First name	Last name	Department
0001	The	Boss	Management and Admin
0002	Gal	Gadot	Accounting and Finance
0003	Eugene	Bou	Accounting and Finance
Phone number	Email address	Street address	City
12345678	theboss@mail.com	101 First St.	Toronto
0987654321	gal@gadot.com	102 First St.	Toronto
31247425645	ebouchard@mail.com	103 First St.	Toronto
Province			

Now we can add error-trapping mechanisms to the adding, editing and deleting methods of the EmployeeController. Since there are many methods involved, here is the full source\empcontrol.d after adding error-trapping code.

```
module empcontrol;

import vibe.vibe;
import empmodel;

class EmployeeController
{
    EmployeeModel empModel;
    private string realm = "The Lorem Ipsum Company";

    this()
    {
        empModel = new EmployeeModel();
    }

    void index()
    {
        render!"index.dt";
    }

    void getAddEmployee(string _error = null)
    {
```

```
    string error = _error;
    render!("empadd.dt", departments, paygrades, provinces, error);
}

@errorDisplay!getAddEmployee
void postAddEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string photopath = "none yet";
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
            try moveFile(pic.tempPath, NativePath(fullpath));
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
        }
        e.photo = photopath;
    }
    if(e.phone.length == 0) e.phone = "(123) 456 7890";
    if(e.paygd.length == 0) e.paygd = "none yet";
    if(e.postcode.length == 0) e.postcode = "A1A 1A1";
}
```

```
e.pword = createDigestPassword(realm, e.email, e.pword);
empModel.addEmployee(e);
redirect("all_employees");
}

void getAllEmployees(string _error = null)
{
    string error = _error;
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps, error);
}

void getEditEmployee(int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    render!("empedit.dt", e, departments, paygrades, provinces, error);
}

@errorDisplay!getEditEmployee
void postEditEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    string photopath = e.photo;
    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string ext = extension(pic.filename.name);
```

```
string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
if(canFind(exts, ext))
{
    photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
    string dir = "./public/uploads/photos/";
    mkdirRecurse(dir);
    string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
    try moveFile(pic.tempPath, NativePath(fullpath));
    catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
}
e.photo = photopath;
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
e.pword = createDigestPassword(realm, e.email, e.pword);
empModel.editEmployee(e);
redirect("all_employees");
}

void getDeleteEmployee(int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    render!("empdelete.dt", e, error);
}

@errorDisplay!getDeleteEmployee
void postDeleteEmployee(int id)
{
    empModel.deleteEmployee(id);
    redirect("all_employees");
```

```
}  
  
@errorDisplay!getAllEmployees  
void postFindEmployee(string fname, string lname)  
{  
    import std.uni; //so we can use the toUpper() function  
  
    string first = toUpper(fname);  
    string last = toUpper(lname);  
    Employee e = empModel.findEmployee(first, last);  
    enforce(e != Employee.init, fname ~ " " ~ lname ~ " not found.");  
    render!("employee.dt", e);  
}  
}
```

We made no changes to the source\empmodel.d, but here it is so far.

```
module empmodel;  
  
import mysql;  
import std.conv;  
import std.array;  
  
struct Employee  
{  
    int id; //row id or record id  
    string empid; //employee number  
    string dept; //department  
    string paygd; //salary grade  
    string email; //email address  
    string pword; //password  
    string fname; //first name
```

```
    string lname; //last name
    string phone; //phone number
    string photo; //ID photo
    string street; //street address
    string city; //city name
    string province; //province name
    string postcode; //postal code
}

struct Admin
{
    string email; //email address
    string pword; //password
}

string[] departments =
[
    "Management and Admin",
    "Accounting and Finance",
    "Production",
    "Maintenance",
    "Shipping and Receiving",
    "Purchasing and Supplies",
    "IT Services",
    "Human Resources",
    "Marketing"
];

string[][] provinces =
[
    ["AB", "Alberta"],
    ["BC", "British Columbia"],
```

```
[ "MB", "Manitoba"],
[ "NB", "New Brunswick"],
[ "NL", "Newfoundland and Labrador"],
[ "NS", "Nova Scotia"],
[ "NT", "Northwest Territories"],
[ "NU", "Nunavut"],
[ "ON", "Ontario"],
[ "PE", "Prince Edward Island"],
[ "QC", "Quebec"],
[ "SK", "Saskatchewan"],
[ "YT", "Yukon Territory"]
];

string[] paygrades =
[
    "A100", "A200", "A300", "A400",
    "B100", "B200", "B300", "B400",
    "C100", "C200", "C300", "C400",
    "D100", "D200", "D300", "D400",
    "E100", "E200", "E300", "E400",
    "F100", "F200", "F300", "F400"
];

class EmployeeModel
{
    Connection conn;

    this()
    {
        string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";
        conn = new Connection(url);
    }
}
```

```
    scope(exit) conn.close;
}

ulong addEmployee(Employee e)
{
    string sql =
        "insert into employees
        (
            empid,
            dept, paygd, email, pword,
            fname, lname, phone, photo,
            street, city, province, postcode
        )
        values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs
    (
        to!string(e.empid),
        to!string(e.dept),
        to!string(e.paygd),
        to!string(e.email),
        to!string(e.pword),
        to!string(e.fname),
        to!string(e.lname),
        to!string(e.phone),
        to!string(e.photo),
        to!string(e.street),
        to!string(e.city),
        to!string(e.province),
        to!string(e.postcode)
    );
    return conn.exec(pstmt);
```

```
}

Employee[] getEmployees()
{
    Employee[] emps;
    string sql = "select * from employees";
    Row[] rows = conn.query(sql).array;
    if(rows.length == 0) return emps;
    return prepareEmployees(rows);
}

Employee[] prepareEmployees(Row[] rows)
{
    Employee[] emps;
    foreach(row; rows)
    {
        Employee e = prepareEmployee(row);
        emps ~= e;
    }
    return emps;
}

Employee prepareEmployee(Row row)
{
    Employee e;
    e.id = to!int(to!string(row[0]));
    e.empid = to!string(row[1]);
    e.deprt = to!string(row[2]);
    e.paygd = to!string(row[3]);
    e.email = to!string(row[4]);
    e.pword = to!string(row[5]);
    e.fname = to!string(row[6]);
}
```

```
e.lname = to!string(row[7]);
e.phone = to!string(row[8]);
e.photo = to!string(row[9]);
e.street = to!string(row[10]);
e.city = to!string(row[11]);
e.province = to!string(row[12]);
e.postcode = to!string(row[13]);
return e;
}

Employee getEmployee(int id)
{
    string sql = "select * from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Employee e;
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return e;
    return prepareEmployee(rows[0]);
}

ulong editEmployee(Employee e)
{
    string sql = "update employees set empid=?,
        dept=? , paygd=? , email=? , pword=? ,
        fname=? , lname=? , phone=? , photo=? ,
        street=? , city=? , province=? , postcode=?
        where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs
    (
        to!string(e.empid),
```

```
    to!string(e.deprt),
    to!string(e.paygd),
    to!string(e.email),
    to!string(e.pword),
    to!string(e.fname),
    to!string(e.lname),
    to!string(e.phone),
    to!string(e.photo),
    to!string(e.street),
    to!string(e.city),
    to!string(e.province),
    to!string(e.postcode),
    to!int(to!string(e.id))
);
return conn.exec(pstmt);
}

void deleteEmployee(int id)
{
    string sql = "delete from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    conn.exec(pstmt);
}

Employee findEmployee(string first, string last)
{
    Employee e;
    string sql = "select * from employees where upper(fname)=? and upper(lname)=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(first, last);
    Row[] rows = conn.query(pstmt).array;
```

```

        if(rows.length == 0) return e;
        return prepareEmployee(rows[0]);
    }
}

```

And here is the views\empadd.dt file.

```

extends layout
block maincontent
    include cssformgrid.dt
    -if(error)
        div.error-div
            span.error-message #{error}
div.form-grid-wrapper
    h2.center-align New employee details
    form.form-grid(method="post", action="add_employee", enctype="multipart/form-data")
        label.form-grid-label Employee number
        input.form-grid-input(type="empid", name="e.empid", placeholder="employee number", required)
        label.form-grid-label Department
        select#deprt.form-grid-input(name="e.deprt")
            -foreach(dep; departments)
                option(value="#{dep}") #{dep}
        label.form-grid-label Salary grade
        select#deprt.form-grid-input(name="e.paygd")
            -foreach(pay; paygrades)
                option(value="#{pay}") #{pay}
        label.form-grid-label Email address
        input.form-grid-input(type="email", name="e_email", placeholder="email@company.com", required)
        label.form-grid-label Password
        input.form-grid-input(type="password", name="e_pword", placeholder="password", required)
        label.form-grid-label First name
        input.form-grid-input(type="text", name="e_fname", placeholder="First name", required)

```

```

label.form-grid-label Last name
input.form-grid-input(type="text", name="e_lname", placeholder="Last name", required)
label.form-grid-label Phone
input.form-grid-input(type="text", name="e_phone", placeholder="Phone number")
label.form-grid-label Street address (no city)
input.form-grid-input(type="text", name="e_street", placeholder="Street address", required)
label.form-grid-label City
input.form-grid-input(type="text", name="e_city", placeholder="City", required)
label.form-grid-label Province
select#province.form-grid-input(name="e_province")
-foreach(prov; provinces)
    option(value="#{prov[0]}") #{prov[1]}
label.form-grid-label Postal code
input.form-grid-input(type="text", name="e_postcode", placeholder="A1A 1A1")
label.form-grid-label ID Picture
input.form-grid-input(type="file", name="picture")
input(type="hidden", name="e_photo")
input(type="hidden", name="e_id", value="1")
div
div
    input.form-grid-button(type="reset", value="Clear the form")
    input.form-grid-button(type="submit", value="Submit")

```

And here is the views\empdelete.dt file.

```

extends layout
block maincontent
    include cssformgrid.dt
    -if(error)
        div.error-div
            span.error-message #{error}
    div.form-grid-wrapper

```

```
h2.center-align Are you sure you want to delete this record?  
form.form-grid(method="post", action="delete_employee")  
    span.form-grid-label Employee number:  
    span.form-grid-field #{e.empid}  
    span.form-grid-label Department:  
    span.form-grid-field #{e.deprt}  
    span.form-grid-label Salary grade:  
    span.form-grid-field #{e.paygd}  
    span.form-grid-label Email address:  
    span.form-grid-field #{e.email}  
    span.form-grid-label First name:  
    span.form-grid-field #{e.fname}  
    span.form-grid-label Last name:  
    span.form-grid-field #{e.lname}  
    span.form-grid-label Phone:  
    span.form-grid-field #{e.phone}  
    span.form-grid-label Street address:  
    span.form-grid-field #{e.street}  
    span.form-grid-label City:  
    span.form-grid-field #{e.city}  
    span.form-grid-label Province:  
    span.form-grid-field #{e.province}  
    span.form-grid-label Postal code:  
    span.form-grid-field #{e.postcode}  
    span.form-grid-label ID Picture:  
    img(src="#{e.photo}", height="80px")  
    input(type="hidden", name="id", value="#{e.id}")  
div  
div  
    a(href="all_employees")  
        button.form-grid-button(type="button") Cancel  
        input.form-grid-button(type="submit", value="Delete")
```

And here is the views\empedit.dt file.

```
extends layout
block maincontent
    include cssformgrid.dt
    -if(error)
        div.error-div
            span.error-message #{error}
    div.form-grid-wrapper
        h2.center-align Edit employee details
        form.form-grid(method="post", action="edit_employee", enctype="multipart/form-data")
            label.form-grid-label Employee number
            input.form-grid-input(type="empid", name="e.empid", value="#{e.empid}")
            label.form-grid-label Department
            select#dept.form-grid-input(name="e.deprt", value="#{e.deprt}")
                -foreach(dep; departments)
                    -if(dep == e.deprt)
                        option(value="#{dep}", selected) #{dep}
                    -else
                        option(value="#{dep}") #{dep}
            label.form-grid-label Salary grade
            select#paygd.form-grid-input(name="e.paygd", value="#{e.paygd}")
                -foreach(pay; paygrades)
                    -if(pay == e.paygd)
                        option(value="#{pay}", selected) #{pay}
                    -else
                        option(value="#{pay}") #{pay}
            label.form-grid-label Email address
            input.form-grid-input(type="email", name="e_email", value="#{e.email}")
            label.form-grid-label Password
            input.form-grid-input(type="password", name="e.pword", value="#{e.pword}")
            label.form-grid-label First name
```

```

input.form-grid-input(type="text", name="e_fname", value="#{e.fname}")
label.form-grid-label Last name
input.form-grid-input(type="text", name="e_lname", value="#{e.lname}")
label.form-grid-label Phone
input.form-grid-input(type="text", name="e_phone", value="#{e.phone}")
label.form-grid-label Street address (no city)
input.form-grid-input(type="text", name="e_street", value="#{e.street}")
label.form-grid-label City
input.form-grid-input(type="text", name="e_city", value="#{e.city}")
label.form-grid-label Province
select#province.form-grid-input(name="e_province", value="#{e.province}")
-foreach(prov; provinces)
-if(prov[0] == e.province)
    option(value="#{prov[0]}", selected) #{prov[1]}
-else
    option(value="#{prov[0]}") #{prov[1]}
label.form-grid-label Postal code
input.form-grid-input(type="text", name="e_postcode", value="#{e.postcode}")
label.form-grid-label ID Picture
img(src="#{e.photo}", height="100px")
div
input.form-grid-input(type="file", name="picture")
input(type="hidden", name="e_photo", value="#{e.photo}")
input(type="hidden", name="e_id", value="#{e.id}")
input(type="hidden", name="id", value="#{e.id}")
div
div
a(href="all_employees")
    button.form-grid-button(type="button") Cancel
    input.form-grid-button(type="submit", value="Submit")

```

And the views\emplistall.dt file.

```
extends layout
block maincontent
  include csstable.dt
  -if(error)
    div.error-div
      span.error-message #{error}
  div.table-wrapper
    table
      tr
        th Employee Id
        th First name
        th Last name
        th Department
        th Phone number
        th Email address
        th Street address
        th City
        th Province
        th PostCode
        th Action
      -foreach(e; emps)
        tr
          td #{e.empid}
          td #{e.fname}
          td #{e.lname}
          td #{e.deprt}
          td #{e.phone}
          td #{e.email}
          td #{e.street}
          td #{e.city}
          td #{e.province}
```

```
td #{e.postcode}
td &nbsp;
  form.form-hidden(method="get", action="edit_employee")
    input(type="hidden", name="id", value="#{e.id}")
    input(type="image", src="images/pencil.ico", height="15px")
  | &nbsp;
  form.form-hidden(method="get", action="delete_employee")
    input(type="hidden", name="id", value="#{e.id}")
    input(type="image", src="images/trash.ico", height="15px")
  | &nbsp;
```

Now let's talk about securing our site so that only authorized users can access or make changes to the data.

## Logging in, authentication and authorization

We need to restrict access to sensitive parts of our site so only authorized users (admins) can make changes to sensitive data. We need a login system to authenticate and to authorize users.

Authentication means verifying that the user is who he/she says he/she is. Authorization means determining which parts of the application the user is allowed access.

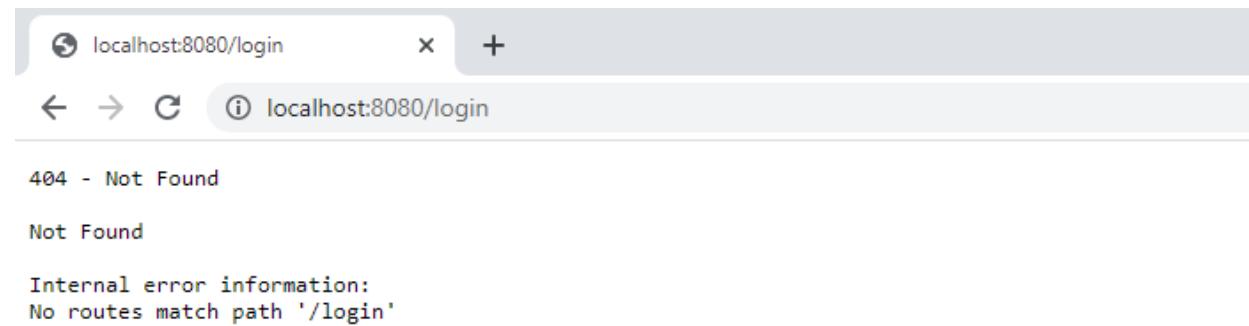
In a commercial or government institution, only a limited number of people are supposed to access sensitive data such as employee records.

To achieve this, we need a login system to authenticate and authorize users.

Click on the Login link on the menu to show the login form and fill out with an existing data.



Of course, when you click the Login button, you get an error:



Let us fix that. Let us do the authentication part of the login system.

## Authenticating the user

We are going to simplify things. We simply match the email and password combination received from the form to what we have on the admins table to verify the user's identity. Since the saved password is encrypted, we also have to encrypt the raw password received from the form before comparing them.

Edit source\empcontrol.d and add this method.

```
@errorDisplay!index
void postLogin(string email, string password)
{
    import vibe.http.auth.digest_auth;

    auto scrambled = createDigestPassword(realname, email, password);
    bool isAdmin = empModel.isAdmin(email, scrambled);
    enforce(isAdmin, "Email and password combination not found.");
    redirect("all_employees");
}
```

We used the enforce() function before. The enforce() function is a built-in D function that does nothing if the first parameter is true, but if it is false, it aborts and returns the error message.

Edit source\empmodel.d and add this method.

```
bool isAdmin(string email, string password)
{
    string sql = "select * from admins where email=? and pword=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(email, password);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return false;
    return true;
```

```
}
```

Compile, run and refresh the browser, then try to log in with a random, non-existent admin first.

The screenshot shows a web application interface. On the left, there is a table titled 'Add employee' with columns: grade, Phone number, Email address, Street address, and City. The table contains five rows of data. On the right, there is a 'Login' dialog box with fields for 'Email address' (containing 'yourid@mail.com') and 'Password' (containing '.....'). Below the fields are 'Clear', 'Login', and 'Cancel' buttons. The 'Login' button is highlighted with a blue border.

grade	Phone number	Email address	Street address	City
	3124742	nat@port.com	102 First St.	Toronto
	0987654321	yourid@mail.com	103 First St.	Toronto
	12345678	gal@gadot.com	104 First St.	Toronto
	0987654321	leigh@meester.com	105 First St.	Toronto
	12345678	alex@dadda.com	106 First St.	Toronto

And click on the Login button. The index page with an error message should be shown.

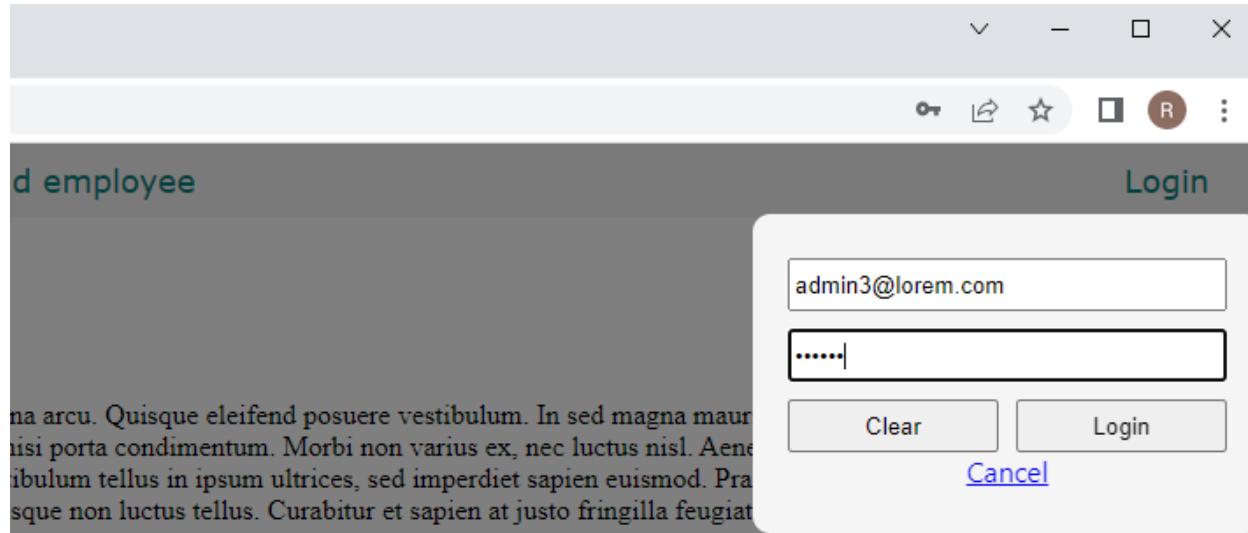
The screenshot shows a browser window for the 'Employee Timekeeping System'. The title bar says 'Employee Timekeeping System'. The address bar shows 'localhost:8080/login'. The navigation bar includes links for 'Home', 'All employees', 'Find employee', and 'Add employee'. The main content area displays the error message 'Email and password combination not found.' in red text. Below the message is a large block of placeholder text in Latin.

The Lorem Ipsum Company

Email and password combination not found.

...  
Nam nec urna arcu. Quisque eleifend posuere  
vulputate. In non suscipit lectus, a laoreet odio. Donec at sapien eu nisi porta condimentum. Morbi no:  
Integer efficitur ornare massa, ac suscipit enim sagittis et. Proin vestibulum tellus in ipsum ultrices, se  
congue tellus, non varius ante. Nullam tincidunt dolor felis. Pellentesque non luctus tellus. Curabitur e

Then click on the login link again and input an existing admin and try to login (remember, ‘secret’ is the password).



If the list of employees is shown, you did good.

Now let's talk about saving the login state.

## Saving the login state to the session

A session is when a user logs in up to when the user logs out. Vibe.d also provides a mechanism to save things to a session.

When we want to retrieve a particular record from the database, we simply use the id field as the key to retrieve the record by passing the key to the next page. When we retrieve another record, we lose this key because we receive another key.

But when a user logs in, the log in state needs to be saved for the duration of the session so that the user can navigate from page to page without having to log in again and again.

If the user is logged in, we should save that state so that as the user navigates from page to page, the system can check if the user is already logged in and is authorized to access that particular page. We need a session for that.

A variable, such as a logged-in state, can be saved to this session facility. Vibe.d has the session store for this purpose, of which there are two kinds: MemorySessionStore and the database-based store. For this project, we will use the MemorySessionStore kind.

Edit source\app.d.

```
import vibe.vibe;
import empcontrol;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];
    settings.sessionStore = new MemorySessionStore;

    auto router = new URLRouter;
    router.get("*", serveStaticFiles("public/"));
    router.registerWebInterface(new EmployeeController);
```

```
    auto listener = listenHTTP(settings, router);
    scope (exit) listener.stopListening();

    runApplication();
}
```

Edit source\empcontrol.d to add a User struct.

```
module empcontrol;

import vibe.vibe;
import empmodel;

struct User
{
    bool loggedIn;
    string email;
}

class EmployeeController
{
    EmployeeModel empModel;
    string realm = "The Lorem Ipsum Company";
    private SessionVar!(User, "user") m_user;

...
```

And edit postLogin() to save the logged-in state to the session.

```
@errorDisplay!index
void postLogin(string email, string password)
```

```
{  
    import vibe.http.auth.digest_auth;  
  
    auto scrambled = createDigestPassword(realm, email, password);  
    bool isAdmin = empModel.isAdmin(email, scrambled);  
    enforce(isAdmin, "Email and password combination not found.");  
    User user = m_user;  
    user.loggedIn = true;  
    user.email = email;  
    m_user = user;  
    redirect("all_employees");  
}
```

We created a User struct to hold the logged-in state

```
struct User  
{  
    bool loggedIn;  
    string email;  
}
```

Then we declared a session variable named `m_user` of that struct type:

```
private SessionVar!(User, "user") m_user;
```

then we changed the value of `m_user` inside `postLogin()` to save and start the session:

```
User user = m_user;  
user.loggedIn = true;  
user.email = email;  
m_user = user;
```

## Changing the value of `m_user` automatically starts the session.

We indicated that `index()` will be called and passed the error if `postLogin()` encounters an error.

```
@errorDisplay!index
void postLogin(string email, string password)
{
    import vibe.http.auth.digest_auth;
...
}
```

Then we changed the `index()` method to accept an `_error` variable from the `postLogin()` method if it encounters an error.

```
void index(string _error = null)
{
    string error = _error;
    render!("index.dt", error);
}
```

Edit `views\index.dt` to display the error, if there is any, returned by the `enforce()` function.

```
extends layout
block maincontent
    h2 The Lorem Ipsum Company
    -if(error)
        div.error-div
            span.error-message #{error}
    div.
        Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Nam nec urna arcu. Quisque eleifend posuere vestibulum.
        In sed magna mauris. Phasellus bibendum ligula et placerat
        vulputate. In non suscipit lectus, a laoreet odio. Donec
        at sapien eu nisi porta condimentum. Morbi non varius ex,
```

```
nec luctus nisl. Aenean varius dui quis arcu auctor luctus.  
Integer efficitur ornare massa, ac suscipit enim sagittis et.  
Proin vestibulum tellus in ipsum ultrices, sed imperdiet  
sapien euismod. Praesent vel facilisis mauris. Proin finibus  
congue tellus, non varius ante. Nullam tincidunt dolor felis.  
Pellentesque non luctus tellus. Curabitur et sapien at justo  
fringilla feugiat et a erat.  
<br /><br />
```

Compile, run and refresh the browser. Click on the Login menu item and input a non-existing admin user. If you get an error, that means the error-trapping mechanism is working.

The screenshot shows a web browser window with the following details:

- Title Bar:** Employee Timekeeping System
- Address Bar:** localhost:8080/login
- Navigation Bar:** Home, All employees, Find employee, Add employee
- Main Content:**

**The Lorem Ipsum Company**

Email and password combination not found.

... (The rest of the page content is obscured by a large black redaction box.)

The Lorem Ipsum Company

Email and password combination not found.

... (The rest of the page content is obscured by a large black redaction box.)

So here is the complete source\empcontrol.d so far:

```
module empcontrol;
```

```
import vibe.vibe;
import empmodel;

struct User
{
    bool loggedIn;
    string email;
}

class EmployeeController
{
    private EmployeeModel empModel;
    private string realm = "The Lorem Ipsum Company";
    private SessionVar!(User, "user") m_user;

    this()
    {
        empModel = new EmployeeModel();
    }

    void index(string _error = null)
    {
        string error = _error;
        render!("index.dt", error);
    }

    void getAddEmployee(string _error = null)
    {
        string error = _error;
        render!("empadd.dt", departments, paygrades, provinces, error);
    }
}
```

```
@errorDisplay!getAddEmployee
void postAddEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string photopath = "none yet";
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
            try moveFile(pic.tempPath, NativePath(fullpath));
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
        }
        e.photo = photopath;
    }
    if(e.phone.length == 0) e.phone = "(123) 456 7890";
    if(e.paygd.length == 0) e.paygd = "none yet";
    if(e.postcode.length == 0) e.postcode = "A1A 1A1";
    e.pword = createDigestPassword(realm, e.email, e.pword);
    empModel.addEmployee(e);
    redirect("all_employees");
}
```

```
void getAllEmployees(string _error = null)
{
    string error = _error;
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps, error);
}

void getEditEmployee(int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    render!("empedit.dt", e, departments, paygrades, provinces, error);
}

@errorDisplay!getEditEmployee
void postEditEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    string photopath = e.photo;
    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
        }
    }
}
```

```
        string dir = "./public/uploads/photos/";
        mkdirRecurse(dir);
        string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
        try moveFile(pic.tempPath, NativePath(fullpath));
        catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
    }
}

e.photo = photopath;
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
e.pword = createDigestPassword(realm, e.email, e.pword);
empModel.editEmployee(e);
redirect("all_employees");
}

void getDeleteEmployee(int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    render!("empdelete.dt", e, error);
}

@errorDisplay!getDeleteEmployee
void postDeleteEmployee(int id)
{
    empModel.deleteEmployee(id);
    redirect("all_employees");
}

@errorDisplay!getAllEmployees
void postFindEmployee(string fname, string lname)
```

```

{
    import std.uni; //so we can use the toUpper() function

    string first = toUpper(fname);
    string last = toUpper(lname);
    Employee e = empModel.findEmployee(first, last);
    enforce(e != Employee.init, fname ~ " " ~ lname ~ " not found.");
    render!("employee.dt", e);
}

@errorDisplay!index
void postLogin(string email, string password)
{
    import vibe.http.auth.digest_auth;

    auto scrambled = createDigestPassword(realm, email, password);
    bool isAdmin = empModel.isAdmin(email, scrambled);
    enforce(isAdmin, "Email and password combination not found.");
    User user = m_user;
    user.loggedIn = true;
    user.email = email;
    m_user = user;
    redirect("all_employees");
}
}

```

And here is source\empmodel.d so far:

```

module empmodel;

import mysql;
import std.conv;

```

```
import std.array;

struct Employee
{
    int id; //row id or record id
    string empid; //employee number
    string dept; //department
    string paygd; //salary grade
    string email; //email address
    string pword; //password
    string fname; //first name
    string lname; //last name
    string phone; //phone number
    string photo; //ID photo
    string street; //street address
    string city; //city name
    string province; //province name
    string postcode; //postal code
}

struct Admin
{
    string email; //email address
    string pword; //password
}

string[] departments =
[
    "Management and Admin",
    "Accounting and Finance",
    "Production",
    "Maintenance",
```

```
    "Shipping and Receiving",
    "Purchasing and Supplies",
    "IT Services",
    "Human Resources",
    "Marketing"
];

string[][] provinces =
[
    ["AB", "Alberta"],
    ["BC", "British Columbia"],
    ["MB", "Manitoba"],
    ["NB", "New Brunswick"],
    ["NL", "Newfoundland and Labrador"],
    ["NS", "Nova Scotia"],
    ["NT", "Northwest Territories"],
    ["NU", "Nunavut"],
    ["ON", "Ontario"],
    ["PE", "Prince Edward Island"],
    ["QC", "Quebec"],
    ["SK", "Saskatchewan"],
    ["YT", "Yukon Territory"]
];

string[] paygrades =
[
    "A100", "A200", "A300", "A400",
    "B100", "B200", "B300", "B400",
    "C100", "C200", "C300", "C400",
    "D100", "D200", "D300", "D400",
    "E100", "E200", "E300", "E400",
    "F100", "F200", "F300", "F400"
]
```

```
];

class EmployeeModel
{
    Connection conn;

    this()
    {
        string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";
        conn = new Connection(url);
        scope(exit) conn.close;
    }

    ulong addEmployee(Employee e)
    {
        string sql =
            "insert into employees
            (
                empid,
                dept, paygd, email, pword,
                fname, lname, phone, photo,
                street, city, province, postcode
            )
            values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
        Prepared pstmt = conn.prepare(sql);
        pstmt.setArgs
        (
            to!string(e.empid),
            to!string(e.dept),
            to!string(e.paygd),
            to!string(e.email),
            to!string(e.pword),
```

```
        to!string(e.fname),
        to!string(e.lname),
        to!string(e.phone),
        to!string(e.photo),
        to!string(e.street),
        to!string(e.city),
        to!string(e.province),
        to!string(e.postcode)
    );
    return conn.exec(pstmt);
}

Employee[ ] getEmployees()
{
    Employee[ ] emps;
    string sql = "select * from employees";
    Row[ ] rows = conn.query(sql).array;
    if(rows.length == 0) return emps;
    return prepareEmployees(rows);
}

Employee[ ] prepareEmployees(Row[ ] rows)
{
    Employee[ ] emps;
    foreach(row; rows)
    {
        Employee e = prepareEmployee(row);
        emps ~= e;
    }
    return emps;
}
```

```
Employee prepareEmployee(Row row)
{
    Employee e;
    e.id = to!int(to!string(row[0]));
    e.empid = to!string(row[1]);
    e.deprt = to!string(row[2]);
    e.paygd = to!string(row[3]);
    e.email = to!string(row[4]);
    e.pword = to!string(row[5]);
    e.fname = to!string(row[6]);
    e.lname = to!string(row[7]);
    e.phone = to!string(row[8]);
    e.photo = to!string(row[9]);
    e.street = to!string(row[10]);
    e.city = to!string(row[11]);
    e.province = to!string(row[12]);
    e.postcode = to!string(row[13]);
    return e;
}

Employee getEmployee(int id)
{
    string sql = "select * from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Employee e;
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return e;
    return prepareEmployee(rows[0]);
}

ulong editEmployee(Employee e)
```

```
{  
    string sql = "update employees set empid=?,  
        dept=? , paygd=? , email=? , pword=? ,  
        fname=? , lname=? , phone=? , photo=? ,  
        street=? , city=? , province=? , postcode=?  
        where id=?";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs  
(  
        to!string(e.empid),  
        to!string(e.deprt),  
        to!string(e.paygd),  
        to!string(e.email),  
        to!string(e.pword),  
        to!string(e.fname),  
        to!string(e.lname),  
        to!string(e.phone),  
        to!string(e.photo),  
        to!string(e.street),  
        to!string(e.city),  
        to!string(e.province),  
        to!string(e.postcode),  
        to!int(to!string(e.id))  
    );  
    return conn.exec(pstmt);  
}  
  
void deleteEmployee(int id)  
{  
    string sql = "delete from employees where id=?";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs(id);
```

```
    conn.exec(pstmt);
}

Employee findEmployee(string first, string last)
{
    Employee e;
    string sql = "select * from employees where upper(fname)=? and upper(lname)=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(first, last);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return e;
    return prepareEmployee(rows[0]);
}

bool isAdmin(string email, string password)
{
    string sql = "select * from admins where email=? and pword=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(email, password);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return false;
    return true;
}
```

Next is how to let the other pages know about the logged-in state.

## Enforcing authorization through the session variable m\_user

We want to restrict access to all the pages in the app except the home page and the login page, but right now, clicking on any of the links on the menu shows the page. Meaning, all the links in the menu are available and viewable by anyone. We need to rectify that.

Only the home page and the login page should be accessible to anyone and the rest should be accessible only to logged-in users. We should add a way to check if the user is logged in before deciding to open a page.

Edit source\empcontrol.d and add this at the end.

```
private enum auth = before!ensureAuth("_authUser");

private string ensureAuth(HTTPRequest req, HTTPServerResponse res)
{
    if(!m_user.loggedIn) redirect("index");
    return m_user.email;
}
mixin PrivateAccessProxy;
```

The @auth annotation is a shortcut for calling the ensureAuth() method to check if the user is logged in before running a method.

That mixin statement there is needed to make this private function accessible.

This function redirects to the index() method if the user is not logged in yet.

Then we defined a shortcut to the ensureAuth() function with this:

```
private enum auth = before!ensureAuth("_authUser");
```

so we can just use @auth to mean we are calling the ensureAuth() private function, like this:

```
@auth
void getAddEmployee(string _error = null)
```

The `@auth` annotation means call the `ensureAuth()` function, which checks the logged-in state, before running the `getAddEmployee()` method.

Since the `ensureAuth()` function is passing the generated `_authUser` variable, we now have to add it as an argument to all the methods that call `ensureAuth()`, like this:

```
@auth
void getAddEmployee(string _authUser, string _error = null)
```

So here is source\app.d.

```
import vibe.vibe;
import empcontrol;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];
    settings.sessionStore = new MemorySessionStore;

    auto router = new URLRouter;
    router.get("*", serveStaticFiles("public/"));
    router.registerWebInterface(new EmployeeController);

    auto listener = listenHTTP(settings, router);
    scope (exit) listener.stopListening();

    runApplication();
}
```

And here is the full source\empcontrol.d file at this point.

```
module empcontrol;

import vibe.vibe;
import empmodel;

struct User
{
    bool loggedIn;
    string email;
}

class EmployeeController
{
    private EmployeeModel empModel;
    private string realm = "The Lorem Ipsum Company";
    private SessionVar!(User, "user") m_user;

    this()
    {
        empModel = new EmployeeModel;
    }

    void index(string _error = null)
    {
        string error = _error;
        render!("index.dt", error);
    }

    @auth
```

```
void getAddEmployee(string _authUser, string _error = null)
{
    string error = _error;
    render!("empadd.dt", departments, paygrades, provinces, error);
}

@errorDisplay!getAddEmployee
void postAddEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string photopath = "none yet";
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
            try moveFile(pic.tempPath, NativePath(fullpath));
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
        }
        e.photo = photopath;
    }
    if(e.phone.length == 0) e.phone = "(123) 456 7890";
}
```

```
    if(e.paygd.length == 0) e.paygd = "none yet";
    if(e.postcode.length == 0) e.postcode = "A1A 1A1";
    e.pword = createDigestPassword(realm, e.email, e.pword);
    empModel.addEmployee(e);
    redirect("all_employees");
}

@auth
void getAllEmployees(string _authUser, string _error = null)
{
    string error = _error;
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps, error);
}

@auth
void getEditEmployee(string _authUser, int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    render!("empedit.dt", e, departments, paygrades, provinces, error);
}

@errorDisplay!getEditEmployee
void postEditEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    string photopath = e.photo;
```

```
auto pic = "picture" in request.files;
if(pic !is null)
{
    string ext = extension(pic.filename.name);
    string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
    if(canFind(exts, ext))
    {
        photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
        string dir = "./public/uploads/photos/";
        mkdirRecurse(dir);
        string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
        try moveFile(pic.tempPath, NativePath(fullpath));
        catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
    }
}
e.photo = photopath;
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
e.pword = createDigestPassword(realms, e.email, e.pword);
empModel.editEmployee(e);
redirect("all_employees");
}

@auth
void getDeleteEmployee(string _authUser, int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    render!("empdelete.dt", e, error);
}
```

```
@errorDisplay!getDeleteEmployee
void postDeleteEmployee(int id)
{
    empModel.deleteEmployee(id);
    redirect("all_employees");
}

@auth
@errorDisplay!getAllEmployees
void postFindEmployee(string _authUser, string fname, string lname)
{
    import std.uni; //so we can use the toUpper() function

    string first = toUpper(fname);
    string last = toUpper(lname);
    Employee e = empModel.findEmployee(first, last);
    enforce(e != Employee.init, fname ~ " " ~ lname ~ " not found.");
    render!("employee.dt", e);
}

@errorDisplay!index
void postLogin(string email, string password)
{
    import vibe.http.auth.digest_auth;

    auto scrambled = createDigestPassword(realM, email, password);
    bool isAdmin = empModel.isAdmin(email, scrambled);
    enforce(isAdmin, "Email and password combination not found.");
    User user = m_user;
    user.loggedIn = true;
    user.email = email;
    m_user = user;
}
```

```
    redirect("all_employees");
}

private enum auth = before!ensureAuth("_authUser");

private string ensureAuth(HTTPServerRequest req, HTTPServerResponse res)
{
    if(!m_user.loggedIn) redirect("/");
    return m_user.email;
}
mixin PrivateAccessProxy;
}
```

The line

```
    enforce(isAdmin, "Email and password combination not found.");
```

means if isAdmin is true, continue with the processing. If not, abort and return the provided error message.

We added the @auth annotation for each method that requires authorization.

We did not make any changes to the source\empmodel.d, so we are good.

Compile, run and refresh the browser. Click any link on the menu except the Login link and you should be redirected to the home (index) page.

However, once you logged in, you will be able to visit all the pages.

Now we are assured that sensitive data is protected and accessible only to authorized users.

How about logging out?

## Logging out

Let's just decide that if the user clicks on the Home link on the menu, the user is logged out.

Let's edit the index() method then.

```
void index(string _error = null)
{
    m_user = User.init;
    terminateSession;
    string error = _error;
    render!("index.dt", error);
}
```

Here, we re-initialized the m\_user session variable then explicitly terminated the session before displaying the index.dt template page.

Compile, run, refresh the browser and test the app. You will see that when you click on the Home link, you have to log in again to see the other pages.

This effectively completes the logging in, authentication, authorization and logging out parts.

Before moving on to the timekeeping system, let us view all the sources again.

## All the sources so far

Here is source\app.d:

```
import vibe.vibe;
import empcontrol;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];
    settings.sessionStore = new MemorySessionStore;

    auto router = new URLRouter;
    router.get("*", serveStaticFiles("public/"));
    router.registerWebInterface(new EmployeeController);

    auto listener = listenHTTP(settings, router);
    scope (exit) listener.stopListening();

    runApplication();
}
```

Here is source\empcontrol.d:

```
module empcontrol;

import vibe.vibe;
import empmodel;

struct User
```

```
{  
    bool loggedIn;  
    string email;  
}  
  
class EmployeeController  
{  
    private EmployeeModel empModel;  
    private string realm = "The Lorem Ipsum Company";  
    private SessionVar!(User, "user") m_user;  
  
    this()  
    {  
        empModel = new EmployeeModel;  
    }  
  
    void index(string _error = null)  
    {  
        string error = _error;  
        m_user = User.init;  
        terminateSession;  
        render!("index.dt", error);  
    }  
  
    @auth  
    void getAddEmployee(string _authUser, string _error = null)  
    {  
        string error = _error;  
        render!("empadd.dt", departments, paygrades, provinces, error);  
    }  
  
    @errorDisplay!getAddEmployee
```

```
void postAddEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string photopath = "none yet";
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
            try moveFile(pic.tempPath, NativePath(fullpath));
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
        }
        e.photo = photopath;
    }
    if(e.phone.length == 0) e.phone = "(123) 456 7890";
    if(e.paygd.length == 0) e.paygd = "none yet";
    if(e.postcode.length == 0) e.postcode = "A1A 1A1";
    e.pword = createDigestPassword(realm, e.email, e.pword);
    empModel.addEmployee(e);
    redirect("all_employees");
}
```

```
@auth
void getAllEmployees(string _authUser, string _error = null)
{
    string error = _error;
    Employee[] emps = empModel.getEmployees();
    render!("emplistall.dt", emps, error);
}

@auth
void getEditEmployee(string _authUser, int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    render!("empedit.dt", e, departments, paygrades, provinces, error);
}

@errorDisplay!getEditEmployee
void postEditEmployee(Employee e)
{
    import std.file;
    import std.path;
    import std.algorithm;
    import vibe.http.auth.digest_auth;

    string photopath = e.photo;
    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
```

```

photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
string dir = "./public/uploads/photos/";
mkdirRecurse(dir);
string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
try moveFile(pic.tempPath, NativePath(fullpath));
catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
}
}

e.photo = photopath;
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.paygd.length == 0) e.paygd = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
e.pword = createDigestPassword(realm, e.email, e.pword);
empModel.editEmployee(e);
redirect("all_employees");
}

@auth
void getDeleteEmployee(string _authUser, int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    render!("empdelete.dt", e, error);
}

@errorDisplay!getDeleteEmployee
void postDeleteEmployee(int id)
{
    empModel.deleteEmployee(id);
    redirect("all_employees");
}

```

```
@auth
@errorDisplay!getAllEmployees
void postFindEmployee(string _authUser, string fname, string lname)
{
    import std.uni; //so we can use the toUpper() function

    string first = toUpper(fname);
    string last = toUpper(lname);
    Employee e = empModel.findEmployee(first, last);
    enforce(e != Employee.init, fname ~ " " ~ lname ~ " not found.");
    render!("employee.dt", e);
}

@errorDisplay!index
void postLogin(string email, string password)
{
    import vibe.http.auth.digest_auth;

    auto scrambled = createDigestPassword(realM, email, password);
    bool isAdmin = empModel.isAdmin(email, scrambled);
    enforce(isAdmin, "Email and password combination not found.");
    User user = m_user;
    user.loggedIn = true;
    user.email = email;
    m_user = user;
    redirect("all_employees");
}

private enum auth = before!ensureAuth("_authUser");

private string ensureAuth(HTTPServerRequest req, HTTPServerResponse res)
{
```

```
    if(!m_user.loggedIn) redirect("/");
    return m_user.email;
}
mixin PrivateAccessProxy;
}
```

And this is source\empmodel.d:

```
module empmodel;

import mysql;
import std.conv;
import std.array;

struct Employee
{
    int id; //row id or record id
    string empid; //employee number
    string dept; //department
    string paygd; //salary grade
    string email; //email address
    string pword; //password
    string fname; //first name
    string lname; //last name
    string phone; //phone number
    string photo; //ID photo
    string street; //street address
    string city; //city name
    string province; //province name
    string postcode; //postal code
}
```

```
struct Admin
{
    string email; //email address
    string pword; //password
}

string[] departments =
[
    "Management and Admin",
    "Accounting and Finance",
    "Production",
    "Maintenance",
    "Shipping and Receiving",
    "Purchasing and Supplies",
    "IT Services",
    "Human Resources",
    "Marketing"
];

string[][] provinces =
[
    ["AB", "Alberta"],
    ["BC", "British Columbia"],
    ["MB", "Manitoba"],
    ["NB", "New Brunswick"],
    ["NL", "Newfoundland and Labrador"],
    ["NS", "Nova Scotia"],
    ["NT", "Northwest Territories"],
    ["NU", "Nunavut"],
    ["ON", "Ontario"],
    ["PE", "Prince Edward Island"],
    ["QC", "Quebec"],
```

```
    ["SK", "Saskatchewan"],
    ["YT", "Yukon Territory"]
];

string[] paygrades =
[
    "A100", "A200", "A300", "A400",
    "B100", "B200", "B300", "B400",
    "C100", "C200", "C300", "C400",
    "D100", "D200", "D300", "D400",
    "E100", "E200", "E300", "E400",
    "F100", "F200", "F300", "F400"
];

class EmployeeModel
{
    Connection conn;

    this()
    {
        string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";
        conn = new Connection(url);
        scope(exit) conn.close;
    }

    ulong addEmployee(Employee e)
    {
        string sql =
            "insert into employees
            (
                empid,
                dept, paygd, email, pword,
```

```
        fname, lname, phone, photo,
        street, city, province, postcode
    )
    values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setArgs
(
    toString(e.empid),
    toString(e.deprt),
    toString(e.paygd),
    toString(e.email),
    toString(e.pword),
    toString(e.fname),
    toString(e.lname),
    toString(e.phone),
    toString(e.photo),
    toString(e.street),
    toString(e.city),
    toString(e.province),
    toString(e.postcode)
);
return conn.exec(pstmt);
}

Employee[] getEmployees()
{
    Employee[] emps;
    String sql = "select * from employees";
    Row[] rows = conn.query(sql).array;
    if(rows.length == 0) return emps;
    return prepareEmployees(rows);
}
```

```
Employee[] prepareEmployees(Row[] rows)
{
    Employee[] emps;
    foreach(row; rows)
    {
        Employee e = prepareEmployee(row);
        emps ~= e;
    }
    return emps;
}

Employee prepareEmployee(Row row)
{
    Employee e;
    e.id = to!int(to!string(row[0]));
    e.empid = to!string(row[1]);
    e.deprt = to!string(row[2]);
    e.paygd = to!string(row[3]);
    e.email = to!string(row[4]);
    e.pword = to!string(row[5]);
    e.fname = to!string(row[6]);
    e.lname = to!string(row[7]);
    e.phone = to!string(row[8]);
    e.photo = to!string(row[9]);
    e.street = to!string(row[10]);
    e.city = to!string(row[11]);
    e.province = to!string(row[12]);
    e.postcode = to!string(row[13]);
    return e;
}
```

```
Employee getEmployee(int id)
{
    string sql = "select * from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Employee e;
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return e;
    return prepareEmployee(rows[0]);
}

ulong editEmployee(Employee e)
{
    string sql = "update employees set empid=?,
        dept=? , paygd=? , email=? , pword=? ,
        fname=? , lname=? , phone=? , photo=? ,
        street=? , city=? , province=? , postcode=?
        where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs
    (
        to!string(e.empid),
        to!string(e.deprt),
        to!string(e.paygd),
        to!string(e.email),
        to!string(e.pword),
        to!string(e.fname),
        to!string(e.lname),
        to!string(e.phone),
        to!string(e.photo),
        to!string(e.street),
        to!string(e.city),
    )
}
```

```
        to!string(e.province),
        to!string(e.postcode),
        to!int(to!string(e.id))
    );
    return conn.exec(pstmt);
}

void deleteEmployee(int id)
{
    string sql = "delete from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    conn.exec(pstmt);
}

Employee findEmployee(string first, string last)
{
    Employee e;
    string sql = "select * from employees where upper(fname)=? and upper(lname)=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(first, last);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return e;
    return prepareEmployee(rows[0]);
}

bool isAdmin(string email, string password)
{
    string sql = "select * from admins where email=? and pword=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(email, password);
    Row[] rows = conn.query(pstmt).array;
```

```
        if(rows.length == 0) return false;
        return true;
    }
}
```

Here is views\cssformgrid.dt:

```
:css
.form-grid-wrapper
{
    width: 700px;
    height: auto;
    margin: 20px auto;
    padding: 1px 20px 20px 0;
    border-radius: 20px;
    background-color: #eef;
}
.form-grid
{
    display: grid;
    grid-template-columns: 1fr 3fr;
    gap: 5px;
}
.center-align
{
    text-align: center;
}
.form-grid-label
{
    width: 100%;
    font-size: 16px;
    text-align: right;
```

```
    padding: 2px;
}
.form-grid-input
{
    width: 100%;
    font-size: 14px;
    padding: 0;
}
.form-grid-field
{
    width: 100%;
    font-size: 16px;
    border-bottom: 1px solid black;
    padding: 2px;
}
.form-grid-button
{
    width: 100%;
    font-size: 14px;
    height: 30px;
    margin-top: 10px;
}
```

Here is views\csstable.dt:

```
:css
.table-wrapper
{
    margin: 20px auto;
    border-radius: 20px;
}
```

```
table
{
    margin: 0 auto;
    padding: 20px 0;
    border-collapse: collapse;
    background-color: #eef;
}
table td, table th
{
    border: 1px solid black;
    margin: 0;
    padding: 0 5px;
}
.no-border
{
    border: 0;
}
```

Here is views\empadd.dt:

```
extends layout
block maincontent
    include cssformgrid.dt
    -if(error)
        div.error-div
            span.error-message #{error}
    div.form-grid-wrapper
        h2.center-align New employee details
        form.form-grid(method="post", action="add_employee", enctype="multipart/form-data")
            label.form-grid-label Employee number
            input.form-grid-input(type="empid", name="e_empid", placeholder="employee number", required)
            label.form-grid-label Department
```

```
select#dept.form-grid-input(name="e_dept")
    -foreach(dep; departments)
        option(value="#{dep}") #{dep}
label.form-grid-label Salary grade
select#dept.form-grid-input(name="e_paygd")
    -foreach(pay; paygrades)
        option(value="#{pay}") #{pay}
label.form-grid-label Email address
input.form-grid-input(type="email", name="e_email", placeholder="email@company.com", required)
label.form-grid-label Password
input.form-grid-input(type="password", name="e_pword", placeholder="password", required)
label.form-grid-label First name
input.form-grid-input(type="text", name="e_fname", placeholder="First name", required)
label.form-grid-label Last name
input.form-grid-input(type="text", name="e_lname", placeholder="Last name", required)
label.form-grid-label Phone
input.form-grid-input(type="text", name="e_phone", placeholder="Phone number")
label.form-grid-label Street address (no city)
input.form-grid-input(type="text", name="e_street", placeholder="Street address", required)
label.form-grid-label City
input.form-grid-input(type="text", name="e_city", placeholder="City", required)
label.form-grid-label Province
select#province.form-grid-input(name="e_province")
    -foreach(prov; provinces)
        option(value="#{prov[0]}") #{prov[1]}
label.form-grid-label Postal code
input.form-grid-input(type="text", name="e_postcode", placeholder="A1A 1A1")
label.form-grid-label ID Picture
input.form-grid-input(type="file", name="picture")
input(type="hidden", name="e_photo")
input(type="hidden", name="e_id", value="1")
div
```

```
div
    input.form-grid-button(type="reset", value="Clear the form")
    input.form-grid-button(type="submit", value="Submit")
```

Here is views\empdelete.dt:

```
extends layout
block maincontent
    include cssformgrid.dt
    -if(error)
        div.error-div
            span.error-message #{error}
    div.form-grid-wrapper
        h2.center-align Are you sure you want to delete this record?
        form.form-grid(method="post", action="delete_employee")
            span.form-grid-label Employee number:
            span.form-grid-field #{e.empid}
            span.form-grid-label Department:
            span.form-grid-field #{e.deprt}
            span.form-grid-label Salary grade:
            span.form-grid-field #{e.paygd}
            span.form-grid-label Email address:
            span.form-grid-field #{e.email}
            span.form-grid-label First name:
            span.form-grid-field #{e.fname}
            span.form-grid-label Last name:
            span.form-grid-field #{e.lname}
            span.form-grid-label Phone:
            span.form-grid-field #{e.phone}
            span.form-grid-label Street address:
            span.form-grid-field #{e.street}
            span.form-grid-label City:
```

```

.form-grid-field #{e.city}
.form-grid-label Province:
.form-grid-field #{e.province}
.form-grid-label Postal code:
.form-grid-field #{e.postcode}
.form-grid-label ID Picture:


```

Here is views\empedit.dt:

```

extends layout
block maincontent
  include cssformgrid.dt
  -if(error)
    div.error-div
      span.error-message #{error}
  div.form-grid-wrapper
    h2.center-align Edit employee details
    form.form-grid(method="post", action="edit_employee", enctype="multipart/form-data")
      label.form-grid-label Employee number
      input.form-grid-input(type="empid", name="e_empid", value="#{e.empid}")
      label.form-grid-label Department
      select#dept.form-grid-input(name="e_dept", value="#{e.deprt}")
        -foreach(dep; departments)
          -if(dep == e.deprt)
            option(value="#{dep}", selected) #{dep}

```

```
-else
    option(value="#{dep}") #{dep}
label.form-grid-label Salary grade
select#paygd.form-grid-input(name="e_paygd", value="#{e.paygd}")
-foreach(pay; paygrades)
    -if(pay == e.paygd)
        option(value="#{pay}", selected) #{pay}
    -else
        option(value="#{pay}") #{pay}
label.form-grid-label Email address
input.form-grid-input(type="email", name="e_email", value="#{e.email}")
label.form-grid-label Password
input.form-grid-input(type="password", name="e_pword", value="#{e.pword}")
label.form-grid-label First name
input.form-grid-input(type="text", name="e_fname", value="#{e.fname}")
label.form-grid-label Last name
input.form-grid-input(type="text", name="e_lname", value="#{e.lname}")
label.form-grid-label Phone
input.form-grid-input(type="text", name="e_phone", value="#{e.phone}")
label.form-grid-label Street address (no city)
input.form-grid-input(type="text", name="e_street", value="#{e.street}")
label.form-grid-label City
input.form-grid-input(type="text", name="e_city", value="#{e.city}")
label.form-grid-label Province
select#province.form-grid-input(name="e_province", value="#{e.province}")
-foreach(prov; provinces)
    -if(prov[0] == e.province)
        option(value="#{prov[0]}", selected) #{prov[1]}
    -else
        option(value="#{prov[0]}") #{prov[1]}
label.form-grid-label Postal code
input.form-grid-input(type="text", name="e_postcode", value="#{e.postcode}")
```

```
label.form-grid-label ID Picture
  img(src="#{e.photo}", height="100px")
  div
    input.form-grid-input(type="file", name="picture")
    input(type="hidden", name="e_photo", value="#{e.photo}")
    input(type="hidden", name="e_id", value="#{e.id}")
    input(type="hidden", name="id", value="#{e.id}")
  div
  div
    a(href="all_employees")
      button.form-grid-button(type="button") Cancel
    input.form-grid-button(type="submit", value="Submit")
```

Here is views\emplistall.dt:

```
extends layout
block maincontent
  include csstable.dt
  -if(error)
    div.error-div
      span.error-message #{error}
  div.table-wrapper
    table
      tr
        th Employee Id
        th First name
        th Last name
        th Department
        th Phone number
        th Email address
        th Street address
        th City
```

```

th Province
th PostCode
th Action
-foreach(e; emps)
tr
td #{e.empid}
td #{e.fname}
td #{e.lname}
td #{e.deprt}
td #{e.phone}
td #{e.email}
td #{e.street}
td #{e.city}
td #{e.province}
td #{e.postcode}
td &nbsp;
form.form-hidden(method="get", action="edit_employee")
input(type="hidden", name="id", value="#{e.id}")
input(type="image", src="images/pencil.ico", height="15px")
| &nbsp;
form.form-hidden(method="get", action="delete_employee")
input(type="hidden", name="id", value="#{e.id}")
input(type="image", src="images/trash.ico", height="15px")
| &nbsp;

```

Here is views\employee.dt:

```

extends layout
block maincontent
  include cssformgrid.dt
  div.form-grid-wrapper
    h2.center-align Employee details

```

```
div.form-grid
  span.form-grid-label Employee number:
  span.form-grid-field #{e.empid}
  span.form-grid-label Department:
  span.form-grid-field #{e.deprt}
  span.form-grid-label Salary grade:
  span.form-grid-field #{e.paygd}
  span.form-grid-label Email address:
  span.form-grid-field #{e.email}
  span.form-grid-label First name:
  span.form-grid-field #{e.fname}
  span.form-grid-label Last name:
  span.form-grid-field #{e.lname}
  span.form-grid-label Phone:
  span.form-grid-field #{e.phone}
  span.form-grid-label Street address:
  span.form-grid-field #{e.street}
  span.form-grid-label City:
  span.form-grid-field #{e.city}
  span.form-grid-label Province:
  span.form-grid-field #{e.province}
  span.form-grid-label Postal code:
  span.form-grid-field #{e.postcode}
  span.form-grid-label ID Picture:
  img(src="#{e.photo}", height="80px")
div
div
  a(href="all_employees")
  button.form-grid-button(type="button") Close
```

Here is views\index.dt:

```
extends layout
block maincontent
    h2 The Lorem Ipsum Company
    -if(error)
        div.error-div
            span.error-message #{error}
    div.
        Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Nam nec urna arcu. Quisque eleifend posuere vestibulum.
        In sed magna mauris. Phasellus bibendum ligula et placerat
        vulputate. In non suscipit lectus, a laoreet odio. Donec
        at sapien eu nisi porta condimentum. Morbi non varius ex,
        nec luctus nisl. Aenean varius dui quis arcu auctor luctus.
        Integer efficitur ornare massa, ac suscipit enim sagittis et.
        Proin vestibulum tellus in ipsum ultrices, sed imperdiet
        sapien euismod. Praesent vel facilisis mauris. Proin finibus
        congue tellus, non varius ante. Nullam tincidunt dolor felis.
        Pellentesque non luctus tellus. Curabitur et sapien at justo
        fringilla feugiat et a erat.
        <br /><br />
```

Here is views\layout.dt:

```
doctype 5
html
    head
        title Employee Timekeeping System
        include styles.dt
    body
        div.container
            div.menu
                include menu.dt
```

```
div.content
  block maincontent
div.footer
  include footer.dt
```

Here is views\menu.dt:

```
div.menu-item
  a.item-link(href="/") Home
div.menu-item
  a.item-link(href="all_employees") All employees
div.menu-item
  a.item-link(href="#find_employee") Find employee
div.menu-item
  a.item-link(href="add_employee") Add employee
div.menu-item
  a.item-link.item-link-right(href="#login") Login
div#find_employee.modal-form
  div.modal-form-wrapper-find_employee
    form.modal-form-grid(method="post", action="find_employee")
      input.text-control(name="fname", type="text", placeholder=" First name", required)
      input.text-control(name="lname", type="text", placeholder=" Last name", required)
      input#but-reset(type="reset", value="Clear")
      input#but-submit(type="submit", value="Find")
    a.close(href="#close") Cancel
div#login.modal-form
  div.modal-form-wrapper-login
    form.modal-form-grid(method="post", action="login")
      input.text-control(name="email", type="email", placeholder=" email address")
      input.text-control(name="password", type="password", placeholder=" password")
      input#but-reset(type="reset", value="Clear")
      input#but-submit(type="submit", value="Login")
```

```
a.close(href="#close") Cancel
```

Here is views\footer.dt:

```
div.copyright Copyright &copy; The Lorem Ipsum Company 2023
```

Here is views\styles.dt:

```
:css
body
{
    margin: 0;
    padding: 0;
}
.container
{
    display: grid;
    grid-template-rows: 40px auto 30px;
    height: 100%;
    width: 100%;
    margin: 0;
    padding: 0;
}
/*menu styles*****
.menu
{
    position: fixed;
    top: 0;
    width: 100%;
    padding: 10px 0;
    background-color: whitesmoke;
```

```
}

.menu-item
{
    display: inline;
}
.item-link
{
    margin-left: 30px;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: 18px;
    color: teal;
    text-decoration: none;
}
.item-link-right
{
    float: right;
    margin-right: 30px;
}
/*end of menu styles*/
/*modal form styles***** */
.modal-form
{
    position: fixed;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    background: rgba(0,0,0,0.5);
    z-index: 99999;
    opacity: 0;
    pointer-events: none;
```

```
}

#login:target, #find_employee:target
{
    opacity: 1;
    pointer-events: auto;
}

.modal-form-grid
{
    display: grid;
    grid-template: 30px 30px 30px / 1fr 1fr;
    grid-gap: 10px;
}

.text-control
{
    grid-column: 1 / 3;
}

#but-reset
{
    grid-column: 1 / 2;
}

#but-submit
{
    grid-column: 2 / 3;
}

.modal-form-wrapper-login
{
    width: 250px;
    position: absolute;
    right: 0;
    margin-top: 40px;
    padding: 25px 20px;
    border-radius: 10px;
```

```
    text-align: center;
    background-color: whitesmoke;
}
.modal-form-wrapper-find_employee
{
    width: 250px;
    position: relative;
    left: 280px;
    margin-top: 40px;
    padding: 25px 20px;
    border-radius: 10px;
    text-align: center;
    background-color: whitesmoke;
}
/*end of modal form styles*/
/*content styles******/
.content
{
    padding: 40px 30px;
}
/*end of content styles*/
/*footer styles******/
.footer
{
    position: fixed;
    bottom: 0;
    width: 100%;
    background-color: lightgray;
    padding: 5px 0;
}
.copyright
{
```

```
font-family: Verdana, Geneva, Tahoma, sans-serif;
font-size: 14px;
text-align: center;
color: teal;
}
.form-hidden
{
  padding: 0;
  margin: 0;
  display: inline;
}
.error-div
{
  width: 90%;
  margin: 20px auto;
}
.error-message
{
  color: brown;
  font-weight: bold;
}
```

And here is dub.json

```
{
  "authors": [
    "Owner"
  ],
  "copyright": "Copyright © 2023, Owner",
  "dependencies": {
    "mysql-native": "~>3.2.0",
    "vibe-d": "~>0.9"
```

```
},
"description": "A simple vibe.d server application.",
"license": "proprietary",
"name": "lorem"
}
```

Now on to the timekeeping part.

## A new project

As in the previous project, we are going to assume this project is going to be deployed to an intranet setting so we don't have to worry about security and huge numbers of users accessing data at the same time. Most security concerns are beyond the control of a web app itself anyway, like setting up a proxy server with nGinx or Apache, hiding behind a firewall, setting up a CA certificate like the free Let's Encrypt for HTTPS encryption, hardening MySQL, etc.

Create a new project named loremtime.

```
C:\Users\Owner>cd \vibeprojects
```

```
C:\vibeprojects>dub init loremtime -t vibe.d
Package recipe format (sdl/json) [json]:
Name [loremtime]:
Description [A simple vibe.d server application.]:
Author name [Owner]:
License [proprietary]:
Copyright string [Copyright T- 2023, Owner]:
Add dependency (leave empty to skip) []:
Success created empty project in C:\vibeprojects\loremtime
Package successfully created in loremtime
```

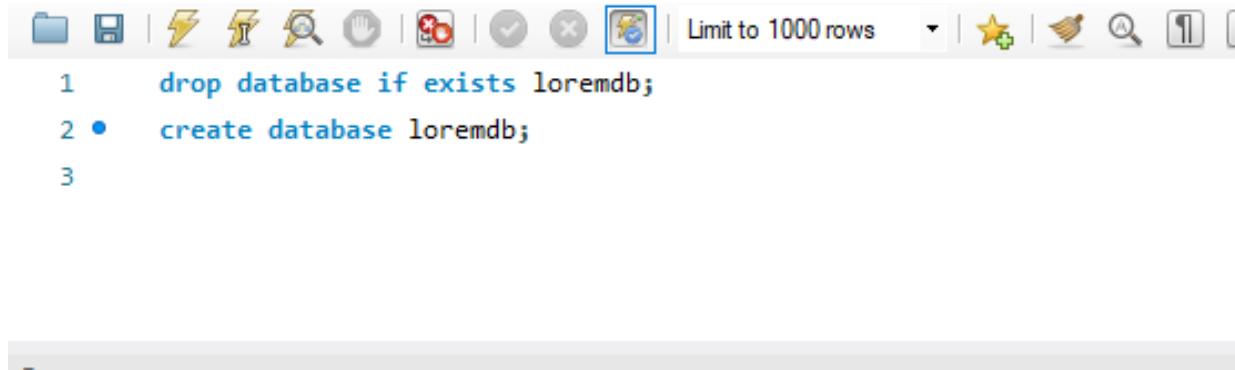
```
C:\vibeprojects>cd loremtime
```

And add the mysql-native library to the project.

```
C:\vibeprojects\loremtime>dub add mysql-native
Adding dependency mysql-native ~>3.2.2
```

```
C:\vibeprojects\loremtime>
```

Then, using MySQL Workbench, create a new database named loremdb.



A screenshot of the MySQL Workbench interface. The top bar includes standard icons for file operations, search, and help, along with a "Limit to 1000 rows" dropdown and a toolbar with various tools. Below the toolbar, the main area shows a SQL query editor with the following code:

```
1 drop database if exists loremdb;
2 • create database loremdb;
3
```

The second line, "create database loremdb;", has a blue dot next to it, indicating it is the currently selected or active statement. The code is displayed in a monospaced font.

Now we are ready to build the timekeeping system for the Lorem Ipsum company.

## The schema

The timekeeping system needs to record the time-in and time-out actions of the employees, like a timecard, so we need a timecard model.

Then, after every salary period, in our case weekly, a timesheet is produced summarizing the number of hours employees worked for the week, so we need a timesheet model.

Here is the structure for the timecard:

Id – the record id

Empid – the employee id

Fullscreen – the employee full name

Timein – the time the employee punched in for the day

Timeout – the time the employee punched out

Hours – the number of hours between time in and time out, which computed after the punch out

Here is the structure for the timesheet:

Id – the record id

Period – the salary period (or week)

Empid – the employee id

Fullscreen – the employee full name

Hours – the accrued hours for the week

The reason the empid and fullname are included is to eliminate the need for the timesheet to look up the employee number and names of employees from the employees table.

This time we will create the administrators table. So we need an admins model as well, like this:

Id – the record id

Email – the admin email address

Password – the admin password

We already have the employees table, so that's covered.

The timesheet will simply extract and summarize data from the timecards. The timecards will become permanent records while the timesheet will be overwritten every time it is generated. The timecards can be edited but the timesheet does not need to be edited as it depends on the timecards.

## The base model

All of the models will be connecting to the same database with the same URL for the connections. To reduce code duplication, we will create a base model.

Create the base model in source\basemodel.d:

```
module basemodel;

import mysql;

class BaseModel
{
    string realm = "The Lorem Ipsum Company";
    Connection conn;

    this()
    {
        string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=loremdb";
        conn = new Connection(url);
        scope(exit) conn.close;
    }
}
```

Our base model connects to the database at its creation, which will be inherited by all of the models. Of course, you replace the username and password with your own.

## The employee model

Edit source\empmodel.d with this contents. The highlighted lines are only for emphasis; there are small changes here and there so better copy the whole thing:

```
module empmodel;

import mysql;
import std.conv;
import std.array;
import basemodel;

struct Employee
{
    int id; //row id or record id
    string empid; //employee number
    string email; //email address
    string passw; //password
    string fname; //first name
    string lname; //last name
    string phone; //phone number
    string photo; //ID photo
    string dept; //department
    string payrate; //salary grade
    string street; //street address
    string city; //city name
    string province; //province name
    string postcode; //postal code
}

string[] departments =
[
```

```
"Management",
"Accounting",
"Production",
"Maintenance",
"Shipping",
"Purchasing",
"IT Services",
"HR Services",
"Marketing"
];
string[][] provinces =
[
    ["AB", "Alberta"],
    ["BC", "British Columbia"],
    ["MB", "Manitoba"],
    ["NB", "New Brunswick"],
    ["NL", "Newfoundland and Labrador"],
    ["NS", "Nova Scotia"],
    ["NT", "Northwest Territories"],
    ["NU", "Nunavut"],
    ["ON", "Ontario"],
    ["PE", "Prince Edward Island"],
    ["QC", "Quebec"],
    ["SK", "Saskatchewan"],
    ["YT", "Yukon Territory"]
];
class EmployeeModel : BaseModel
{
    void createTable()
    {
```

```
string sql = "drop table if exists employees";
conn.exec(sql);
sql =
"create table employees
(
    id int auto_increment primary key,
    empid char(4) not null unique,
    email varchar(50) not null,
    passw varchar(255) not null,
    fname varchar(25) not null,
    lname varchar(25) not null,
    phone varchar(15) default '123-456-7890',
    photo varchar(50) default 'none provided',
    dept varchar(30) default 'not assigned yet',
    payrate char(4) default 'B400',
    street varchar(50) default '123 First Street',
    city varchar(30) default 'Toronto',
    province char(2) default 'ON',
    postcode char(7) default 'Z9Z 9Z9',
    created timestamp default current_timestamp,
    updated timestamp on update current_timestamp
)";
conn.exec(sql);
}

void createPayrates()
{
    int[string] payrates =
    [
        "A100":100, "A200":75, "A300":50, "A400":40,
        "B100":30, "B200":29, "B300":28, "B400":27,
        "C100":26, "C200":25, "C300":24, "C400":23,
```

```
"D100":22, "D200":21, "D300":20, "D400":19,
"E100":18, "E200":17, "E300":16, "E400":15,
"F100":14, "F200":13, "F300":12, "F400":11
];
string sql = "drop table if exists payrates";
conn.exec(sql);
sql =
"create table payrates
(
    id int auto_increment primary key,
    payrate char(4) not null,
    hourly int not null
)";
conn.exec(sql);
foreach(k,v; payrates)
{
    sql =
        "insert into payrates(payrate, hourly)
        values('" ~ k ~ "', " ~ to!string(v) ~ ")";
    conn.exec(sql);
}
}

string[] getPayrates()
{
    string sql = "select payrate from payrates order by payrate";
    Row[] rows = conn.query(sql).array;
    string[] rates;
    foreach(row; rows) rates ~= to!string(row[0]);
    return rates;
}
```

```
void addEmployee(Employee e)
{
    import vibe.http.auth.digest_auth;
    e.passw = createDigestPassword(realm, e.email, e.passw);
    string sql =
        "insert into employees
        (
            empid,
            email, passw, fname, lname,
            phone, photo, dept, payrate,
            street, city, province, postcode
        )
        values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs
    (
        to!string(e.empid),
        to!string(e.email),
        to!string(e.passw),
        to!string(e.fname),
        to!string(e.lname),
        to!string(e.phone),
        to!string(e.photo),
        to!string(e.deprt),
        to!string(e.payrate),
        to!string(e.street),
        to!string(e.city),
        to!string(e.province),
        to!string(e.postcode)
    );
    conn.exec(pstmt);
}
```

```
Employee[] getEmployees()
{
    Employee[] emps;
    String sql = "select * from employees";
    Row[] rows = conn.query(sql).array;
    if(rows.length == 0) return emps;
    foreach(row; rows) emps ~= prepareEmployee(row);
    return emps;
}

Employee prepareEmployee(Row row)
{
    Employee e;
    e.id = to!int(to!string(row[0]));
    e.empid = to!string(row[1]);
    e.email = to!string(row[2]);
    e.passw = to!string(row[3]);
    e.fname = to!string(row[4]);
    e.lname = to!string(row[5]);
    e.phone = to!string(row[6]);
    e.photo = to!string(row[7]);
    e.deprt = to!string(row[8]);
    e.payrate = to!string(row[9]);
    e.street = to!string(row[10]);
    e.city = to!string(row[11]);
    e.province = to!string(row[12]);
    e.postcode = to!string(row[13]);
    return e;
}

Employee getEmployee(int id)
```

```
{  
    string sql = "select * from employees where id=?";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs(id);  
    Employee e;  
    Row[] rows = conn.query(pstmt).array;  
    if(rows.length == 0) return e;  
    return prepareEmployee(rows[0]);  
}  
  
Employee getEmployee(string empid)  
{  
    string sql = "select * from employees where empid=?";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs(empid);  
    Employee e;  
    Row[] rows = conn.query(pstmt).array;  
    if(rows.length == 0) return e;  
    return prepareEmployee(rows[0]);  
}  
  
void editEmployee(Employee e, bool newpass)  
{  
    if(newpass) // if password was changed, encrypt it  
    {  
        import vibe.http.auth.digest_auth;  
        e.passw = createDigestPassword(realms, e.email, e.passw);  
    }  
    string sql =  
        "update employees set empid=?,  
        email=?, passw=?, fname=?, lname=?,  
        phone=?, photo=?, deprt=?, payrate=?,"
```

```
    street=?, city=?, province=?, postcode=?
    where id=?";
Prepared pstmt = conn.prepareStatement(sql);
pstmt.setArgs
(
    to!string(e.empid),
    to!string(e.email),
    to!string(e.passw),
    to!string(e.fname),
    to!string(e.lname),
    to!string(e.phone),
    to!string(e.photo),
    to!string(e.deprt),
    to!string(e.payrate),
    to!string(e.street),
    to!string(e.city),
    to!string(e.province),
    to!string(e.postcode),
    to!int(to!string(e.id))
);
conn.exec(pstmt);
}

void deleteEmployee(int id)
{
    string sql = "delete from employees where id=?";
    Prepared pstmt = conn.prepareStatement(sql);
    pstmt.setArgs(id);
    conn.exec(pstmt);
}

Employee findEmployee(string first, string last)
```

```
{  
    Employee e;  
    string sql = "select * from employees where upper(fname)=? and upper(lname)=?";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs(first, last);  
    Row[] rows = conn.query(pstmt).array;  
    if(rows.length == 0) return e;  
    return prepareEmployee(rows[0]);  
}  
}
```

We added the line

```
import basemodel;
```

near the top to make the BaseModel class visible.

The line

```
class EmployeeModel : BaseModel
```

means EmployeeModel inherits from BaseModel, which is another way of saying the EmployeeModel class is derived from the base class BaseModel, or EmployeeModel is a subclass of BaseModel.

The line

```
int[string] payrates =
```

means create an **associative array** of integers with strings as keys.

We created this associative array:

```
int[string] payrates =
```

```
[  
    "A100":100, "A200":75, "A300":50, "A400":40,  
    "B100":30, "B200":29, "B300":28, "B400":27,  
    "C100":26, "C200":25, "C300":24, "C400":23,  
    "D100":22, "D200":21, "D300":20, "D400":19,  
    "E100":18, "E200":17, "E300":16, "E400":15,  
    "F100":14, "F200":13, "F300":12, "F400":11  
];
```

So when we say

```
payrates["A300"]
```

it will give us the value of 50.

The line

```
foreach(k,v; payrates)
```

can be read as “for each key and value pair in payrates, do the following”. Of course, “k” and “v” are random variable names; you can name them “rate” and “pay” and they will have the same meaning:

```
foreach(rate,pay; payrates)
```

Through that construct, we were able to populate the payrates table with data from the payrates associative array.

## The timecard model

Create source\cardmodel.d with this contents:

```
module cardmodel;

import mysql;
import std.array;
import std.conv;
import basemodel;

struct Timecard
{
    int id;
    string empid;
    string fullname;
    string timein;
    string timeout;
    float hours;
}

class TimecardModel : BaseModel
{
    void createTable()
    {
        string sql = "drop table if exists timecards";
        conn.exec(sql);
        sql =
            "create table timecards
            (
                id int auto_increment primary key,
                empid char(4) not null references employees(empid),
```

```
    fullname varchar(50) not null,  
    timein datetime not null default current_timestamp,  
    timeout datetime,  
    hours float default 0.0  
  );  
  conn.exec(sql);  
}
```

Here we also indicated that TimecardModel is also derived from BaseModel.

## The timesheet model

Create source\sheetmodel.d:

```
module sheetmodel;

import mysql;
import std.array;
import std.conv;
import basemodel;

struct Timesheet
{
    int id;
    int period;
    string empid;
    string fullname;
    float hours;
}

class TimesheetModel : BaseModel
{
    void createTable()
    {
        string sql = "drop table if exists timesheets";
        conn.exec(sql);
        sql =
            "create table timesheets
            (
                id int auto_increment primary key,
                period int not null,
                empid char(4) not null,
            "
    }
}
```

```
    fullname varchar(50) not null,  
    hours float  
);  
conn.exec(sql);  
}  
}
```

And we also indicated here that TimesheetModel is derived from BaseModel.

## The administrators model

We are adding an administrators model so in the future we can simply provide a facility to add, edit and delete administrators.

Create source\adminmodel.d:

```
module adminmodel;

import mysql;
import std.array;
import vibe.http.auth.digest_auth;
import basemodel;

struct Admin
{
    int id;
    string email;
    string passw;
}

class AdminModel : BaseModel
{
    void createTable()
    {
        conn.exec("drop table if exists admins");
        string sql =
            "create table admins
            (
                id int auto_increment primary key,
                email varchar(30) not null unique,
                passw varchar(255) not null
            )";
    }
}
```

```
conn.exec(sql);
string[] emails =
[
    "admin1@lorem.com",
    "admin2@lorem.com",
    "admin3@lorem.com",
    "admin4@lorem.com",
    "admin5@lorem.com",
    "admin6@lorem.com"
];
foreach(email; emails)
{
    string pword = createDigestPassword(realm, email, "secret");
    sql = "insert into admins(email, passw) values ('" ~ email ~ "','" ~ pword ~ "')";
    conn.exec(sql);
}
}

bool getAdmin(string email, string pword)
{
    string password = createDigestPassword(realm, email, pword);
    string sql = "select * from admins where email=? and passw=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(email, password);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return false;
    return true;
}
```

We added the `getAdmin()` method as we already used this earlier.

## The base controller

We are going to make several controllers so that each can focus on a particular model. This will make debugging files easier as well as follow the notion of separation of concerns. Besides, having all the business logic code in one controller file will make the file big and unwieldy.

To avoid duplication of code, we are going to make a base controller class that hosts the common code and make all the other controllers inherit from it. We will call it BaseController.

Here is source\basecontrol.d:

```
module basecontrol;

import vibe.vibe;
import empmodel;
import cardmodel;
import sheetmodel;
import adminmodel;

struct User
{
    bool loggedIn;
    string email;
}

class BaseController
{
    protected SessionVar!(User, "user") m_user;
    protected enum auth = before!ensureAuth("_authUser");
    protected EmployeeModel empModel;
    protected TimecardModel cardModel;
    protected TimesheetModel sheetModel;
```

```
protected AdminModel adminModel;

this()
{
    empModel = new EmployeeModel;
    cardModel = new TimecardModel;
    sheetModel = new TimesheetModel;
    adminModel = new AdminModel;
}

string ensureAuth(HTTPServerRequest req, HTTPServerResponse res)
{
    if(!m_user.loggedIn) redirect("#login");
    return m_user.email;
}
mixin PrivateAccessProxy;

string getToday()
{
    import std.datetime.date;
    auto time = Clock.currTime;
    return to!string(Date(time.year, time.month, time.day));
}
```

There are two common accessibility attributes for variables in D: public and private.

A public variable is accessible from the class and from outside of the class and subclasses.

A private variable is accessible only to the class where it was declared; even subclasses do not have access.

In D, variables are public by default.

A protected variable is not visible to other classes except subclasses.

So, in a sense, a protected variable is “protected” from other classes while accessible from subclasses. You can say it is a cross between a public and a private variable.

We instantiated all the models here so they are ready for use by all the controllers that inherit from this class. We also included the ensureAuth() method here so they are also inherited by the other controllers.

## The employee controller

We have already used the employee controller earlier so let's review it. This time, the employee controller will inherit from the base controller.

Here is source\empcontrol.d; there are changes here and there so simply copy the whole file:

```
module empcontrol;

import vibe.vibe;
import empmodel;
import adminmodel;
import basecontrol;

class EmployeeController : BaseController
{
    @auth
    void getAddEmployee(string _authUser, string _error = null)
    {
        string error = _error;
        bool loggedIn = m_user.loggedIn;
        string[] payrates = empModel.getPayrates();
        render!("empadd.dt", departments, payrates, provinces, error, loggedIn);
    }

    @errorDisplay!getAddEmployee
    void postAddEmployee(Employee e)
    {
        import std.file;
        import std.path;
        import std.algorithm;
```

```
auto pic = "picture" in request.files;
if(pic !is null)
{
    string photopath = "none yet";
    string ext = extension(pic.filename.name);
    string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
    if(canFind(exts, ext))
    {
        photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
        string dir = "./public/uploads/photos/";
        mkdirRecurse(dir);
        string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
        try moveFile(pic.tempPath, NativePath(fullpath));
        catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
    }
    e.photo = photopath;
}
if(e.phone.length == 0) e.phone = "(123) 456 7890";
if(e.payrate.length == 0) e.payrate = "none yet";
if(e.postcode.length == 0) e.postcode = "A1A 1A1";
empModel.addEmployee(e);
redirect("all_employees");
}

@auth
void getAllEmployees(string _authUser, string _error = null)
{
    string error = _error;
    Employee[] emps = empModel.getEmployees();
    bool loggedIn = m_user.loggedIn;
    render!("emplist.dt", emps, error, loggedIn);
}
```

```
@auth
void getEditEmployee(string _authUser, int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    bool loggedIn = m_user.loggedIn;
    string[] payrates = empModel.getPayrates();
    render!("empedit.dt", e, departments, payrates, provinces, error, loggedIn);
}

@errorDisplay!getEditEmployee
void postEditEmployee(Employee e, string prevPassw)
{
    import std.file;
    import std.path;
    import std.algorithm;

    string photopath = e.photo;
    auto pic = "picture" in request.files;
    if(pic !is null)
    {
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
            try moveFile(pic.tempPath, NativePath(fullpath));
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
        }
    }
}
```

```
        }
    }
    e.photo = photopath;
    if(e.phone.length == 0) e.phone = "(123) 456 7890";
    if(e.payrate.length == 0) e.payrate = "none yet";
    if(e.postcode.length == 0) e.postcode = "A1A 1A1";
    bool newpass = (prevPassw != e.passw) ? true : false;
    empModel.editEmployee(e, newpass);
    redirect("all_employees");
}

@auth
void getDeleteEmployee(string _authUser, int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    bool loggedIn = m_user.loggedIn;
    render!("empdelete.dt", e, error, loggedIn);
}

@errorDisplay!getDeleteEmployee
void postDeleteEmployee(int id)
{
    empModel.deleteEmployee(id);
    redirect("all_employees");
}

@auth
@errorDisplay!getAllEmployees
void postFindEmployee(string _authUser, string fname, string lname)
{
    import std.uni; //so we can use the toUpper() function
```

```
Employee e = empModel.findEmployee(toUpper(fname), toUpper(lname));
enforce(e != Employee.init, "Cannot find employee " ~ fname ~ " " ~ lname);
bool loggedIn = m_user.loggedIn;
render!("empshow.dt", e, loggedIn);
}
```

We made the EmployeeController inherit from BaseController.

## The login system and a new menu

It is desirable to let the menu reflect the login status. For example, once the login is successful, the menu item should change from ‘Login’ to ‘Logout’. Also, some parts of the menu should have different links depending on the login status.

Here is views\layout.dt:

```
doctype 5
html
  head
    title Employee Timekeeping System
    include cssclock
    include cssfooter
    include cssformgrid
    include csslayout
    include cssmenu
    include csstable
  body
    div.container
      nav
        include menu
      main.content
        block maincontent
      footer
        include footer
```

Here is views\menu.dt:

```
ul
  li
    a(href="/") Home
  li
```

```
a(href="#") Time cards
ul
  li
    a(href="timecards") View timecards
  li
    a(href="create_timesheet") Create timesheet
li
  a(href="#") Employees
  ul
    li
      a(href="all_employees") View employees
      -if(loggedIn)
        li
          a(href="#find_employee") Find an employee
      -else
        li
          a(href="#login") Find an employee
      li
        a(href="add_employee") New employee
ul.link-right
  li
  -if(loggedIn)
    a(href="logout") Logout
  -else
    a(href="#login") Login
div#find_employee.modal-form
  div.modal-form-wrapper-find_employee
    form.modal-form-grid(method="post", action="find_employee")
      input.text-control(name="fname", type="text", placeholder=" First name", required)
      input.text-control(name="lname", type="text", placeholder=" Last name", required)
      input#but-reset(type="reset", value="Clear")
      input#but-submit(type="submit", value="Find")
```

```
br
a.close(href="#close") Cancel
div#login.modal-form
div.modal-form-wrapper-login
form.modal-form-grid(method="post", action="login")
input.text-control(name="email", type="email", placeholder=" email address")
input.text-control(name="password", type="password", placeholder=" password")
input#but-reset(type="reset", value="Clear")
input#but-submit(type="submit", value="Login")
br
a.close(href="#close") Cancel
```

Here is the views\footer.dt which remains the same:

```
div.copyright Copyright &copy; The Lorem Ipsum Company 2023
```

Here is views\cssclock.dt:

```
:css
.clock-wrapper
{
  margin: 0 10px;
  text-align: center;
}

.clock-grid
{
  margin: 0 auto;
  padding: 0;
  width: 900px;
  height: auto;
  display: grid;
```

```
grid-template: 90px auto / repeat(12, 1fr);
grid-template-areas:
  "t0 t0 t0"
  "ci ci ci ci ci ci co co co co co co";
gap: 20px;
color: #444;
text-align: center;
}

.clock-heading
{
  margin-bottom: 10px;
  padding: 0;
  grid-area: t0;
  line-height: 90px;
  color: black;
  font-weight: bold;
  text-align: center;
}

#clock-title
{
  font-size: 30px;
}

.clock-text-center
{
  text-align: center;
}

.clock-area
{
```

```
font-size: 40px;
font-weight: bold;
text-align: center;
width: 370px;
height: 400px;
margin: 10px auto;
padding: 0;
}

#clock-in
{
  grid-area: ci;
}

#clock-out
{
  grid-area: co;
}

.clock-btn
{
  margin: 20px auto;
  padding: 10px 60px;
  border: none;
  background: #363636;
  width: 100%;
  transition: ease-in all 0.3s;
  color: #fff;
  height: auto;
  text-align: center;
  border-radius: 30px;
  font-size: 20px;
```

```
    font-weight: bold;
    text-decoration: none;
    cursor: pointer;
}

.clock-btn:hover
{
    background: brown;
    color: #fff;
}

#clock-time
{
    font-size: 24px;
    margin: 100px auto 0 auto;
    color: black;
}

#clock-form-select
{
    margin: 0 auto;
    text-align: center;
    font-size: 20px;
    font-weight: bold;
}

.success
{
    font-size: 16px;
    font-weight: bold;
    color: black;
    text-align: center;
```

```
}
```

Here is views\cssfooter.dt:

```
:css
footer
{
    position: fixed;
    bottom: 0;
    margin-top: 15px;
    background-color: #076;
    padding: 5px 0;
    width: 100%;
}

.copyright
{
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: 14px;
    text-align: center;
    color: white;
}
```

Here is views\cssformgrid.dt:

```
:css
.form-grid-wrapper
{
    width: 500px;
    margin: 20px;
    padding: 1px 20px 20px 0;
    border-radius: 20px;
```

```
}

.form-grid
{
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 10px;
  padding-top: 5px;
}

.form-grid-label
{
  width: 100%;
  font-size: 16px;
  text-align: right;
  padding: 2px;
}

.form-grid-input
{
  width: 100%;
  font-size: 14px;
  padding: 0;
}

.form-grid-field
{
  width: 100%;
  font-size: 16px;
  border-bottom: 1px solid black;
  padding: 2px;
}
```

```
.form-grid-button
{
    width: 100%;
    font-size: 14px;
    height: 30px;
    margin-top: 10px;
}

.form-hidden
{
    padding: 0;
    margin: 0;
    display: inline;
}
```

Here is views\csslayout.dt:

```
:css
html, body, container
{
    margin: 0;
    padding: 0;
    width: 100%;
    height: 100%;
}

.container
{
    display: grid;
    grid-template-rows: 38px auto 10px;
    background-color: cyan;
```

```
}

main
{
    margin: 0;
    padding: 40px 15px;
    font-family: Sans-serif;
    width: auto;
    height: 100%;
}

.text-control
{
    grid-column: 1 / 3;
}

#but-reset
{
    grid-column: 1 / 2;
}

#but-submit
{
    grid-column: 2 / 3;
}

.error-div
{
    margin-bottom: 5px;
}

.error-message
```

```
{  
    color: brown;  
    font-size: 16px;  
    font-weight: bold;  
    text-align: left;  
}  
  
.center-align  
{  
    margin: 0 auto;  
    text-align: center;  
}  
  
.right-align  
{  
    text-align: right;  
}
```

Here is views\cssmenu.dt:

```
:css  
nav  
{  
    position: fixed;  
    top: 0;  
    margin: 0;  
    padding: 0;  
    display: inline-block;  
    background: #076;  
    font-family: Verdana, Geneva, Tahoma, sans-serif;  
    width: 100%;  
}
```

```
nav ul
{
    margin:0;
    padding:0;
    list-style-type:none;
    float:left;
    display:inline-block;
}

nav ul li
{
    position: relative;
    margin: 0;
    float: left;
    display: inline-block;
}

li > a:only-child:after { content: ''; }

nav ul li a
{
    padding: 15px 20px;
    display: inline-block;
    color: white;
    text-decoration: none;
}

nav ul li a:hover
{
    opacity: 0.5;
}
```

```
nav ul li ul
{
  display: none;
  position: absolute;
  left: 0;
  background: #076;
  float: left;
  width: 190px;
}

nav ul li ul li
{
  width: 100%;
  border-bottom: 1px solid rgba(255,255,255,.3);
}

nav ul li ul li a
{
  padding: 10px 20px;
}

nav ul li:hover ul
{
  display: block;
}

.link-right
{
  float: right;
  margin-right: 20px;
}
```

```
.modal-form
{
    position: fixed;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    background: rgba(0,0,0,0.5);
    z-index: 99999;
    opacity: 0;
    pointer-events: none;
}

#login:target, #find_employee:target
{
    opacity: 1;
    pointer-events: auto;
}

.modal-form-grid
{
    display: grid;
    grid-template: 30px 30px 30px / 1fr 1fr;
    grid-gap: 10px;
}

.modal-form-wrapper-login
{
    width: 250px;
    position: absolute;
```

```
    right: 0;
    margin-top: 40px;
    padding: 25px 20px;
    border-radius: 10px;
    text-align: center;
    background-color: whitesmoke;
}

.modal-form-wrapper-find_employee
{
    width: 250px;
    position: relative;
    left: 220px;
    margin-top: 40px;
    padding: 25px 20px;
    border-radius: 10px;
    text-align: center;
    background-color: whitesmoke;
}
```

And here is views\csstable.dt:

```
:css
.table-wrapper
{
    margin: 10px 0;
    padding-bottom: 40px;
}

table
{
    padding: 10px 20px;
```

```
border-spacing: 0;
border-collapse: collapse;
}

table td, table th
{
margin: 0;
padding: 2px 10px;
text-align: left;
}

.no-border
{
border: 0;
}

tr:nth-child(odd)
{
background-color: #DADADA;
}
```

## The index, time in and time out templates

Here is the views\index.dt:

```
extends layout
block maincontent
div.main
  div.clock-wrapper
    div.clock-grid
      div.clock-heading#clock-title Lorem Ipsum Company Timekeeping System
      div#clock-in.clock-area
        img(src="images/timein.png", alt="photo of a clock")
        a.clock-btn(href="time_in") Punch In
      div#clock-out.clock-area
        img(src="images/timeout.png", alt="photo of a clock")
        a.clock-btn(href="time_out") Punch Out
```

This is the image I used for the punch in part which I named timein.png:



And this is the image I used for the punch out part, which I named timeout.png:



And these are the other images we used earlier:

The pencil.png:



And the trash.png:



Place all four pictures into public\images.

Here is the views\timein.dt:

```
extends layout
block maincontent
    div.main
        div.clock-wrapper
            div.clock-grid
                div.clock-heading#clock-title Lorem Ipsum Company Timekeeping System
                form#clock-in.clock-area(method="post", action="time_in")
                    div#clock-time
                        select#clock-form-select(name="empidname")
                            -foreach(e; emps)
                                option(value="#{e.empid} #{e.fname} #{e.lname}") #{e.empid} #{e.fname} #{e.lname}
                        input.clock-btn(type="submit", value="Punch In")
                    -if(error)
                        div.error-div
                            span.error-message #{error}
                    -else if(success)
                        div.success #{success}
```

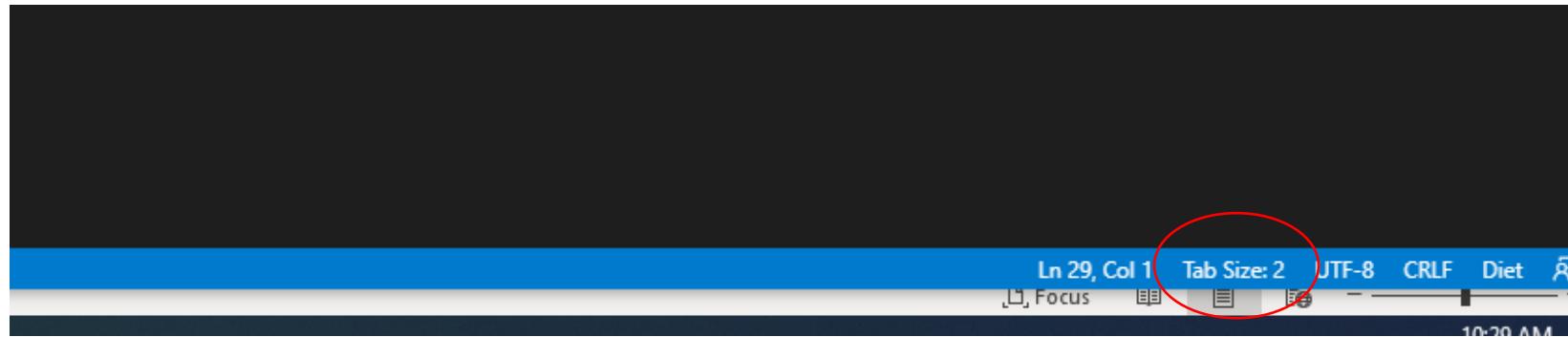
```
div#clock-out.clock-area
  img(src="images/timeout.png", alt="photo of a clock")
  a.clock-btn(href="time_out") Punch Out

:javascript
const displayTime = document.querySelector("#clock-time");
function showTime() {
  let time = new Date();
  displayTime.innerText = time.toLocaleString("en-CA", { hour12: true });
  setTimeout(showTime, 1000);
}
showTime();
```

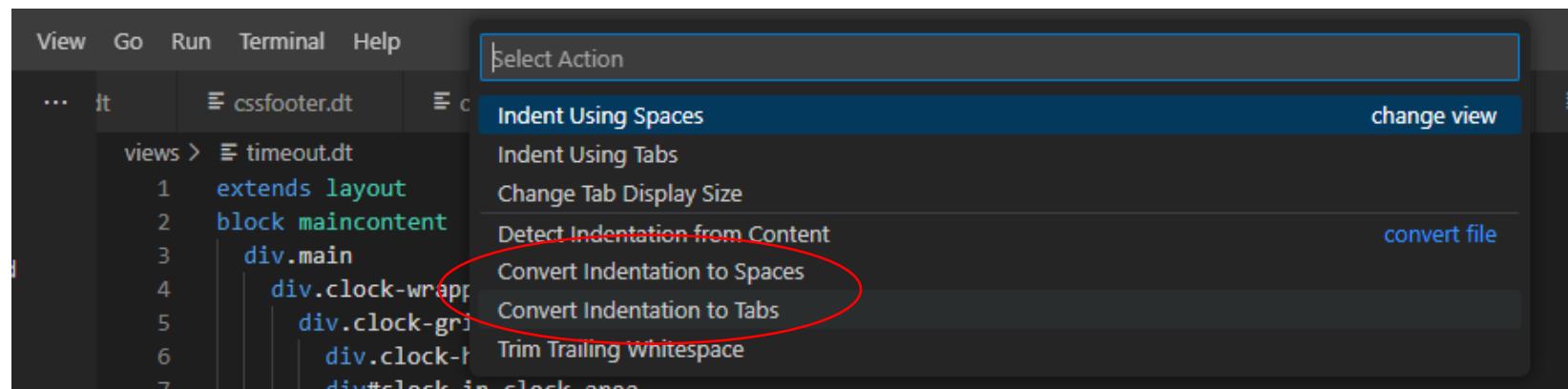
And here is views\timeout.dt:

```
-else if(success)
    div.success #{success}
:javascript
const displayTime = document.querySelector("#clock-time");
function showTime() {
    let time = new Date();
    displayTime.innerText = time.toLocaleString("en-CA", { hour12: true });
    setTimeout(showTime, 1000);
}
showTime();
```

As usual, make sure all the tabs are consistent or you will get errors when compiling. To make sure, click on the ‘Tab size:’ feature on the footer of VS Code.



And click on either ‘Convert indentation to Spaces’ or ‘Convert indentation to Tabs’, whichever is your preference. The important thing is to be consistent. Then save the file again.



## The employee views

Here is views\empadd.dt:

```
extends layout
block maincontent
-if(error)
  div.error-div
    span.error-message #{error}
div.form-grid-wrapper
  h2.center-align New employee details
  br
  form.form-grid(method="post", action="add_employee", enctype="multipart/form-data")
    label.form-grid-label Employee number
    input.form-grid-input(type="empid", name="e.empid", placeholder="employee number", required)
    label.form-grid-label Email address
    input.form-grid-input(type="email", name="e_email", placeholder="email@company.com", required)
    label.form-grid-label Password
    input.form-grid-input(type="password", name="e_passw", placeholder="password", required)
    label.form-grid-label First name
    input.form-grid-input(type="text", name="e_fname", placeholder="First name", required)
    label.form-grid-label Last name
    input.form-grid-input(type="text", name="e_lname", placeholder="Last name", required)
    label.form-grid-label Phone
    input.form-grid-input(type="text", name="e_phone", placeholder="Phone number")
    label.form-grid-label Department
    select#dept.form-grid-input(name="e_dept")
      -foreach(dep; departments)
        option(value="#{dep}") #{dep}
    label.form-grid-label Salary grade
    select#dept.form-grid-input(name="e_payrate")
      -foreach(pay; payrates)
```

```

option(value="#{pay}") #{pay}
label.form-grid-label Street address (no city)
input.form-grid-input(type="text", name="e_street", placeholder="Street address", required)
label.form-grid-label City
input.form-grid-input(type="text", name="e_city", placeholder="City", required)
label.form-grid-label Province
select#province.form-grid-input(name="e_province")
-foreach(prov; provinces)
    option(value="#{prov[0]}") #{prov[1]}
label.form-grid-label Postal code
input.form-grid-input(type="text", name="e_postcode", placeholder="A1A 1A1")
label.form-grid-label ID Picture
input.form-grid-input(type="file", name="picture")
input(type="hidden", name="e_photo")
input(type="hidden", name="e_id", value="1")
input.form-grid-button(type="reset", value="Clear form")
input.form-grid-button(type="submit", value="Submit")

```

Here is views\empdelete.dt:

```

extends layout
block maincontent
    div.form-grid-wrapper
        -if(error)
            div.error-div
                span.error-message #{error}
        h2.center-align Delete #{e.fname} #{e.lname}'s record?
        br
        form.form-grid(method="post", action="delete_employee")
            span.form-grid-label Employee number:
            span.form-grid-field #{e.empid}
            span.form-grid-label Email address:

```

```
span.form-grid-field #{e.email}
span.form-grid-label First name:
span.form-grid-field #{e.fname}
span.form-grid-label Last name:
span.form-grid-field #{e.lname}
span.form-grid-label Phone:
span.form-grid-field #{e.phone}
span.form-grid-label Department:
span.form-grid-field #{e.deprt}
span.form-grid-label Salary grade:
span.form-grid-field #{e.payrate}
span.form-grid-label Street address:
span.form-grid-field #{e.street}
span.form-grid-label City:
span.form-grid-field #{e.city}
span.form-grid-label Province:
span.form-grid-field #{e.province}
span.form-grid-label Postal code:
span.form-grid-field #{e.postcode}
span.form-grid-label ID Picture:

```

Here is views\empedit.dt:

```
extends layout
block maincontent
    div.form-grid-wrapper
        -if(error)
```

```
div.error-div
    span.error-message #{error}
h2.center-align Edit #{e.fname} #{e.lname} details
br
form.form-grid(method="post", action="edit_employee", enctype="multipart/form-data")
    label.form-grid-label Employee number
    input.form-grid-input(type="empid", name="e.empid", value="#{e.empid}")
    label.form-grid-label Email address
    input.form-grid-input(type="email", name="e_email", value="#{e.email}")
    label.form-grid-label Password
    input.form-grid-input(type="password", name="e_passw", value="#{e.passw}")
    label.form-grid-label First name
    input.form-grid-input(type="text", name="e_fname", value="#{e.fname}")
    label.form-grid-label Last name
    input.form-grid-input(type="text", name="e_lname", value="#{e.lname}")
    label.form-grid-label Phone
    input.form-grid-input(type="text", name="e_phone", value="#{e.phone}")
    label.form-grid-label Department
    select#dept.form-grid-input(name="e_dept")
        -foreach(dep; departments)
            -if(dep == e.deprt)
                option(value="#{dep}", selected) #{dep}
            -else
                option(value="#{dep}") #{dep}
    label.form-grid-label Salary grade
    select#paygd.form-grid-input(name="e_payrate", value="#{e.payrate}")
        -foreach(pay; payrates)
            -if(pay == e.payrate)
                option(value="#{pay}", selected) #{pay}
            -else
                option(value="#{pay}") #{pay}
    label.form-grid-label Street address (no city)
```

```

input.form-grid-input(type="text", name="e_street", value="#{e.street}")
label.form-grid-label City
input.form-grid-input(type="text", name="e_city", value="#{e.city}")
label.form-grid-label Province
select#province.form-grid-input(name="e_province", value="#{e.province}")
    -foreach(prov; provinces)
        -if(prov[0] == e.province)
            option(value="#{prov[0]}", selected) #{prov[1]}
        -else
            option(value="#{prov[0]}") #{prov[1]}
label.form-grid-label Postal code
input.form-grid-input(type="text", name="e_postcode", value="#{e.postcode}")
label.form-grid-label ID Picture
img(src="#{e.photo}", height="100px")
div
    input.form-grid-input(type="file", name="picture")
    input(type="hidden", name="prevPassw", value="#{e.passw}")
    input(type="hidden", name="e_photo", value="#{e.photo}")
    input(type="hidden", name="e_id", value="#{e.id}")
    input(type="hidden", name="id", value="#{e.id}")
    a(href="all_employees")
        button.form-grid-button(type="button") Cancel
    input.form-grid-button(type="submit", value="Submit")

```

Here is views\emplist.dt:

```

extends layout
block maincontent
    h2 Lorem Ipsum Employees
    -if(error)
        div.error-div
            span.error-message #{error}

```

```

div.table-wrapper
table
tr
th Emp #
th Name
th Department
th Phone number
th Email address
th Action
-foreach(e; emps)
tr
td #{e.empid}
td #{e.fname} #{e.lname}
td #{e.deprt}
td #{e.phone}
td #{e.email}
td &nbsp;
form.form-hidden(method="get", action="edit_employee")
input(type="hidden", name="id", value="#{e.id}")
input(type="image", src="images/pencil.png", height="15px")
| &nbsp;
form.form-hidden(method="get", action="delete_employee")
input(type="hidden", name="id", value="#{e.id}")
input(type="image", src="images/trash.png", height="15px")
| &nbsp;

```

Here is views\empshow.dt:

```

extends layout
block maincontent
div.form-grid-wrapper
h2.center-align #{e.fname} #{e.lname} details

```

```
br
div.form-grid
    span.form-grid-label Employee number:
    span.form-grid-field #{e.empid}
    span.form-grid-label Email address:
    span.form-grid-field #{e.email}
    span.form-grid-label First name:
    span.form-grid-field #{e.fname}
    span.form-grid-label Last name:
    span.form-grid-field #{e.lname}
    span.form-grid-label Phone:
    span.form-grid-field #{e.phone}
    span.form-grid-label Department:
    span.form-grid-field #{e.deprt}
    span.form-grid-label Salary grade:
    span.form-grid-field #{e.payrate}
    span.form-grid-label Street address:
    span.form-grid-field #{e.street}
    span.form-grid-label City:
    span.form-grid-field #{e.city}
    span.form-grid-label Province:
    span.form-grid-field #{e.province}
    span.form-grid-label Postal code:
    span.form-grid-field #{e.postcode}
    span.form-grid-label ID Picture:
    img(src="#{e.photo}", height="80px")
a(href="all_employees")
    button.form-grid-button(type="button") Close
a(href="edit_employee?id=#{e.id}")
    button.form-grid-button(type="button") Edit
```

## The home controller and app.d

We don't have a controller that displays the home page which is in views\index.dt. Let's create a new controller, which we will name the HomeController, which will also inherit from BaseController.

Here is the source\homecontrol.d file:

```
module homecontrol;

import vibe.vibe;
import adminmodel;
import basecontrol;

class HomeController : BaseController
{
    void index(string _error = null)
    {
        string error = _error;
        bool loggedIn = m_user.loggedIn;
        render!("index.dt", error, loggedIn);
    }

    @errorDisplay!index
    void postLogin(string email, string password)
    {
        bool isAdmin = adminModel.getAdmin(email, password);
        enforce(isAdmin, "Email and password combination not found.");
        User user = m_user;
        user.loggedIn = true;
        user.email = email;
        m_user = user;
        redirect("timecards");
    }
}
```

```
}

void getLogout()
{
    m_user = User.init;
    terminateSession;
    redirect("/");
}
```

Here we added the postLogin() and getLogout() methods which do not belong to other controllers.

Since we created another controller, we have to register it to the router as another web interface, so we have to make changes to app.d.

Here is source\app.d:

```
import vibe.vibe;
import homecontrol;
import empcontrol;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];
    settings.sessionStore = new MemorySessionStore;

    auto router = new URLRouter;
    router.get("*", serveStaticFiles("public/"));
    router.registerWebInterface(new HomeController);
    router.registerWebInterface(new EmployeeController);
```

```
auto listener = listenHTTP(settings, router);
scope (exit) listener.stopListening();

runApplication();
}
```

Now we can test the whole thing.

Compile, run and refresh the browser. If you see this screen, you did good.

Home   Time cards   Employees   Login

## **Lorem Ipsum Company Timekeeping System**

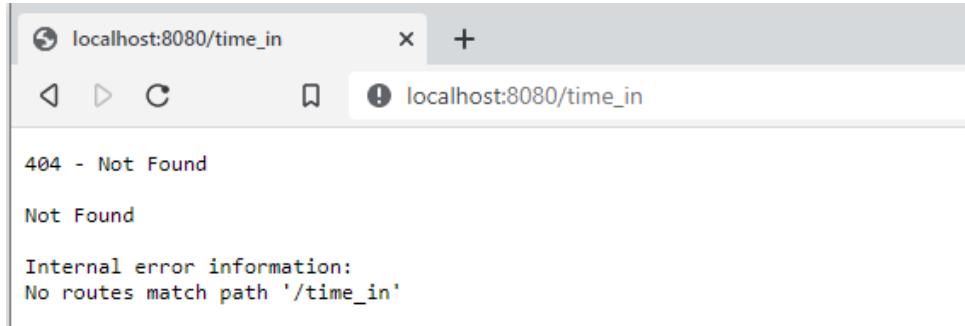


**Punch In**      **Punch Out**

Copyright © The Lorem Ipsum Company 2023

## Displaying the punch-in page

If you click on the ‘Punch In’ button, you only get an error like this:



That's because we don't have a handler yet for the /time\_in link. Let's create the timecard controller to handle all timecard-related actions.

Here is source\cardcontrol.d:

```
module cardcontrol;

import vibe.vibe;
import basecontrol;
import cardmodel;
import sheetmodel;
import empmodel;

class TimecardController : BaseController
{
    string success = null;

    void getTimeIn(string _error = null)
```

```
{  
    Employee[] emps = empModel.getEmployees;  
    bool loggedIn = m_user.loggedIn;  
    string error = _error;  
    render!("timein.dt", emps, loggedIn, error, success);  
}  
}
```

Since TimecardController is another web interface, we have to register it to the router in app.d.

Here is source\app.d:

```
import vibe.vibe;  
import homecontrol;  
import empcontrol;  
import cardcontrol;  
  
void main()  
{  
    auto settings = new HTTPServerSettings;  
    settings.port = 8080;  
    settings.bindAddresses = [":1", "127.0.0.1"];  
    settings.sessionStore = new MemorySessionStore;  
  
    auto router = new URLRouter;  
    router.get("*", serveStaticFiles("public/"));  
    router.registerWebInterface(new HomeController);  
    router.registerWebInterface(new EmployeeController);  
    router.registerWebInterface(new TimecardController);  
  
    auto listener = listenHTTP(settings, router);  
    scope (exit) listener.stopListening();
```

```
    runApplication();
}
```

Before we test it, let us create some database tables.

Edit source\homecontrol.d:

```
module homecontrol;

import vibe.vibe;
import basecontrol;
import adminmodel;

class HomeController : BaseController
{
    this()
    {
        adminModel.createTable;
    }

    void index(string _error = null)
    {
        string error = _error;
        bool loggedIn = m_user.loggedIn;
        render!("index.dt", error, loggedIn);
    }

    @errorDisplay!index
    void postLogin(string email, string password)
    {
        bool isAdmin = adminModel.getAdmin(email, password);
    }
}
```

```

enforce(isAdmin, "Email and password combination not found.");
User user = m_user;
user.loggedIn = true;
user.email = email;
m_user = user;
redirect("timecards");
}

void getLogout()
{
    m_user = User.init;
    terminateSession;
    redirect("/");
}
}

```

So that when the HomeController is instantiated, the admins table is created. After compilation, you should delete those lines.

Edit source\empcontrol.d by adding this code at the beginning of the class declaration:

```

class EmployeeController : BaseController
{
    this()
    {
        empModel.createTable;
        empModel.createPayrates;
    }

    @auth
    void getAddEmployee(string _authUser, string _error = null)
    {

```

This means the employees table and the payrates table will be created when EmployeeController is instantiated. Go ahead and test the whole thing by compiling, running and refreshing the browser.

```
C:\vibeprojects\loremtime>dub
```

```
Starting Performing "debug" build using C:\D\dmd2\windows\bin64\dmd.exe for x86_64.
```

```
Up-to-date taggedalgebraic 0.11.22: target for configuration [library] is up to date.
```

```
Up-to-date eventcore 0.9.25: target for configuration [winapi] is up to date.
```

```
...
```

And refreshing the browser, you see this:

The screenshot shows a web page titled "Lorem Ipsum Company Timekeeping System". At the top, there is a navigation bar with links for "Home", "Time cards", "Employees", and "Login". Below the title, there are two large alarm clock icons. The left clock has its hands set to approximately 3:55 and is labeled "Punch In". The right clock has its hands set to approximately 4:45 and is labeled "Punch Out". At the bottom of the page, a copyright notice reads "Copyright © The Lorem Ipsum Company 2023".

We have seen this before, so nothing new there.

Then **erase** this code from source\empcontrol file or else we will always be erasing the employee data:

```
---this()
---{
---    empModel.createTable;
---    empModel.createPayrates;
---}
```

And **erase** this from source\homecontrol file or else we will always be erasing the admin data:

```
---this()
---{
---    adminModel.createTable;
---}
```

After compilation, **erase the lines** above or else we will always be recreating those tables with all the data gone every time we compile and run.

## Adding employees

After compiling, running and refreshing the browser, click on the ‘Punch In’ button. You should see something like this.

Home   Time cards   Employees   Login

### **Lorem Ipsum Company Timekeeping System**

2023-04-09, 1:12:54 p.m.

Punch In



Punch Out



Copyright © The Lorem Ipsum Company 2023

The dropdown list of employees doesn't show because our table of employees has no data since we just created it. So let's add some employees to the database first.

[Home](#)[Time cards](#)[Employees](#)[Login](#)

Lorem

[View employees](#)[Find an employee](#)[New employee](#)**2023-04-09, 1:16:30 p.m.****Punch In****Punch Out**

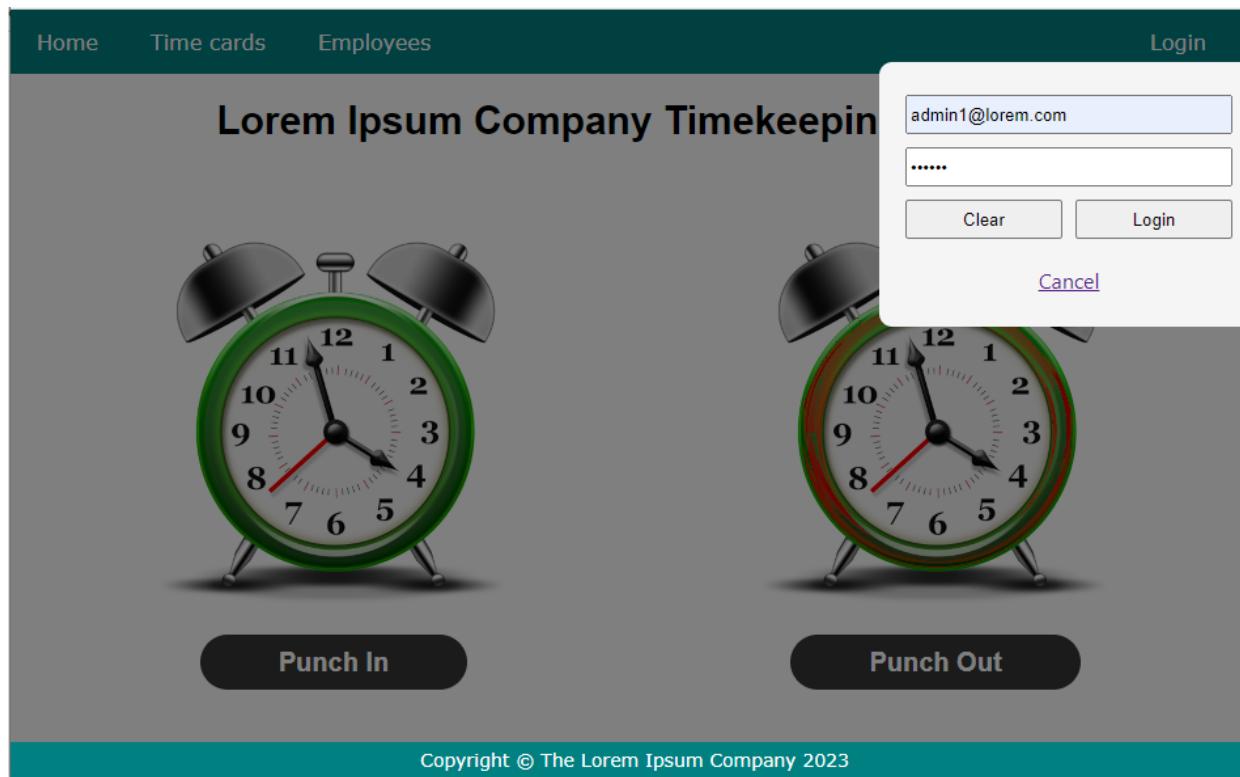
localhost:8080/add\_employee

Copyright © The Lorem Ipsum Company 2023

But then we are redirected to the Login dialogue box, so let's log in first.

User: admin1@lorem.com

Password: secret



After logging in, we are brought back to the index page, but this time, the 'Login' prompt on the menu becomes 'Logout', indicating that we are now logged in.



So click on the 'New employee' once again to add new employees. Which brings us to this screen.

### New employee details

Employee number

Email address

Password

First name

Last name

Phone

Department ▼

Salary grade ▼

Street address (no city)

City

Province ▼

Postal code

ID Picture  No file chosen

Clear formSubmit

Add some new employees so we can fill up the employees list like this:

## Lorem Ipsum Employees

Emp #	Name	Department	Phone number	Email address	Action
1002	Emma Rob	Maintenance	1234567890	emma@lorem.com	 
1001	Camille Bell	Production	1234567890	camille@lorem.com	 
1003	Emma Stone	Shipping	0987654321	emmas@lorem.com	 
1004	Emma Watson	Accounting	1234567890	emmw@lorem.com	 
1005	Emmi Clarke	Management	0987654321	emmi@lorem.com	 
1006	Hailee Stein	Shipping	0987654321	hailee@lorem.com	 
1007	Heidi Lamar	Production	1234567890	heidi@lorem.com	 
1008	Jen Law	Management	1234567890	jen@lorem.com	 
1009	Jessica Alba	Management	1234567890	jessica@lorem.com	 
1010	Kate Beck	IT Services	1234567890	kate@lorem.com	 
1011	Mila Kunis	Marketing	1234567890	mila@lorem.com	 
1012	Nicole Kidman	Maintenance	1234567890	nico@lorem.com	 
1013	Priya Chopra	Purchasing	0987654321	priya@lorem.com	 
1014	Scarlett Johann	Production	0987654321	scarlett@lorem.com	 
1015	Who Ever	Accounting	1234567890	who@lorem.com	 
1016	Who2 Ever	Marketing	1234567890	who2@lorem.com	 

Now let's go back to the home page and click on the 'Punch In' button again. This time, the dropdown list of employees is populated.

## Lore ipsum Company Timekeeping System

2023-04-10, 10:56:27 a.m.

1002 Emma Rob ▾

- 1002 Emma Rob
- 1001 Camille Bell
- 1003 Emma Stone
- 1004 Emma Watson
- 1005 Emmi Clarke
- 1006 Hailee Stein
- 1007 Heidi Lamar
- 1008 Jen Law
- 1009 Jessica Alba
- 1010 Kate Beck



Punch Out

ght © The Lore ipsum Company 2023

Select an employee and click on the ‘Punch In’ button. You see this error:

[Home](#)[Time cards](#)[Employees](#)[Logout](#)

## Lorem Ipsum Company Timekeeping System

2023-04-09, 2:27:12 p.m.

1002 Emma Rob ▾

Punch In

MySQL error: Table 'loremdb.timecards' doesn't  
exist



Punch Out

Copyright © The Lorem Ipsum Company 2023

So let's create the timecards table.

Edit source\cardcontrol.d:

```
module cardcontrol;

import vibe.vibe;
import basecontrol;
import cardmodel;
import sheetmodel;
import empmodel;
```

```
class TimecardController : BaseController
{
    this()
    {
        cardModel.createTable;
    }

    string success = null;

    void getTimeIn(string _error = null)
    {
        Employee[] emps = empModel.getEmployees;
        bool loggedIn = m_user.loggedIn;
        string error = _error;
        render!("timein.dt", emps, loggedIn, error, success);
    }
}
```

We only added these lines to call the `createTable()` method on the timecard model:

```
this()
{
    cardModel.createTable;
}
```

After compiling, **delete these lines** or else you will always lose the timecard data every time you compile and run the app.

```
—this()
—{
—    cardModel.createTable;
—}
```

## Punching in

When punching in, the system should check if the employee already punched in earlier before a new record is created so there is no duplication.

After compiling, go back to the home page and click on the ‘Punch In’ button. We get this error:

```
404 - Not Found  
Not Found  
Internal error information:  
No routes match path '/time_in'
```

So let's edit source\cardcontrol.d and append this code:

```
@errorDisplay!getTimeIn  
void postTimeIn(string empidname)  
{  
    string empid = empidname[0..4];  
    string fullname = empidname[5..$];  
    //the record should not exist, otherwise employee already punched in earlier  
    enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in today!");  
    bool timedin = cardModel.timeIn(empid, fullname);  
    if(timedin) success = fullname ~ " punched in successfully.";  
    else success = null;  
    redirect("time_in");  
}
```

The lines

```
string empid = empidname[0..4];
```

```
string fullname = empidname[5..$];
```

mean we are extracting a substring from a string. In D, parts of arrays are called slices, so a substring is a slice. Strings in D are treated like an array of characters so getting substrings out of strings is relatively trivial. Indexing starts with zero as in other languages, but ranges are exclusive of the upper bound. The \$ character refers to the length of the array. This means the slice 0..4 actually gets characters 0 to 3, which are four characters.

An example of an empidname data looks like this:

```
empidname = '1002 Emma Rob'
```

So the line

```
string empid = empidname[0..4];
```

means the variable empid will get the characters from 0 up to 3, which is '1002'.

The line

```
string fullname = empidname[5..$];
```

means the variable fullname will get the characters from index 5 up to the end of the string, which is 'Emma Rob'.

The postTimeIn() method is calling two cardModel methods, noTimeIn() and timeIn(), so let's write them.

Open source\cardmodel.d and append this code:

```
bool noTimeIn(string empid)
{
    string sql = "select * from timecards where empid=? and date(timein)=?";
    Prepared pstmt = conn.prepare(sql);
    string today = getToday;
    pstmt.setArgs(empid, today);
```

```
Row[] rows = conn.query(pstmt).array;
if(rows.length == 0) return true;
return false;
}

bool timeIn(string empid, string fullname)
{
    string sql = "insert into timecards(empid, fullname, timein) values(?, ?, now())";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(empid, fullname);
    if(conn.exec(pstmt) > 0) return true;
    return false;
}

string getToday()
{
    Row[] rows = conn.query("select curdate() + 0").array;
    return to!string(rows[0][0]);
}
```

The noTimeIn() method checks if there is no record yet in the table and is calling the getToday() method, so we added that too.

Compile, run and refresh the browser. Then click on the ‘Punch In’ button again. Select an employee and click ‘Punch In’.

This time we get a successful message.

## Lorem Ipsum Company Timekeeping System

2023-04-09, 2:37:41 p.m.

1002 Emma Rob ▾

**Punch In**

Emma Rob punched in successfully.



**Punch Out**

Copyright © The Lorem Ipsum Company 2023

So go ahead and punch in all the other employees.

Here is the full source\cardcontrol.d so far:

```
module cardcontrol;

import vibe.vibe;
import basecontrol;
import cardmodel;
import sheetmodel;
```

```

import empmode;
class TimecardController : BaseController
{
    string success = null;

    void getTimeIn(string _error = null)
    {
        Employee[] emps = empModel.getEmployees;
        bool loggedIn = m_user.loggedIn;
        string error = _error;
        render!("timein.dt", emps, loggedIn, error, success);
    }

    @errorDisplay!getTimeIn
    void postTimeIn(string empidname)
    {
        string empid = empidname[0..4];
        string fullname = empidname[5..$];
        //the record should not exist, otherwise employee already punched in earlier
        enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in today!");
        bool timedin = cardModel.timeIn(empid, fullname);
        if(timedin) success = fullname ~ " punched in successfully.";
        else success = null;
        redirect("time_in");
    }
}

```

And here is the full source\cardmodel.d so far:

```
module cardmodel;
```

```
import mysql;
import std.array;
import std.conv;
import basemodel;

struct Timecard
{
    int id;
    string empid;
    string fullname;
    string timein;
    string timeout;
    float hours;
}

class TimecardModel : BaseModel
{
    void createTable()
    {
        string sql = "drop table if exists timecards";
        conn.exec(sql);
        sql =
            "create table timecards
            (
                id int auto_increment primary key,
                empid char(4) not null references employees(empid),
                fullname varchar(50) not null,
                timein datetime not null default current_timestamp,
                timeout datetime,
                hours float default 0.0
            )";
        conn.exec(sql);
    }
}
```

```
}

bool noTimeIn(string empid)
{
    string sql = "select * from timecards where empid=? and date(timein)=?";
    Prepared pstmt = conn.prepare(sql);
    string today = getToday();
    pstmt.setArgs(empid, today);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return true;
    return false;
}

bool timeIn(string empid, string fullname)
{
    string sql = "insert into timecards(empid, fullname, timein) values(?, ?, now())";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(empid, fullname);
    if(conn.exec(pstmt) > 0) return true;
    return false;
}

string getToday()
{
    Row[] rows = conn.query("select curdate() + 0").array;
    return to!string(rows[0][0]);
}
}
```

## Viewing the timecards

Click on the ‘View timecards’ on the menu to see the timecards.

Screenshot of the Ipsum Company Timekeeping System interface:

- Header: Home, Time cards, Employees, Logout
- Left sidebar menu:
  - View timecards** (circled in red)
  - Create timesheet
- Main title: Ipsum Company Timekeeping System
- Date and time: 2023-04-09, 2:42:24 p.m.
- User dropdown: 1002 Emma Rob ▾
- Punch In button
- Status message: Who2 Ever punched in successfully.
- Right side: A 3D rendering of a traditional alarm clock.

But then you see this error:

Screenshot of a browser showing a 404 Not Found error:

- Address bar: localhost:8080/timecards
- Error message:

404 - Not Found  
Not Found  
Internal error information:  
No routes match path '/timecards'

Edit source\cardcontrol.d and append this code:

```
void getTimecards(string _error = null)
{
    string error = _error;
    string today = getToday;
    bool loggedIn = m_user.loggedIn;
    Timecard[] cards = cardModel.getTimecards(loggedIn);
    render!("cardlist.dt", cards, today, error, loggedIn);
}
```

This code calls cardModel.getTimecards(), which we haven't written yet, so let's write that.

Edit source\cardmodel.d and append this code:

```
Timecard[] getTimecards(bool loggedIn)
{
    Timecard[] cards;
    string sql;
    if(loggedIn)
        sql = "select * from timecards order by timein desc";
    else
        sql = "select * from timecards where date(timein) = curdate() order by timein desc";
    Row[] rows = conn.query(sql).array;
    if(rows.length != 0)
    {
        foreach(row; rows)
        {
            Timecard c;
            c.id = to!int(to!string(row[0]));
            c.empid = to!string(row[1]);
            c.fullname = to!string(row[2]);
        }
    }
}
```

```

        c.timein = to!string(row[3]);
        c.timeout = to!string(row[4]);
        c.hours = to!float(to!string(row[5]));
        cards ~= c;
    }
}
return cards;
}

```

Administrators who are logged in get to view all the timecards regardless of date. However, ordinary employees only get to view the timecards of the day so they can review their status for the day.

The cardControl.getTimecards() method needs the cardlist.dt template, so let's create it.

Here is the views\cardlist.dt:

```

extends layout
block maincontent
div.table-wrapper
-if(loggedIn)
    h2 All time cards
-else
    h2 #{today}
table
tr
th Emp #
th Name
th Time in
th Time out
th Hours
-if(loggedIn)
    th Action

```

```
-foreach(c; cards)
  tr
    td #{c.empid}
    td #{c.fullname}
    td #{c.timein}
    td #{c.timeout}
    td #{c.hours}
    -if(loggedIn)
      td &nbsp;
      form.form-hidden(method="get", action="edit_timecard")
        input(type="hidden", name="id", value="#{c.id}")
        input(type="image", src="images/pencil.png", height="15px")
      | &nbsp;
      form.form-hidden(method="get", action="delete_timecard")
        input(type="hidden", name="id", value="#{c.id}")
        input(type="image", src="images/trash.png", height="15px")
      | &nbsp;
```

Compile, run and refresh the browser. If you are not logged in, you see the list of time cards only for the day.

**2023-Apr-09**

Emp #	Name	Time in	Time out	Hours
1016	Who2 Ever	2023-Apr-09 14:42:10	null	0
1015	Who Ever	2023-Apr-09 14:42:05	null	0
1014	Scarlett Johann	2023-Apr-09 14:42:01	null	0
1013	Priya Chopra	2023-Apr-09 14:41:57	null	0
1012	Nicole Kidman	2023-Apr-09 14:41:53	null	0
1011	Mila Kunis	2023-Apr-09 14:41:48	null	0
1010	Kate Beck	2023-Apr-09 14:41:37	null	0
1009	Jessica Alba	2023-Apr-09 14:41:34	null	0
1008	Jen Law	2023-Apr-09 14:41:31	null	0
1007	Heidi Lamar	2023-Apr-09 14:41:28	null	0
1006	Hailee Stein	2023-Apr-09 14:41:25	null	0
1005	Emmi Clarke	2023-Apr-09 14:41:22	null	0
1004	Emma Watson	2023-Apr-09 14:41:18	null	0
1003	Emma Stone	2023-Apr-09 14:41:15	null	0
1001	Camille Bell	2023-Apr-09 14:41:08	null	0
1002	Emma Rob	2023-Apr-09 14:37:33	null	0

So here's the full source\cardcontrol.d so far:

```
module cardcontrol;

import vibe.vibe;
import basecontrol;
import cardmodel;
import sheetmodel;
import empmodel;

class TimecardController : BaseController
{
    string success = null;
```

```

void getTimeIn(string _error = null)
{
    Employee[] emps = empModel.getEmployees;
    bool loggedIn = m_user.loggedIn;
    string error = _error;
    render!("timein.dt", emps, loggedIn, error, success);
}

@errorDisplay!getTimeIn
void postTimeIn(string empidname)
{
    string empid = empidname[0..4];
    string fullname = empidname[5..$];
    //the record should not exist, otherwise employee already punched in earlier
    enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in today!");
    bool timedin = cardModel.timeIn(empid, fullname);
    if(timedin) success = fullname ~ " punched in successfully.";
    else success = null;
    redirect("time_in");
}

void getTimecards(string _error = null)
{
    string error = _error;
    string today = getToday;
    bool loggedIn = m_user.loggedIn;
    Timecard[] cards = cardModel.getTimecards(loggedIn);
    render!("cardlist.dt", cards, today, error, loggedIn);
}

```

And here is the full source\cardmodel.d so far:

```
module cardmodel;

import mysql;
import std.array;
import std.conv;
import basemodel;

struct Timecard
{
    int id;
    string empid;
    string fullname;
    string timein;
    string timeout;
    float hours;
}

class TimecardModel : BaseModel
{
    void createTable()
    {
        string sql = "drop table if exists timecards";
        conn.exec(sql);
        sql =
            "create table timecards
            (
                id int auto_increment primary key,
                empid char(4) not null references employees(empid),
                fullname varchar(50) not null,
                timein datetime not null default current_timestamp,
                timeout datetime,
                hours float
            )";
        conn.exec(sql);
    }
}
```

```
    hours float default 0.0
    )";
conn.exec(sql);
}

bool noTimeIn(string empid)
{
    string sql = "select * from timecards where empid=? and date(timein)=?";
    Prepared pstmt = conn.prepare(sql);
    string today = getToday();
    pstmt.setArgs(empid, today);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return true;
    return false;
}

bool timeIn(string empid, string fullname)
{
    string sql = "insert into timecards(empid, fullname, timein) values(?, ?, now())";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(empid, fullname);
    if(conn.exec(pstmt) > 0) return true;
    return false;
}

string getToday()
{
    Row[] rows = conn.query("select curdate() + 0").array;
    return to!string(rows[0][0]);
}

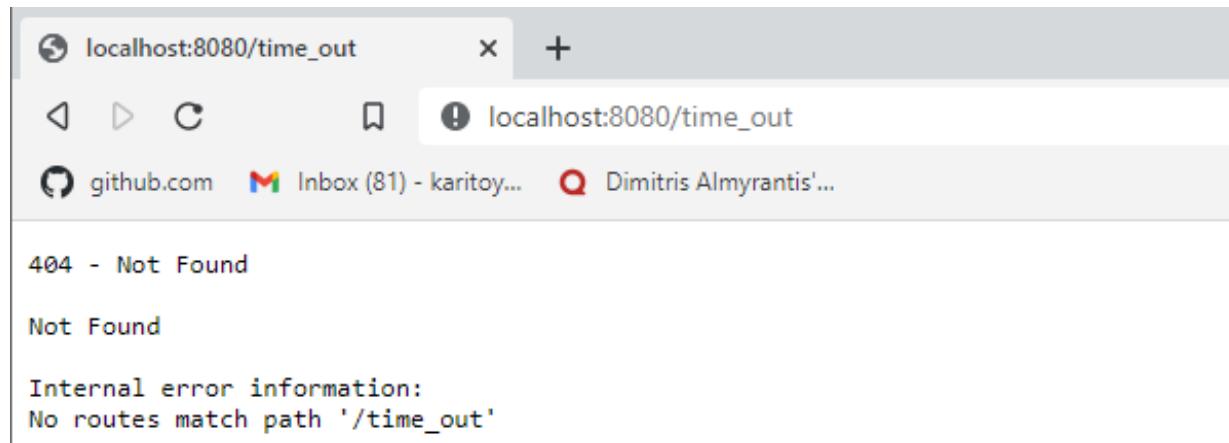
Timecard[] getTimecards(bool loggedIn)
```

```
{  
    Timecard[] cards;  
    string sql;  
    if(loggedIn)  
        sql = "select * from timecards order by timein desc";  
    else  
        sql = "select * from timecards where date(timein) = curdate() order by timein desc";  
    Row[] rows = conn.query(sql).array;  
    if(rows.length != 0)  
    {  
        foreach(row; rows)  
        {  
            Timecard c;  
            c.id = to!int(to!string(row[0]));  
            c.empid = to!string(row[1]);  
            c.fullname = to!string(row[2]);  
            c.timein = to!string(row[3]);  
            c.timeout = to!string(row[4]);  
            c.hours = to!float(to!string(row[5]));  
            cards ~= c;  
        }  
    }  
    return cards;  
}  
}
```

## Punching out

When an employee punches out, the system should check first if the employee already punched in earlier, meaning, the record already exists and the timeout column having a null value. If not, the employee cannot punch out since there is no timeout to edit. If the record exists, then the app should check if the employee has already punched out earlier, in that case the employee cannot punch out again.

As of now, when you click the ‘Punch Out’ button on the home page, this is what you get:



So edit source\cardcontrol.d and append this code:

```
void getTimeOut(string _error = null)
{
    Employee[] emps = empModel.getEmployees();
    bool loggedIn = m_user.loggedIn;
    string error = _error;
    render!("timeout.dt", emps, loggedIn, error, success);
}
```

```

@errorDisplay!getTimeOut
void postTimeOut(string empidname)
{
    string empid = empidname[0..4];
    string fullname = empidname[5..$];
    //the record should exist, otherwise employee did not punch in today
    enforce(!cardModel.noTimeIn(empid), fullname ~ ", you did not punch in today.");
    //the timeout field should be null, otherwise employee already punched out
    enforce(cardModel.noTimeOut(empid), fullname ~ ", you already punched out today!");
    bool timedout = cardModel.timeOut(empid);
    if(timedout) success = fullname ~ " punched out successfully.";
    else success = null;
    redirect("time_out");
}

```

The postTimeOut() method is calling two new methods from cardModel: noTimeOut() and timeOut(), so let's write them.

Edit source\cardmodel.d and append this code:

```

bool noTimeOut(string empid)
{
    string sql = "select * from timecards where timeout is null and empid=? and date(timein)=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(empid, getToday());
    Row[] rows = conn.query(pstmt).array;
    if(rows.length != 0) return true;
    return false;
}

bool timeOut(string empid)
{
    // we want to include the fraction of the hour
}

```

```

// so we use the MySQL function time_to_sec()
// 3600 is seconds in an hour
// but we round out the quotient to just two decimals
String sql =
"update timecards set timeout=now(),
    hours=round((time_to_sec(now())-time_to_sec(timein))/3600,2)
    where empid=? and date(timein)=curdate()";
Prepared pstmt = conn.prepareStatement(sql);
pstmt.setArgs(empid);
if(conn.exec(pstmt) > 0) return true;
return false;
}

```

Then compile, run and go back to the home page and click on the ‘Punch Out’ button. This time, the punch out screen appears.

The screenshot shows the homepage of the 'Lorem Ipsum Company Timekeeping System'. At the top, there is a navigation bar with links for 'Home', 'Time cards', 'Employees', and 'Login'. Below the navigation bar, the title 'Lorem Ipsum Company Timekeeping System' is displayed. To the left, there is a large green alarm clock icon. To the right of the clock, the current date and time are shown as '2023-04-09, 10:43:15 p.m.'. Below the date and time is a dropdown menu containing the text '1002 - Emma Rob'. Underneath the dropdown is a dark grey button labeled 'Punch Out'. At the bottom left of the page is another dark grey button labeled 'Punch In'. At the very bottom of the page, there is a footer bar with the copyright notice 'Copyright © The Lorem Ipsum Company 2023'.

Go ahead and punch out some employees. You get a message each time.

Either successful



2023-04-09, 10:45:24 p.m.

1002 - Emma Rob ▾

Punch Out

Emma Rob punched out successfully.

Or unsuccessful



Punch In

2023-04-09, 10:47:56 p.m.

1002 - Emma Rob ▾

Punch Out

Kate Beck, you already punched out today!

Then click on the 'View timecards' link on the menu to see the list of timecards. This time, you see the punch out times and hours.

**2023-Apr-09**

Emp #	Name	Time in	Time out	Hours
1016	Who2 Ever	2023-Apr-09 14:42:10	null	0
1015	Who Ever	2023-Apr-09 14:42:05	null	0
1014	Scarlett Johann	2023-Apr-09 14:42:01	null	0
1013	Priya Chopra	2023-Apr-09 14:41:57	null	0
1012	Nicole Kidman	2023-Apr-09 14:41:53	null	0
1011	Mila Kunis	2023-Apr-09 14:41:48	null	0
1010	Kate Beck	2023-Apr-09 14:41:37	2023-Apr-09 22:47:47	8.1
1009	Jessica Alba	2023-Apr-09 14:41:34	2023-Apr-09 22:47:43	8.1
1008	Jen Law	2023-Apr-09 14:41:31	2023-Apr-09 22:47:40	8.1
1007	Heidi Lamar	2023-Apr-09 14:41:28	2023-Apr-09 22:47:36	8.1
1006	Hailee Stein	2023-Apr-09 14:41:25	2023-Apr-09 22:47:33	8.1
1005	Emmi Clarke	2023-Apr-09 14:41:22	2023-Apr-09 22:47:27	8.1
1004	Emma Watson	2023-Apr-09 14:41:18	2023-Apr-09 22:47:23	8.1
1003	Emma Stone	2023-Apr-09 14:41:15	2023-Apr-09 22:47:20	8.1
1001	Camille Bell	2023-Apr-09 14:41:08	2023-Apr-09 22:47:17	8.1
1002	Emma Rob	2023-Apr-09 14:37:33	2023-Apr-09 22:45:14	8.13

Copyright © The Lorem Ipsum Company 2023

The steps to follow by the employee should be:

1. Punch in even though it is late; this will create the record
2. Then punch out
3. Let the admin know that you forgot to punch in earlier
4. The admin edits the punch-in time

When an employee reports he/she forgot to punch in earlier, the supervisor (who should be one of the administrators) could simply login to edit the record.

So let us mimic the administrator.

Go ahead and log in.

The screenshot shows a web-based application interface. At the top, there is a dark header bar with three menu items: "Home", "Time cards", and "Employees". To the right of these, the word "Login" is displayed. Below the header, the date "2023-Apr-09" is prominently shown. Underneath this date, there is a table listing employee time cards. The columns in the table are "Emp #", "Name", "Time in", "Time out", and "Hours". The data in the table is as follows:

Emp #	Name	Time in	Time out	Hours
1016	Who2 Ever	2023-Apr-09 14:42:10	null	0
1015	Who Ever	2023-Apr-09 14:42:05	null	0
1014	Scarlett Johann	2023-Apr-09 14:42:01	null	0
1013	Priya Chopra	2023-Apr-09 14:41:57	null	0
1012	Nicole Kidman	2023-Apr-09 14:41:53	null	0
1011	Mila Kunis	2023-Apr-09 14:41:48	null	0
1010	Kate Beck	2023-Apr-09 14:41:37	2023-Apr-09 22:47:47	8.1

A modal dialog box titled "Login" is overlaid on the page. It contains two input fields: one for email ("admin1@lorem.com") and one for password ("....."). Below the password field are two buttons: "Clear" and "Login". At the bottom of the dialog is a link labeled "Cancel".

Remember that 'secret' is the password. After a successful login, the 'Login' item on the menu becomes 'Logout'

[Home](#)[Time cards](#)[Employees](#)[Logout](#)

## **LOREM IPSUM COMPANY TIMEKEEPING SYSTEM**

[Punch In](#)[Punch Out](#)

Then click on the ‘View timecards’ link on the menu to see the list of timecards. This time, the ‘edit’ and ‘delete’ icons appear.

## All time cards

Emp #	Name	Time in	Time out	Hours	Action
1016	Who2 Ever	2023-Apr-09 14:42:10	null	0	 
1015	Who Ever	2023-Apr-09 14:42:05	null	0	 
1014	Scarlett Johann	2023-Apr-09 14:42:01	null	0	 
1013	Priya Chopra	2023-Apr-09 14:41:57	null	0	 
1012	Nicole Kidman	2023-Apr-09 14:41:53	null	0	 
1011	Mila Kunis	2023-Apr-09 14:41:48	null	0	 
1010	Kate Beck	2023-Apr-09 14:41:37	2023-Apr-09 22:47:47	8.1	 
1009	Jessica Alba	2023-Apr-09 14:41:34	2023-Apr-09 22:47:43	8.1	 
1008	Jen Law	2023-Apr-09 14:41:31	2023-Apr-09 22:47:40	8.1	 
1007	Heidi Lamar	2023-Apr-09 14:41:28	2023-Apr-09 22:47:36	8.1	 
1006	Hailee Stein	2023-Apr-09 14:41:25	2023-Apr-09 22:47:33	8.1	 
1005	Emmi Clarke	2023-Apr-09 14:41:22	2023-Apr-09 22:47:27	8.1	 
1004	Emma Watson	2023-Apr-09 14:41:18	2023-Apr-09 22:47:23	8.1	 
1003	Emma Stone	2023-Apr-09 14:41:15	2023-Apr-09 22:47:20	8.1	 
1001	Camille Bell	2023-Apr-09 14:41:08	2023-Apr-09 22:47:17	8.1	 
1002	Emma Rob	2023-Apr-09 14:37:33	2023-Apr-09 22:45:14	8.13	 



Because you have logged in as an administrator, the ‘edit’ and ‘delete’ buttons appear.

## Editing a timecard entry

But then, when you click on one of the pencil icons to edit a record, you get this error message:

```
404 - Not Found

Not Found

Internal error information:
No routes match path '/edit_timecard?id=4&x=6&y=5'
```

because we still haven't written the editing part yet.

Open source\cardcontrol.d and append this code:

```
void getEditTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getTimecard(id);
    Employee e = empModel.getEmployee(t.empid);
    bool loggedIn = m_user.loggedIn;
    render!("cardedit.dt", t, e, error, loggedIn);
}

@errorDisplay!getEditTimecard
void postEditTimecard(Timecard t)
{
    cardModel.editTimecard(t);
    redirect("timecards");
}
```

This code is calling the TimecardModel methods getTimecard() and editTimecard() as well as rendering the cardedit.dt template, so let's write them.

Append this code to source\cardmodel.d:

```
Timecard getTimecard(int id)
{
    string sql = "select * from timecards where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Row[] rows = conn.query(pstmt).array;
    Timecard c;
    if(rows.length > 0)
    {
        Row row = rows[0];
        c.id = to!int(to!string(row[0]));
        c.empid = to!string(row[1]);
        c.fullname = to!string(row[2]);
        c.timein = to!string(row[3]);
        c.timeout = to!string(row[4]);
        c.hours = to!float(to!string(row[5]));
    }
    return c;
}

void editTimecard(Timecard t)
{
    Prepared pstmt;
    if(to!string(t.timeout) == "null")
    {
        string sql =
            "update timecards set
```

```

        timein=str_to_date(?,'%Y-%b-%d %H:%i:%S'),
        timeout=null
        where id=?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setArgs(t.timein, t.id);
    }
else
{
    string sql =
        "update timecards set
        timein=str_to_date(?,'%Y-%b-%d %H:%i:%S'),
        timeout=str_to_date(?,'%Y-%b-%d %H:%i:%S'),
        hours=round((time_to_sec(timeout)-time_to_sec(timein))/3600,2)
        where id=?";
    pstmt = conn.prepareStatement(sql);
    pstmt.setArgs(t.timein, t.timeout, t.id);
}
conn.exec(pstmt);
}

```

And here is views\cardedit.dt:

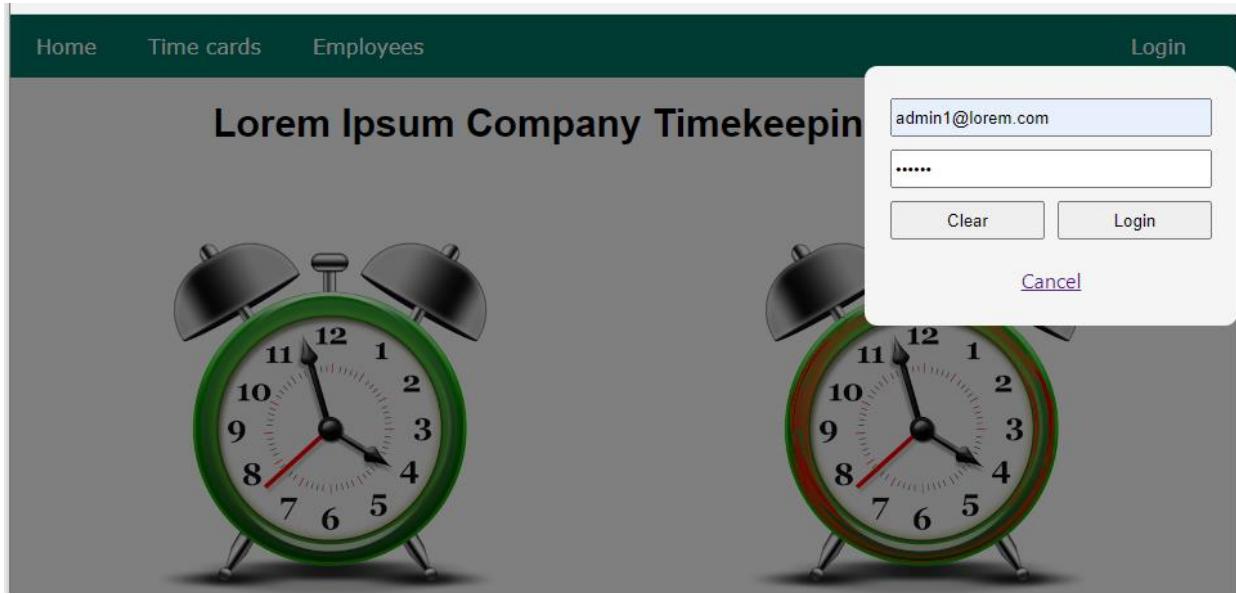
```

extends layout
block maincontent
div.form-grid-wrapper
    h2 Edit time card entry
    -if(error)
        div.error-div
            span.error-message #{error}
form.form-grid(method="post", action="edit_timecard")
    label.form-grid-label Employee :
    label.form-grid-text #{t.empid} - #{t.fullname}

```

```
label.form-grid-label Time in :  
input.form-grid-input(type="text", name="t_timein", value="#{t.timein}")  
label.form-grid-label Time out :  
input.form-grid-input(type="text", name="t_timeout", value="#{t.timeout}")  
div  
input(type="hidden", name="t_id", value="#{t.id}")  
input(type="hidden", name="t.empid", value="#{t.empid}")  
input(type="hidden", name="t_fullname", value="#{t.fullname}")  
input(type="hidden", name="t_hours", value="#{t.hours}")  
div  
div  
a(href="timecards")  
    button.form-grid-button(type="button") Cancel  
div  
    input.form-grid-button(type="submit", value="Submit")
```

Now we can compile, run and refresh the browser. Go back to the home page to refresh the browser. Try to log in again.



Then try to view the timecards again

A screenshot of the application showing the "View timecards" option selected in the navigation menu. The main content area displays the title "Lorem Ipsum Company Timekeeping System".

Home	Time cards	Employees		Logout
1005	Emmi Clarke	2023-Apr-10 09:18:46	null	0
1001	Camille Bell	2023-Apr-10 09:18:35	null	0
1003	Emma Stone	2023-Apr-10 09:18:32	null	0
1004	Emma Watson	2023-Apr-10 09:18:29	null	0
1002	Emma Rob	2023-Apr-10 09:18:26	null	0
1009	Jessica Alba	2023-Apr-10 09:18:24	null	0
1016	Who2 Ever	2023-Apr-09 14:42:10	null	0
1015	Who Ever	2023-Apr-09 14:42:05	null	0
1014	Scarlett Johann	2023-Apr-09 14:42:01	null	0
1013	Priya Chopra	2023-Apr-09 14:41:57	null	0
1012	Nicole Kidman	2023-Apr-09 14:41:53	null	0
1011	Mila Kunis	2023-Apr-09 14:41:48	null	0
1010	Kate Beck	2023-Apr-09 14:41:37	2023-Apr-09 22:47:47	8.1
1009	Jessica Alba	2023-Apr-09 14:41:34	2023-Apr-09 22:47:43	8.1
1008	Jen Law	2023-Apr-09 14:41:31	2023-Apr-09 22:47:40	8.1
1007	Heidi Lamar	2023-Apr-09 14:41:28	2023-Apr-09 22:47:36	8.1
1006	Hailee Stein	2023-Apr-09 14:41:25	2023-Apr-09 22:47:33	8.1
1005	Emmi Clarke	2023-Apr-09 14:41:22	2023-Apr-09 22:47:27	8.1

Click on one of the pencil icons. This time it will open the timecard-editing page.

Home	Time cards	Employees		Logout
<b>Edit time card entry</b>				
Employee : 1011 - Mila Kunis				
Time in : <input type="text" value="2023-Apr-09 14:41:48"/>				
Time out : <input type="text" value="null"/>				
<input type="button" value="Cancel"/>		<input type="button" value="Submit"/>		

Now you can edit either the time-in or the time-out or both, as long as you follow the format of the date and time, including the space in between.

So here is the full source\cardcontrol.d so far:

```
module cardcontrol;

import vibe.vibe;
import basecontrol;
import cardmodel;
import sheetmodel;
import empmodel;

class TimecardController : BaseController
{
    string success = null;

    void getTimeIn(string _error = null)
    {
        Employee[] emps = empModel.getEmployees();
        bool loggedIn = m_user.loggedIn;
        string error = _error;
        render!("timein.dt", emps, loggedIn, error, success);
    }

    @errorDisplay!getTimeIn
    void postTimeIn(string empidname)
    {
        string empid = empidname[0..4];
        string fullname = empidname[5..$];
        //the record should not exist, otherwise employee already punched in earlier
        enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in today!");
        bool timedin = cardModel.timeIn(empid, fullname);
        if(timedin) success = fullname ~ " punched in successfully.";
        else success = null;
    }
}
```

```

    redirect("time_in");
}

void getTimecards(string _error = null)
{
    string error = _error;
    string today = getToday;
    bool loggedIn = m_user.loggedIn;
    Timecard[] cards = cardModel.getTimecards(loggedIn);
    render!("cardlist.dt", cards, today, error, loggedIn);
}

void getTimeOut(string _error = null)
{
    Employee[] emps = empModel.getEmployees;
    bool loggedIn = m_user.loggedIn;
    string error = _error;
    render!("timeout.dt", emps, loggedIn, error, success);
}

@errorDisplay!getTimeOut
void postTimeOut(string empidname)
{
    string empid = empidname[0..4];
    string fullname = empidname[5..$];
    //the record should exist, otherwise employee did not punch in today
    enforce(!cardModel.noTimeIn(empid), fullname ~ ", you did not punch in today.");
    //the timeout field should be null, otherwise employee already punched out
    enforce(cardModel.noTimeOut(empid), fullname ~ ", you already punched out today!");
    bool timedout = cardModel.timeOut(empid);
    if(timedout) success = fullname ~ " punched out successfully.";
    else success = null;
}

```

```

        redirect("time_out");
    }

void getEditTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getTimecard(id);
    Employee e = empModel.getEmployee(t.empid);
    bool loggedIn = m_user.loggedIn;
    render!("cardedit.dt", t, e, error, loggedIn);
}

@errorDisplay!getEditTimecard
void postEditTimecard(Timecard t)
{
    cardModel.editTimecard(t);
    redirect("timecards");
}
}

```

And here is the full source\cardmodel.d so far:

```

module cardmodel;

import mysql;
import std.array;
import std.conv;
import basemodel;

struct Timecard
{
    int id;

```

```
    string empid;
    string fullname;
    string timein;
    string timeout;
    float hours;
}

class TimecardModel : BaseModel
{
    void createTable()
    {
        string sql = "drop table if exists timecards";
        conn.exec(sql);
        sql =
            "create table timecards
            (
                id int auto_increment primary key,
                empid char(4) not null references employees(empid),
                fullname varchar(50) not null,
                timein datetime not null default current_timestamp,
                timeout datetime,
                hours float default 0.0
            )";
        conn.exec(sql);
    }

    bool timeIn(string empid, string fullname)
    {
        string sql = "insert into timecards(empid, fullname, timein) values(?, ?, now())";
        Prepared pstmt = conn.prepare(sql);
        pstmt.setArgs(empid, fullname);
        if(conn.exec(pstmt) > 0) return true;
    }
}
```

```
        return false;
    }

    bool noTimeIn(string empid)
    {
        string sql = "select * from timecards where empid=? and date(timein)=?";
        Prepared pstmt = conn.prepare(sql);
        string today = getToday();
        pstmt.setArgs(empid, today);
        Row[] rows = conn.query(pstmt).array;
        if(rows.length == 0) return true;
        return false;
    }

    string getToday()
    {
        Row[] rows = conn.query("select curdate() + 0").array;
        return toString(rows[0][0]);
    }

    Timecard[] getTimecards(bool loggedIn)
    {
        Timecard[] cards;
        string sql;
        if(loggedIn)
            sql = "select * from timecards order by timein desc";
        else
            sql = "select * from timecards where date(timein) = curdate() order by timein desc";
        Row[] rows = conn.query(sql).array;
        if(rows.length != 0)
        {
            foreach(row; rows)
```

```
{  
    Timecard c;  
    c.id = to!int(to!string(row[0]));  
    c.empid = to!string(row[1]);  
    c.fullname = to!string(row[2]);  
    c.timein = to!string(row[3]);  
    c.timeout = to!string(row[4]);  
    c.hours = to!float(to!string(row[5]));  
    cards ~= c;  
}  
}  
return cards;  
}  
  
bool noTimeOut(string empid)  
{  
    string sql = "select * from timecards where timeout is null and empid=? and date(timein)=?";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs(empid, getToday);  
    Row[] rows = conn.query(pstmt).array;  
    if(rows.length != 0) return true;  
    return false;  
}  
  
bool timeOut(string empid)  
{  
    // we want to include the fraction of the hour  
    // so we use the MySQL function time_to_sec()  
    // 3600 is seconds in an hour  
    // but we round out the result to just two decimals  
    string sql =  
        "update timecards set timeout=now(),
```

```
hours=round((time_to_sec(now())-time_to_sec(timein))/3600,2)
where empid=? and date(timein)=curdate()");
Prepared pstmt = conn.prepare(sql);
pstmt.setArgs(empid);
if(conn.exec(pstmt) > 0) return true;
return false;
}

Timecard getTimecard(int id)
{
    string sql = "select * from timecards where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Row[] rows = conn.query(pstmt).array;
    Timecard c;
    if(rows.length > 0)
    {
        Row row = rows[0];
        c.id = to!int(to!string(row[0]));
        c.empid = to!string(row[1]);
        c.fullname = to!string(row[2]);
        c.timein = to!string(row[3]);
        c.timeout = to!string(row[4]);
        c.hours = to!float(to!string(row[5]));
    }
    return c;
}

void editTimecard(Timecard t)
{
    Prepared pstmt;
    if(to!string(t.timeout) == "null")
```

```
{  
    string sql =  
        "update timecards set  
        timein=str_to_date(?, '%Y-%b-%d %H:%i:%S'),  
        timeout=null  
        where id=?";  
    pstmt = conn.prepareStatement(sql);  
    pstmt.setArgs(t.timein, t.id);  
}  
else  
{  
    string sql =  
        "update timecards set  
        timein=str_to_date(?, '%Y-%b-%d %H:%i:%S'),  
        timeout=str_to_date(?, '%Y-%b-%d %H:%i:%S'),  
        hours=round((time_to_sec(timeout)-time_to_sec(timein))/3600,2)  
        where id=?";  
    pstmt = conn.prepareStatement(sql);  
    pstmt.setArgs(t.timein, t.timeout, t.id);  
}  
conn.exec(pstmt);  
}  
}
```

## Deleting a timecard

As we have done with the employee records earlier, when the user clicks on the delete button (the trash image), the record should be shown on another page and ask the user for confirmation. That is what we will do here.

Append this code to source\cardcontrol.d:

```
void getDeleteTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getTimecard(id);
    Employee e = empModel.getEmployee(t.empid);
    bool loggedIn = m_user.loggedIn;
    render!("carddelete.dt", t, e, error, loggedIn);
}

@errorDisplay!getDeleteTimecard
void postDeleteTimecard(int id)
{
    cardModel.deleteTimecard(id);
    redirect("timecards");
}
```

The getDeleteTimecard() is rendering the carddelete.dt, so let's create it.

Here is views\carddelete.dt:

```
extends layout
block maincontent
    div.form-grid-wrapper
        h2 Delete this time card entry?
        form.form-grid(method="post", action="delete_timecard")
```

```
label.form-grid-label Employee :  
label.form-grid-text #{e.empid} - #{e.fname} #{e.lname}  
label.form-grid-label Punched in :  
label.form-grid-text #{t.timein}  
label.form-grid-label Punched out :  
label.form-grid-text #{t.timeout}  
input(type="hidden", name="id", value="#{t.id}")  
div  
    a(href="timecards")  
        button.form-grid-button(type="button") Cancel  
div  
    input.form-grid-button(type="submit", value="Delete")
```

The postDeleteTimecard() method is calling the cardModel.deleteTimecard(), so let's write it.

Append this code to source\cardmodel.d:

```
void deleteTimecard(int id)  
{  
    string sql = "delete from timecards where id=?";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs(id);  
    conn.exec(pstmt);  
}
```

Now compile, run and go back to the home page to refresh the browser. Open the list of timecards.

**2023-Apr-10**

Emp #	Name	Time in	Time out	Hours
1007	Heidi Lamar	2023-Apr-10 11:16:23	null	0
1014	Scarlett Johann	2023-Apr-10 09:28:28	null	0
1008	Jen Law	2023-Apr-10 09:28:18	null	0
1013	Priya Chopra	2023-Apr-10 09:19:04	null	0
1012	Nicole Kidman	2023-Apr-10 09:19:00	null	0
1016	Who2 Ever	2023-Apr-10 09:18:56	null	0
1006	Hailee Stein	2023-Apr-10 09:18:53	null	0
1005	Emmi Clarke	2023-Apr-10 09:18:46	null	0
1001	Camille Bell	2023-Apr-10 09:18:35	null	0
1003	Emma Stone	2023-Apr-10 09:18:32	null	0
1004	Emma Watson	2023-Apr-10 09:18:29	null	0
1002	Emma Rob	2023-Apr-10 09:18:26	null	0
1009	Jessica Alba	2023-Apr-10 09:18:24	null	0

There are no ‘edit’ and ‘delete’ icons. You have to log in again because this is a new version of the app and the session values were erased when you recompiled and ran again.

**2023-Apr-10**

Emp #	Name	Time in	Time out	Hours
1007	Heidi Lamar	2023-Apr-10 11:16:23	null	0
1014	Scarlett Johann	2023-Apr-10 09:28:28	null	0
1008	Jen Law	2023-Apr-10 09:28:18	null	0
1013	Priya Chopra	2023-Apr-10 09:19:04	null	0
1012	Nicole Kidman	2023-Apr-10 09:19:00	null	0
1016	Who2 Ever	2023-Apr-10 09:18:56	null	0
1006	Hailee Stein	2023-Apr-10 09:18:53	null	0
1005	Emmi Clarke	2023-Apr-10 09:18:46	null	0

admin1@lorem.com

.....

[Cancel](#)

Open the list of timecards again. This time you will see the ‘edit’ and ‘delete’ icons.

## All time cards

Emp #	Name	Time in	Time out	Hours	Action
1007	Heidi Lamar	2023-Apr-10 11:16:23	null	0	
1014	Scarlett Johann	2023-Apr-10 09:28:28	null	0	
1008	Jen Law	2023-Apr-10 09:28:18	null	0	
1013	Priya Chopra	2023-Apr-10 09:19:04	null	0	
1012	Nicole Kidman	2023-Apr-10 09:19:00	null	0	
1016	Who2 Ever	2023-Apr-10 09:18:56	null	0	
1006	Hailee Stein	2023-Apr-10 09:18:53	null	0	
1005	Emmi Clarke	2023-Apr-10 09:18:46	null	0	
1001	Camille Bell	2023-Apr-10 09:18:35	null	0	
1003	Emma Stone	2023-Apr-10 09:18:32	null	0	
1004	Emma Watson	2023-Apr-10 09:18:29	null	0	
1002	Emma Rob	2023-Apr-10 09:18:26	null	0	
1009	Jessica Alba	2023-Apr-10 09:18:24	null	0	
1016	Who2 Ever	2023-Apr-09 14:42:10	null	0	
1015	Who Ever	2023-Apr-09 14:42:05	null	0	
1014	Scarlett Johann	2023-Apr-09 14:42:01	null	0	
1013	Priya Chopra	2023-Apr-09 14:41:57	null	0	
1012	Nicole Kidman	2023-Apr-09 14:41:53	null	0	

Click on one of the trash icons to open the record in a separate page.

### Delete this time card entry?

Employee : 1002 - Emma Rob

Punched in : 2023-Apr-10 09:18:26

Punched out : null

[Cancel](#)[Delete](#)

And if you press ‘Delete’, the record is gone from the list.

All time cards						Logout
Emp #	Name	Time in	Time out	Hours	Action	
1007	Heidi Lamar	2023-Apr-10 11:16:23	null	0	 	
1014	Scarlett Johann	2023-Apr-10 09:28:28	null	0	 	
1008	Jen Law	2023-Apr-10 09:28:18	null	0	 	
1013	Priya Chopra	2023-Apr-10 09:19:04	null	0	 	
1012	Nicole Kidman	2023-Apr-10 09:19:00	null	0	 	
1016	Who2 Ever	2023-Apr-10 09:18:56	null	0	 	
1006	Hailee Stein	2023-Apr-10 09:18:53	null	0	 	
1005	Emmi Clarke	2023-Apr-10 09:18:46	null	0	 	
1001	Camille Bell	2023-Apr-10 09:18:35	null	0	 	
1003	Emma Stone	2023-Apr-10 09:18:32	null	0	 	
1004	Emma Watson	2023-Apr-10 09:18:29	null	0	 	
1009	Jessica Alba	2023-Apr-10 09:18:24	null	0	 	
1015	Who Ever	2023-Apr-09 14:42:05	null	0	 	
1014	Scarlett Johann	2023-Apr-09 14:42:01	null	0	 	
1013	Priya Chopra	2023-Apr-09 14:41:57	null	0	 	
1012	Nicole Kidman	2023-Apr-09 14:41:53	null	0	 	
1011	Mila Kunis	2023-Apr-09 14:41:48	null	0	 	
1010	Kate Beck	2023-Apr-09 14:41:37	2023-Apr-09 22:47:47	8.1	 	
1009	Jessica Alba	2023-Apr-09 14:41:34	2023-Apr-09 22:47:43	8.1	 	

Copyright © The Lorem Ipsum Company 2023

So here is the full source\cardcontrol.d:

```
module cardcontrol;

import vibe.vibe;
import basecontrol;
import cardmodel;
import sheetmodel;
import empmodel;
```

```
class TimecardController : BaseController
{
    string success = null;

    void getTimeIn(string _error = null)
    {
        Employee[] emps = empModel.getEmployees();
        bool loggedIn = m_user.loggedIn;
        string error = _error;
        render!("timein.dt", emps, loggedIn, error, success);
    }

    @errorDisplay!getTimeIn
    void postTimeIn(string empidname)
    {
        string empid = empidname[0..4];
        string fullname = empidname[5..$];
        //the record should not exist, otherwise employee already punched in earlier
        enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in today!");
        bool timedin = cardModel.timeIn(empid, fullname);
        if(timedin) success = fullname ~ " punched in successfully.";
        else success = null;
        redirect("time_in");
    }

    void getTimecards(string _error = null)
    {
        string error = _error;
        string today = getToday;
        bool loggedIn = m_user.loggedIn;
        Timecard[] cards = cardModel.getTimecards(loggedIn);
```

```

    render!("cardlist.dt", cards, today, error, loggedIn);
}

void getTimeOut(string _error = null)
{
    Employee[] emps = empModel.getEmployees;
    bool loggedIn = m_user.loggedIn;
    string error = _error;
    render!("timeout.dt", emps, loggedIn, error, success);
}

@errorDisplay!getTimeOut
void postTimeOut(string empidname)
{
    string empid = empidname[0..4];
    string fullname = empidname[5..$];
    //the record should exist, otherwise employee did not punch in today
    enforce(!cardModel.noTimeIn(empid), fullname ~ ", you did not punch in today.");
    //the timeout field should be null, otherwise employee already punched out
    enforce(cardModel.noTimeOut(empid), fullname ~ ", you already punched out today!");
    bool timedout = cardModel.timeOut(empid);
    if(timedout) success = fullname ~ " punched out successfully.";
    else success = null;
    redirect("time_out");
}

void getEditTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getTimecard(id);
    Employee e = empModel.getEmployee(t.empid);
    bool loggedIn = m_user.loggedIn;
}

```

```

    render!("cardedit.dt", t, e, error, loggedIn);
}

@errorDisplay!getEditTimecard
void postEditTimecard(Timecard t)
{
    cardModel.editTimecard(t);
    redirect("timecards");
}

void getDeleteTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getTimecard(id);
    Employee e = empModel.getEmployee(t.empid);
    bool loggedIn = m_user.loggedIn;
    render!("carddelete.dt", t, e, error, loggedIn);
}

@errorDisplay!getDeleteTimecard
void postDeleteTimecard(int id)
{
    cardModel.deleteTimecard(id);
    redirect("timecards");
}
}

```

And here is the full source\cardmodel.d:

```

module cardmodel;

import mysql;

```

```
import std.array;
import std.conv;
import basemodel;

struct Timecard
{
    int id;
    string empid;
    string fullname;
    string timein;
    string timeout;
    float hours;
}

class TimecardModel : BaseModel
{
    void createTable()
    {
        string sql = "drop table if exists timecards";
        conn.exec(sql);
        sql =
            "create table timecards
            (
                id int auto_increment primary key,
                empid char(4) not null references employees(empid),
                fullname varchar(50) not null,
                timein datetime not null default current_timestamp,
                timeout datetime,
                hours float default 0.0
            )";
        conn.exec(sql);
    }
}
```

```
bool timeIn(string empid, string fullname)
{
    string sql = "insert into timecards(empid, fullname, timein) values(?, ?, now())";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(empid, fullname);
    if(conn.exec(pstmt) > 0) return true;
    return false;
}

bool noTimeIn(string empid)
{
    string sql = "select * from timecards where empid=? and date(timein)=?";
    Prepared pstmt = conn.prepare(sql);
    string today = getToday();
    pstmt.setArgs(empid, today);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return true;
    return false;
}

string getToday()
{
    Row[] rows = conn.query("select curdate() + 0").array;
    return to!string(rows[0][0]);
}

Timecard[] getTimecards(bool loggedIn)
{
    Timecard[] cards;
    string sql;
    if(loggedIn)
```

```
    sql = "select * from timecards order by timein desc";
else
    sql = "select * from timecards where date(timein) = curdate() order by timein desc";
Row[] rows = conn.query(sql).array;
if(rows.length != 0)
{
    foreach(row; rows)
    {
        Timecard c;
        c.id = to!int(to!string(row[0]));
        c.empid = to!string(row[1]);
        c.fullname = to!string(row[2]);
        c.timein = to!string(row[3]);
        c.timeout = to!string(row[4]);
        c.hours = to!float(to!string(row[5]));
        cards ~= c;
    }
}
return cards;
}

bool noTimeOut(string empid)
{
    string sql = "select * from timecards where timeout is null and empid=? and date(timein)=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(empid, getToday);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length != 0) return true;
    return false;
}

bool timeOut(string empid)
```

```
{  
    // we want to include the fraction of the hour  
    // so we use the MySQL function time_to_sec()  
    // 3600 is seconds in an hour  
    // but we round the result to just two decimals  
    string sql =  
        "update timecards set timeout=now(),  
            hours=round((time_to_sec(now())-time_to_sec(timein))/3600,2)  
            where empid=? and date(timein)=curdate()";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs(empid);  
    if(conn.exec(pstmt) > 0) return true;  
    return false;  
}  
  
Timecard getTimecard(int id)  
{  
    string sql = "select * from timecards where id=?";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs(id);  
    Row[] rows = conn.query(pstmt).array;  
    Timecard c;  
    if(rows.length > 0)  
    {  
        Row row = rows[0];  
        c.id = to!int(to!string(row[0]));  
        c.empid = to!string(row[1]);  
        c.fullname = to!string(row[2]);  
        c.timein = to!string(row[3]);  
        c.timeout = to!string(row[4]);  
        c.hours = to!float(to!string(row[5]));  
    }  
}
```

```
        return c;
    }

    void editTimecard(Timecard t)
    {
        Prepared pstmt;
        if(to!string(t.timeout) == "null")
        {
            string sql =
                "update timecards set
                 timein=str_to_date(?,'%Y-%b-%d %H:%i:%S'),
                 timeout=null
                 where id=?";
            pstmt = conn.prepare(sql);
            pstmt.setArgs(t.timein, t.id);
        }
        else
        {
            string sql =
                "update timecards set
                 timein=str_to_date(?,'%Y-%b-%d %H:%i:%S'),
                 timeout=str_to_date(?,'%Y-%b-%d %H:%i:%S'),
                 hours=round((time_to_sec(timeout)-time_to_sec(timein))/3600,2)
                 where id=?";
            pstmt = conn.prepare(sql);
            pstmt.setArgs(t.timein, t.timeout, t.id);
        }
        conn.exec(pstmt);
    }

    void deleteTimecard(int id)
    {
```

```
string sql = "delete from timecards where id=?";
Prepared pstmt = conn.prepare(sql);
pstmt.setArgs(id);
conn.exec(pstmt);
}
```

## Generating the timesheet

The timesheet computes the hours worked by each employee at end of wage period, in this case weekly. It looks up the data from the timecards table and generates the timesheet based on that data. The timesheet is overwritten every time it is created but the timecards remain so they are permanent records.

So let's edit source\sheetsmodel.d

```
module sheetmodel;

import mysql;
import std.array;
import std.conv;
import basemodel;

struct Timesheet
{
    int id;
    int period;
    string empid;
    string fullname;
    float hours;
}

class TimesheetModel : BaseModel
{
    void createTable()
    {
        string sql = "drop table if exists timesheets";
        conn.exec(sql);
        sql =
            "create table timesheets
```

```
(  
    id int auto_increment primary key,  
    period int not null,  
    empid char(4) not null,  
    fullname varchar(50) not null,  
    hours float  
)";  
conn.exec(sql);  
}  
  
void createTimesheet(int week)  
{  
    string sql = "delete from timesheets";  
    conn.exec(sql);  
    sql =  
        "insert into timesheets(period, hours, empid, fullname)  
            (select week(timein) as period, sum(hours), empid, fullname from timecards  
             where week(timein)=? group by period, empid, fullname)";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs(week);  
    conn.exec(pstmt);  
}  
  
Timesheet[] getTimesheets()  
{  
    string sql = "select * from timesheets order by empid";  
    Row[] rows = conn.query(sql).array;  
    Timesheet[] sheets;  
    foreach (row; rows)  
    {  
        Timesheet t;  
        t.id = to!int(to!string(row[0]));
```

```

        t.period = to!int(to!string(row[1]));
        t.empid = to!string(row[2]);
        t.fullname = to!string(row[3]);
        t.hours = to!double(to!string(row[4]));
        sheets ~= t;
    }
    return sheets;
}

int getTheWeek()
{
    Row[] rows = conn.query("select period from timesheets limit 1").array;
    if(rows.length == 0) return 0;
    return to!int(to!string(rows[0][0]));
}
}

```

We added the method `createTimesheet()` to the model which is the code that actually creates the timesheet, as well as the `getTheWeek()` method which will be needed by the `TimesheetController`.

Here is source\sheetcontrol.d:

```

module sheetcontrol;

import vibe.vibe;
import basecontrol;
import empmodel;
import cardmodel;
import sheetmodel;

class TimesheetController : BaseController
{

```

```

@auth
void getCreateTimesheet(string _authUser, string _error = null)
{
    string error = _error;
    int[] weeks = cardModel.getTheWeeks();
    bool loggedIn = m_user.loggedIn;
    render!("sheetcreate.dt", error, loggedIn, weeks);
}

@errorDisplay!getCreateTimesheet
void postCreateTimesheet(int week)
{
    sheetModel.createTimesheet(week);
    redirect("timesheets");
}

void getTimesheets()
{
    Timesheet[] sheets = sheetModel.getTimesheets();
    bool loggedIn = m_user.loggedIn;
    int week = sheetModel.getTheWeek();
    render!("sheetlist.dt", sheets, loggedIn, week);
}
}

```

The getCreateTimesheet() method is calling cardModel.getTheWeeks() before rendering the form sheetcreate.dt, so let's create them.

Append this code to source\cardmodel.dt:

```

int[] getTheWeeks()
{

```

```

    string sql = "select distinct week(timein) as weeks from timecards order by weeks";
    Row[] rows = conn.query(sql).array;
    int[] weeks;
    foreach (row; rows) weeks ~= to!int(to!string(row[0]));
    return weeks;
}

```

And here is views\sheetcreate.dt:

```

extends layout
block maincontent
    include cssformgrid.dt
    div.form-grid-wrapper
        -if(error)
            div.error-div
                span.error-message #{error}
        form.form-grid(method="post", action="create_timesheet")
            div
                label.form-grid-label For week number
                select.form-grid-input(name="week")
                    -foreach(w; weeks)
                        option(value="#{w}") #{w}
            div
                input.form-grid-button(type="submit", value="Generate timesheet")

```

The sheetControl.getTimesheets() does the actual display of the timesheet by calling sheetModel.getTimesheets() and sheetModel.getTheWeek(), which we have already written, then rendering sheetlist.dt, which we haven't created yet, so let's create it.

Here is views\sheetslist.dt:

```

extends layout

```

```
block maincontent
  h2 Timesheet for week #{week}
  div.table-wrapper
    table
      tr
        th Emp #
        th Name
        th Hours
      -foreach(s; sheets)
        tr
          td #{s.empid}
          td #{s.fullname}
          td.right-align #{s.hours}
```

Now compile and run the app.

```
C:\vibeprojects\loremtime>dub
  Starting Performing "debug" build using C:\D\dmd2\windows\bin64\dmd.exe for x86_64.
...
  Running loremtime.exe
[main(---) INF] Listening for requests on http://[::1]:8080/
[main(---) INF] Listening for requests on http://127.0.0.1:8080/
```

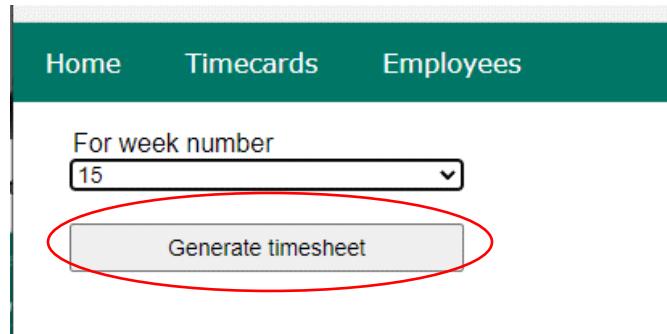
Try to log in again.



Then click on the ‘Create timesheet’ link.

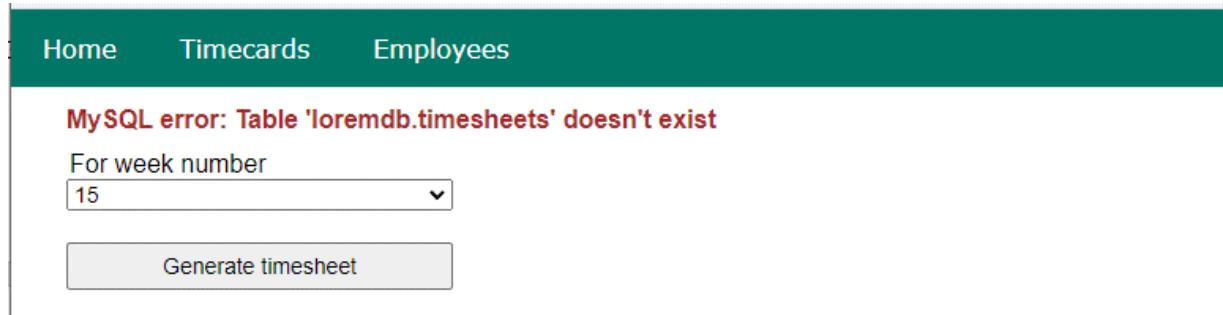


And this screen will show. Choose a week from the drop-down list.



After choosing the week, click on the ‘Generate timesheet’ button.

And you get this error:



So let's create the timesheets table.

Edit source\sheetcontrol.d and add this code right after the class declaration:

```
class TimesheetController : BaseController
{
    this()
    {
        sheetModel.createTable;
    }

    @auth
    void getCreateTimesheet(string _authUser, string _error = null)
    {
```

Then compile and run.

After compiling and running, delete the code that you have just added!

```
---this()
---{
```

```
sheetModel.createTable;  
})
```

Then compile again.

Click on the ‘Create timesheet’ link. You are redirected to the login dialogue.

Login again, then click on the ‘Create timesheet’ button again.

After selecting the week and clicking ‘Generate timesheet’, you should see this:

Home   Timecards   Employees   Logout

### Timesheet for week 15

Emp #	Name	Hours
1001	Camille Bell	8.1
1002	Emma Rob	8.13
1003	Emma Stone	8.1
1004	Emma Watson	8.1
1005	Emmi Clarke	8.1
1006	Hailee Stein	8.1
1007	Heidi Lamar	8.1
1008	Jen Law	8.1
1009	Jessica Alba	8.1
1010	Kate Beck	8.1
1011	Mila Kunis	0
1012	Nicole Kidman	0
1013	Priya Chopra	0
1014	Scarlett Johann	0
1015	Who Ever	0
1016	Who2 Ever	0

Copyright © The Lorem Ipsum Company 2023

You should be able to see the total number of hours the employees worked for the week. As mentioned before, the timesheet is not editable because it simply extracts the data from the timecards table. The timecards table is editable and is a permanent record while the timesheets table is simply generated when needed.

## The end

That's it, we reached the end of this tutorial. I hope you learned something!

If you want to give me some feedback or criticism, you can email me at [karitoy@gmail.com](mailto:karitoy@gmail.com).

Thanks!

The following pages show all the source code.

## The main program

source\app.d:

```
import vibe.vibe;
import homecontrol;
import empcontrol;
import cardcontrol;
import sheetcontrol;

void main()
{
    auto settings = new HTTPServerSettings;
    settings.port = 8080;
    settings.bindAddresses = [":1", "127.0.0.1"];
    settings.sessionStore = new MemorySessionStore;

    auto router = new URLRouter;
    router.get("*", serveStaticFiles("public/"));
    router.registerWebInterface(new HomeController);
    router.registerWebInterface(new EmployeeController);
    router.registerWebInterface(new TimecardController);
    router.registerWebInterface(new TimesheetController);

    auto listener = listenHTTP(settings, router);
    scope (exit) listener.stopListening();

    runApplication();
}
```

## The base controller

source\basecontrol.d:

```
module basecontrol;

import vibe.vibe;
import empmodel;
import cardmodel;
import sheetmodel;
import adminmodel;

struct User
{
    bool loggedIn;
    string email;
}

class BaseController
{
    protected SessionVar!(User, "user") m_user;
    protected enum auth = before!ensureAuth("_authUser");
    protected EmployeeModel empModel;
    protected TimecardModel cardModel;
    protected TimesheetModel sheetModel;
    protected AdminModel adminModel;

    this()
    {
        empModel = new EmployeeModel;
        cardModel = new TimecardModel;
        sheetModel = new TimesheetModel;
```

```
    adminModel = new AdminModel;
}

string ensureAuth(HTTPServerRequest req, HTTPServerResponse res)
{
    if(!m_user.loggedIn) redirect("#login");
    return m_user.email;
}
mixin PrivateAccessProxy;

string getToday()
{
    import std.datetime.date;
    auto time = Clock.currTime;
    return to!string(Date(time.year, time.month, time.day));
}
}
```

## The home controller

source\homecontrol.d:

```
module homecontrol;

import vibe.vibe;
import basecontrol;
import adminmodel;

class HomeController : BaseController
{
    void index(string _error = null)
    {
        string error = _error;
        bool loggedIn = m_user.loggedIn;
        render!("index.dt", error, loggedIn);
    }

    @errorDisplay!index
    void postLogin(string email, string password)
    {
        bool isAdmin = adminModel.getAdmin(email, password);
        enforce(isAdmin, "Email and password combination not found.");
        User user = m_user;
        user.loggedIn = true;
        user.email = email;
        m_user = user;
        redirect("/");
    }

    void getLogout()
```

```
{  
    m_user = User.init;  
    terminateSession;  
    redirect("/");  
}  
}
```

## The employees controller

source\empcontrol.d:

```
module empcontrol;

import vibe.vibe;
import basecontrol;
import empmodel;
import adminmodel;

class EmployeeController : BaseController
{
    @auth
    void getAddEmployee(string _authUser, string _error = null)
    {
        string error = _error;
        bool loggedIn = m_user.loggedIn;
        string[] payrates = empModel.getPayrates();
        render!("empadd.dt", departments, payrates, provinces, error, loggedIn);
    }

    @errorDisplay!getAddEmployee
    void postAddEmployee(Employee e)
    {
        import std.file;
        import std.path;
        import std.algorithm;

        auto pic = "picture" in request.files;
        if(pic !is null)
        {
```

```
        string photopath = "none yet";
        string ext = extension(pic.filename.name);
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];
        if(canFind(exts, ext))
        {
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;
            string dir = "./public/uploads/photos/";
            mkdirRecurse(dir);
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
            try moveFile(pic.tempPath, NativePath(fullpath));
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);
        }
        e.photo = photopath;
    }

    if(e.phone.length == 0) e.phone = "(123) 456 7890";
    if(e.payrate.length == 0) e.payrate = "none yet";
    if(e.postcode.length == 0) e.postcode = "A1A 1A1";
    empModel.addEmployee(e);
    redirect("all_employees");
}

@auth
void getAllEmployees(string _authUser, string _error = null)
{
    string error = _error;
    Employee[] emps = empModel.getEmployees();
    bool loggedIn = m_user.loggedIn;
    render!("emplist.dt", emps, error, loggedIn);
}

@auth
void getEditEmployee(string _authUser, int id, string _error = null)
```

```
{  
    string error = _error;  
    Employee e = empModel.getEmployee(id);  
    bool loggedIn = m_user.loggedIn;  
    string[] payrates = empModel.getPayrates;  
    render!("empedit.dt", e, departments, payrates, provinces, error, loggedIn);  
}  
  
@errorDisplay!getEditEmployee  
void postEditEmployee(Employee e, string prevPassw)  
{  
    import std.file;  
    import std.path;  
    import std.algorithm;  
  
    string photopath = e.photo;  
    auto pic = "picture" in request.files;  
    if(pic !is null)  
    {  
        string ext = extension(pic.filename.name);  
        string[] exts = [".jpg", ".jpeg", ".png", ".gif"];  
        if(canFind(exts, ext))  
        {  
            photopath = "uploads/photos/" ~ e.fname ~ "_" ~ e.lname ~ ext;  
            string dir = "./public/uploads/photos/";  
            mkdirRecurse(dir);  
            string fullpath = dir ~ e.fname ~ "_" ~ e.lname ~ ext;  
            try moveFile(pic.tempPath, NativePath(fullpath));  
            catch (Exception ex) copyFile(pic.tempPath, NativePath(fullpath), true);  
        }  
    }  
    e.photo = photopath;
```

```

        if(e.phone.length == 0) e.phone = "(123) 456 7890";
        if(e.payrate.length == 0) e.payrate = "none yet";
        if(e.postcode.length == 0) e.postcode = "A1A 1A1";
        bool newpass = (prevPassw != e.passw) ? true : false;
        empModel.editEmployee(e, newpass);
        redirect("all_employees");
    }

@auth
void getDeleteEmployee(string _authUser, int id, string _error = null)
{
    string error = _error;
    Employee e = empModel.getEmployee(id);
    bool loggedIn = m_user.loggedIn;
    render!("empdelete.dt", e, error, loggedIn);
}

@errorDisplay!getDeleteEmployee
void postDeleteEmployee(int id)
{
    empModel.deleteEmployee(id);
    redirect("all_employees");
}

@auth
@errorDisplay!getAllEmployees
void postFindEmployee(string _authUser, string fname, string lname)
{
    import std.uni; //so we can use the toUpper() function
    Employee e = empModel.findEmployee(toUpper(fname), toUpper(lname));
    enforce(e != Employee.init, "Cannot find employee " ~ fname ~ " " ~ lname);
    bool loggedIn = m_user.loggedIn;
}

```

```
    render!("empshow.dt", e, loggedIn);
}
```

## The employee model

source\empmodel.d:

```
module empmodel;

import mysql;
import std.conv;
import std.array;
import basemodel;

struct Employee
{
    int id; //row id or record id
    string empid; //employee number
    string email; //email address
    string passw; //password
    string fname; //first name
    string lname; //last name
    string phone; //phone number
    string photo; //ID photo
    string dept; //department
    string payrate; //salary grade
    string street; //street address
    string city; //city name
    string province; //province name
    string postcode; //postal code
}

string[] departments =
[
    "Management",
```

```
"Accounting",
"Production",
"Maintenance",
"Shipping",
"Purchasing",
"IT Services",
"HR Services",
"Marketing"
];

string[][] provinces =
[
    ["AB", "Alberta"],
    ["BC", "British Columbia"],
    ["MB", "Manitoba"],
    ["NB", "New Brunswick"],
    ["NL", "Newfoundland and Labrador"],
    ["NS", "Nova Scotia"],
    ["NT", "Northwest Territories"],
    ["NU", "Nunavut"],
    ["ON", "Ontario"],
    ["PE", "Prince Edward Island"],
    ["QC", "Quebec"],
    ["SK", "Saskatchewan"],
    ["YT", "Yukon Territory"]
];
}

class EmployeeModel : BaseModel
{
    void createTable()
    {
        string sql = "drop table if exists employees";
```

```
conn.exec(sql);
sql =
    "create table employees
    (
        id int auto_increment primary key,
        empid char(4) not null unique,
        email varchar(50) not null,
        passw varchar(255) not null,
        fname varchar(25) not null,
        lname varchar(25) not null,
        phone varchar(15) default '123-456-7890',
        photo varchar(50) default 'none provided',
        dept varchar(30) default 'not assigned yet',
        payrate char(4) default 'B400',
        street varchar(50) default '123 First Street',
        city varchar(30) default 'Toronto',
        province char(2) default 'ON',
        postcode char(7) default 'Z9Z 9Z9',
        created timestamp default current_timestamp,
        updated timestamp on update current_timestamp
    )";
conn.exec(sql);
}

void createPayrates()
{
    int[string] payrates =
    [
        "A100":100, "A200":75, "A300":50, "A400":40,
        "B100":30, "B200":29, "B300":28, "B400":27,
        "C100":26, "C200":25, "C300":24, "C400":23,
        "D100":22, "D200":21, "D300":20, "D400":19,
```

```
"E100":18, "E200":17, "E300":16, "E400":15,
    "F100":14, "F200":13, "F300":12, "F400":11
];
string sql = "drop table if exists payrates";
conn.exec(sql);
sql =
    "create table payrates
(
    id int auto_increment primary key,
    payrate char(4) not null,
    hourly float not null
)";
conn.exec(sql);
foreach(k,v; payrates)
{
    sql =
        "insert into payrates(payrate, hourly)
         values('" ~ k ~ "', " ~ to!string(v) ~ ")";
    conn.exec(sql);
}
}

string[] getPayrates()
{
    string sql = "select payrate from payrates order by payrate";
    Row[] rows = conn.query(sql).array;
    string[] rates;
    foreach(row; rows) rates ~= to!string(row[0]);
    return rates;
}

void addEmployee(Employee e)
```

```
{  
    import vibe.http.auth.digest_auth;  
    e.passw = createDigestPassword(realm, e.email, e.passw);  
    string sql =  
        "insert into employees  
        (  
            empid,  
            email, passw, fname, lname,  
            phone, photo, dept, payrate,  
            street, city, province, postcode  
        )  
        values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";  
    Prepared pstmt = conn.prepare(sql);  
    pstmt.setArgs  
    (  
        to!string(e.empid),  
        to!string(e.email),  
        to!string(e.passw),  
        to!string(e.fname),  
        to!string(e.lname),  
        to!string(e.phone),  
        to!string(e.photo),  
        to!string(e.dept),  
        to!string(e.payrate),  
        to!string(e.street),  
        to!string(e.city),  
        to!string(e.province),  
        to!string(e.postcode)  
    );  
    conn.exec(pstmt);  
}
```

```
Employee[] getEmployees()
{
    Employee[] emps;
    String sql = "select * from employees";
    Row[] rows = conn.query(sql).array();
    if(rows.length == 0) return emps;
    foreach(row; rows) emps ~= prepareEmployee(row);
    return emps;
}

Employee prepareEmployee(Row row)
{
    Employee e;
    e.id = to!int(to!string(row[0]));
    e.empid = to!string(row[1]);
    e.email = to!string(row[2]);
    e.passw = to!string(row[3]);
    e.fname = to!string(row[4]);
    e.lname = to!string(row[5]);
    e.phone = to!string(row[6]);
    e.photo = to!string(row[7]);
    e.deprt = to!string(row[8]);
    e.payrate = to!string(row[9]);
    e.street = to!string(row[10]);
    e.city = to!string(row[11]);
    e.province = to!string(row[12]);
    e.postcode = to!string(row[13]);
    return e;
}

Employee getEmployee(int id)
{
```

```
string sql = "select * from employees where id=?";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setArgs(id);
Employee e;
Row[] rows = conn.query(pstmt).array;
if(rows.length == 0) return e;
return prepareEmployee(rows[0]);
}

Employee getEmployee(string empid)
{
    string sql = "select * from employees where empid=?";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setArgs(empid);
    Employee e;
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return e;
    return prepareEmployee(rows[0]);
}

void editEmployee(Employee e, bool newpass)
{
    if(newpass) // if password was changed, encrypt it
    {
        import vibe.http.auth.digest_auth;
        e.passw = createDigestPassword(realm, e.email, e.passw);
    }
    string sql =
        "update employees set empid=?,
        email=?, passw=?, fname=?, lname=?,
        phone=?, photo=?, deprt=?, payrate=?,
        street=?, city=?, province=?, postcode=?
```

```
    where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs
    (
        to!string(e.empid),
        to!string(e.email),
        to!string(e.passw),
        to!string(e.fname),
        to!string(e.lname),
        to!string(e.phone),
        to!string(e.photo),
        to!string(e.deprt),
        to!string(e.payrate),
        to!string(e.street),
        to!string(e.city),
        to!string(e.province),
        to!string(e.postcode),
        to!int(to!string(e.id))
    );
    conn.exec(pstmt);
}

void deleteEmployee(int id)
{
    string sql = "delete from employees where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    conn.exec(pstmt);
}

Employee findEmployee(string first, string last)
{
```

```
Employee e;
string sql = "select * from employees where upper(fname)=? and upper(lname)=?";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setArgs(first, last);
Row[] rows = conn.query(pstmt).array;
if(rows.length == 0) return e;
return prepareEmployee(rows[0]);
}
```

## The timecard controller

source\cardcontrol.d:

```
module cardcontrol;

import vibe.vibe;
import basecontrol;
import cardmodel;
import sheetmodel;
import empmodel;

class TimecardController : BaseController
{
    string success = null;

    void getTimeIn(string _error = null)
    {
        Employee[] emps = empModel.getEmployees();
        bool loggedIn = m_user.loggedIn;
        string error = _error;
        render!("timein.dt", emps, loggedIn, error, success);
    }

    @errorDisplay!getTimeIn
    void postTimeIn(string empidname)
    {
        string empid = empidname[0..4];
        string fullname = empidname[5..$];
        //the record should not exist, otherwise employee already punched in earlier
        enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in today!");
        bool timedin = cardModel.timeIn(empid, fullname);
    }
}
```

```

    if(timedin) success = fullname ~ " punched in successfully.";
    else success = null;
    redirect("time_in");
}

void getTimecards(string _error = null)
{
    string error = _error;
    string today = getToday;
    bool loggedIn = m_user.loggedIn;
    Timecard[] cards = cardModel.getTimecards(loggedIn);
    render!("cardlist.dt", cards, today, error, loggedIn);
}

void getTimeOut(string _error = null)
{
    Employee[] emps = empModel.getEmployees;
    bool loggedIn = m_user.loggedIn;
    string error = _error;
    render!("timeout.dt", emps, loggedIn, error, success);
}

@errorDisplay!getTimeOut
void postTimeOut(string empidname)
{
    string empid = empidname[0..4];
    string fullname = empidname[5..$];
    //the record should exist, otherwise employee did not punch in today
    enforce(!cardModel.noTimeIn(empid), fullname ~ ", you did not punch in today.");
    //the timeout field should be null, otherwise employee already punched out
    enforce(cardModel.noTimeOut(empid), fullname ~ ", you already punched out today!");
    bool timedout = cardModel.timeOut(empid);
}

```

```
    if(timedout) success = fullname ~ " punched out successfully.";
    else success = null;
    redirect("time_out");
}

void getEditTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getTimecard(id);
    Employee e = empModel.getEmployee(t.empid);
    bool loggedIn = m_user.loggedIn;
    render!("cardedit.dt", t, e, error, loggedIn);
}

@errorDisplay!getEditTimecard
void postEditTimecard(Timecard t)
{
    cardModel.editTimecard(t);
    redirect("timecards");
}

void getDeleteTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getTimecard(id);
    Employee e = empModel.getEmployee(t.empid);
    bool loggedIn = m_user.loggedIn;
    render!("carddelete.dt", t, e, error, loggedIn);
}

@errorDisplay!getDeleteTimecard
void postDeleteTimecard(int id)
```

```
{  
    cardModel.deleteTimecard(id);  
    redirect("timecards");  
}  
}
```

## The timecard model

source\cardmodel.d:

```
module cardmodel;

import mysql;
import std.array;
import std.conv;
import basemodel;

struct Timecard
{
    int id;
    string empid;
    string fullname;
    string timein;
    string timeout;
    float hours;
}

class TimecardModel : BaseModel
{
    void createTable()
    {
        string sql = "drop table if exists timecards";
        conn.exec(sql);
        sql =
            "create table timecards
            (
                id int auto_increment primary key,
                empid char(4) not null references employees(empid),
```

```
    fullname varchar(50) not null,
    timein datetime not null default current_timestamp,
    timeout datetime,
    hours float default 0.0
  ");
conn.exec(sql);
}

bool timeIn(string empid, string fullname)
{
  string sql = "insert into timecards(empid, fullname, timein) values(?, ?, now())";
  Prepared pstmt = conn.prepare(sql);
  pstmt.setArgs(empid, fullname);
  if(conn.exec(pstmt) > 0) return true;
  return false;
}

bool noTimeIn(string empid)
{
  string sql = "select * from timecards where empid=? and date(timein)=?";
  Prepared pstmt = conn.prepare(sql);
  string today = getToday();
  pstmt.setArgs(empid, today);
  Row[] rows = conn.query(pstmt).array;
  if(rows.length == 0) return true;
  return false;
}

string getToday()
{
  Row[] rows = conn.query("select curdate() + 0").array;
  return to!string(rows[0][0]);
}
```

```
}

Timecard[] getTimecards(bool loggedIn)
{
    Timecard[] cards;
    string sql;
    if(loggedIn)
        sql = "select * from timecards order by timein desc";
    else
        sql = "select * from timecards where date(timein) = curdate() order by timein desc";
    Row[] rows = conn.query(sql).array;
    if(rows.length != 0)
    {
        foreach(row; rows)
        {
            Timecard c;
            c.id = to!int(to!string(row[0]));
            c.empid = to!string(row[1]);
            c.fullname = to!string(row[2]);
            c.timein = to!string(row[3]);
            c.timeout = to!string(row[4]);
            c.hours = to!float(to!string(row[5]));
            cards ~= c;
        }
    }
    return cards;
}

bool noTimeOut(string empid)
{
    string sql = "select * from timecards where timeout is null and empid=? and date(timein)=?";
    Prepared pstmt = conn.prepare(sql);
```

```
        pstmt.setArgs(empid, getToday);
        Row[] rows = conn.query(pstmt).array;
        if(rows.length != 0) return true;
        return false;
    }

    bool timeOut(string empid)
    {
        // we want to include the fraction of the hour
        // so we use the MySQL function time_to_sec()
        // 3600 is seconds in an hour
        // but we round the result to just two decimals
        string sql =
            "update timecards set timeout=now(),
                hours=round((time_to_sec(now())-time_to_sec(timein))/3600,2)
                where empid=? and date(timein)=curdate()";
        Prepared pstmt = conn.prepare(sql);
        pstmt.setArgs(empid);
        if(conn.exec(pstmt) > 0) return true;
        return false;
    }

    Timecard getTimecard(int id)
    {
        string sql = "select * from timecards where id=?";
        Prepared pstmt = conn.prepare(sql);
        pstmt.setArgs(id);
        Row[] rows = conn.query(pstmt).array;
        Timecard c;
        if(rows.length > 0)
        {
            Row row = rows[0];
```

```
c.id = to!int(to!string(row[0]));
c.empid = to!string(row[1]);
c.fullname = to!string(row[2]);
c.timein = to!string(row[3]);
c.timeout = to!string(row[4]);
c.hours = to!float(to!string(row[5]));
}
return c;
}

void editTimecard(Timecard t)
{
    Prepared pstmt;
    if(to!string(t.timeout) == "null")
    {
        string sql =
            "update timecards set
            timein=str_to_date(?, '%Y-%b-%d %H:%i:%S'),
            timeout=null
            where id=?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.id);
    }
    else
    {
        string sql =
            "update timecards set
            timein=str_to_date(?, '%Y-%b-%d %H:%i:%S'),
            timeout=str_to_date(?, '%Y-%b-%d %H:%i:%S'),
            hours=round((time_to_sec(timeout)-time_to_sec(timein))/3600,2)
            where id=?";
        pstmt = conn.prepare(sql);
    }
}
```

```
        pstmt.setArgs(t.timein, t.timeout, t.id);
    }
    conn.exec(pstmt);
}

void deleteTimecard(int id)
{
    string sql = "delete from timecards where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    conn.exec(pstmt);
}

int[] getTheWeeks()
{
    string sql = "select distinct week(timein) as weeks from timecards order by weeks";
    Row[] rows = conn.query(sql).array;
    int[] weeks;
    foreach (row; rows) weeks ~= to!int(to!string(row[0]));
    return weeks;
}
}
```

## The timesheet controller

source\sheetscontrol.d:

```
module sheetcontrol;

import vibe.vibe;
import basecontrol;
import empmodel;
import cardmodel;
import sheetmodel;

class TimesheetController : BaseController
{
    @auth
    void getCreateTimesheet(string _authUser, string _error = null)
    {
        string error = _error;
        int[] weeks = cardModel.getTheWeeks;
        bool loggedIn = m_user.loggedIn;
        render!("sheetcreate.dt", error, loggedIn, weeks);
    }

    @errorDisplay!getCreateTimesheet
    void postCreateTimesheet(int week)
    {
        sheetModel.createTimesheet(week);
        redirect("timesheets");
    }

    void getTimesheets()
    {
```

```
    Timesheet[] sheets = sheetModel.getTimesheets;
    bool loggedIn = m_user.loggedIn;
    int week = sheetModel.getTheWeek;
    render!("sheetlist.dt", sheets, loggedIn, week);
}
}
```

## The timesheet model

source\sheetsmodel.d:

```
module sheetmodel;

import mysql;
import std.array;
import std.conv;
import basemodel;

struct Timesheet
{
    int id;
    int period;
    string empid;
    string fullname;
    float hours;
}

class TimesheetModel : BaseModel
{
    void createTable()
    {
        string sql = "drop table if exists timesheets";
        conn.exec(sql);
        sql =
            "create table timesheets
            (
                id int auto_increment primary key,
                period int not null,
                empid char(4) not null,
            "
    }
}
```

```
    fullname varchar(50) not null,
    hours float
  )";
conn.exec(sql);
}

void createTimesheet(int week)
{
  string sql = "delete from timesheets";
conn.exec(sql);
sql =
  "insert into timesheets(period, hours, empid, fullname)
   (select week(timein) as period, sum(hours), empid, fullname from timecards
    where week(timein)=? group by period, empid, fullname)";
Prepared pstmt = conn.prepare(sql);
pstmt.setArgs(week);
conn.exec(pstmt);
}

Timesheet[] getTimesheets()
{
  string sql = "select * from timesheets order by empid";
Row[] rows = conn.query(sql).array;
Timesheet[] sheets;
foreach (row; rows)
{
  Timesheet t;
  t.id = to!int(to!string(row[0]));
  t.period = to!int(to!string(row[1]));
  t.empid = to!string(row[2]);
  t.fullname = to!string(row[3]);
  t.hours = to!double(to!string(row[4]));
}
```

```
        sheets ~= t;
    }
    return sheets;
}

int getTheWeek()
{
    Row[] rows = conn.query("select period from timesheets limit 1").array;
    if(rows.length == 0) return 0;
    return to!int(to!string(rows[0][0]));
}
```

## The CSS files

Here is views\cssclock.dt:

```
:css
.clock-wrapper
{
    margin: 0 10px;
    text-align: center;
}

.clock-grid
{
    margin: 0 auto;
    padding: 0;
    width: 900px;
    height: auto;
    display: grid;
    grid-template: 90px auto / repeat(12, 1fr);
    grid-template-areas:
        "t0 t0 t0"
        "ci ci ci ci ci ci co co co co co co";
    gap: 20px;
    color: #444;
    text-align: center;
}

.clock-heading
{
    margin-bottom: 10px;
    padding: 0;
    grid-area: t0;
```

```
line-height: 90px;
color: black;
font-weight: bold;
text-align: center;
}

#clock-title
{
  font-size: 30px;
}

.clock-text-center
{
  text-align: center;
}

.clock-area
{
  font-size: 40px;
  font-weight: bold;
  text-align: center;
  width: 370px;
  height: 400px;
  margin: 10px auto;
  padding: 0;
}

#clock-in
{
  grid-area: ci;
}
```

```
#clock-out
{
  grid-area: co;
}

.clock-btn
{
  margin: 20px auto;
  padding: 10px 60px;
  border: none;
  background: #363636;
  width: 100%;
  transition: ease-in all 0.3s;
  color: #fff;
  height: auto;
  text-align: center;
  border-radius: 30px;
  font-size: 20px;
  font-weight: bold;
  text-decoration: none;
  cursor: pointer;
}

.clock-btn:hover
{
  background: brown;
  color: #fff;
}

#clock-time
{
  font-size: 26px;
```

```
margin: 100px auto 0 auto;
color: black;
text-align: center;
}

#clock-form-select
{
margin: 0 auto;
text-align: center;
font-size: 20px;
font-weight: bold;
}

.success
{
font-size:16px;
font-weight: bold;
color: black;
text-align: center;
}
```

Here is views\cssfooter.dt:

```
:css
footer
{
    position: fixed;
    bottom: 0;
    margin-top: 15px;
    background: #076;
    padding: 5px 0;
    width: 100%;
}

.copyright
{
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    font-size: 14px;
    text-align: center;
    color: white;
}
```

Here is views\cssformgrid.dt:

```
:css
.form-grid-wrapper
{
    width: 500px;
    margin: 20px;
    padding: 1px 20px 20px 0;
    border-radius: 20px;
}

.form-grid
{
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 10px;
    padding-top: 5px;
}

.form-grid-label
{
    width: 100%;
    font-size: 16px;
    text-align: right;
    padding: 2px;
}

.form-grid-input
{
    width: 100%;
    font-size: 14px;
    padding: 0;
```

```
}

.form-grid-field
{
    width: 100%;
    font-size: 16px;
    border-bottom: 1px solid black;
    padding: 2px;
}

.form-grid-button
{
    width: 100%;
    font-size: 14px;
    height: 30px;
    margin-top: 10px;
}

.form-hidden
{
    padding: 0;
    margin: 0;
    display: inline;
}
```

Here is views\csslayout.dt:

```
:css
html, body, container
{
    margin: 0;
    padding: 0;
    width: 100%;
    height: 100%;
}

.container
{
    display: grid;
    grid-template-rows: 38px auto 10px;
    background-color: cyan;
}

main
{
    margin: 0;
    padding: 40px 15px;
    font-family: Sans-serif;
    width: auto;
    height: 100%;
}

.text-control
{
    grid-column: 1 / 3;
}
```

```
#but-reset
{
  grid-column: 1 / 2;
}

#but-submit
{
  grid-column: 2 / 3;
}

.error-div
{
  margin-bottom: 5px;
}

.error-message
{
  color: brown;
  font-size: 16px;
  font-weight: bold;
  text-align: left;
}

.center-align
{
  margin: 0 auto;
  text-align: center;
}

.right-align
{
  text-align: right;
```

}

Here is views\cssmenu.dt:

```
:css
nav
{
    position: fixed;
    top: 0;
    margin: 0;
    padding: 0;
    display: inline-block;
    background: #076;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    width: 100%;
}

nav ul
{
    margin:0;
    padding:0;
    list-style-type:none;
    float:left;
    display:inline-block;
}

nav ul li
{
    position: relative;
    margin: 0;
    float: left;
    display: inline-block;
}
```

```
li > a:only-child:after { content: ''; }

nav ul li a
{
    padding: 15px 20px;
    display: inline-block;
    color: white;
    text-decoration: none;
}

nav ul li a:hover
{
    opacity: 0.5;
}

nav ul li ul
{
    display: none;
    position: absolute;
    left: 0;
    background: #076;
    float: left;
    width: 190px;
}

nav ul li ul li
{
    width: 100%;
    border-bottom: 1px solid rgba(255,255,255,.3);
}

nav ul li ul li a
```

```
{  
  padding: 10px 20px;  
}  
  
nav ul li:hover ul  
{  
  display: block;  
}  
  
.link-right  
{  
  float: right;  
  margin-right: 20px;  
}  
  
.modal-form  
{  
  position: fixed;  
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
  top: 0;  
  right: 0;  
  bottom: 0;  
  left: 0;  
  background: rgba(0,0,0,0.5);  
  z-index: 99999;  
  opacity: 0;  
  pointer-events: none;  
}  
  
#login:target, #find_employee:target  
{  
  opacity: 1;
```

```
    pointer-events: auto;
}

.modal-form-grid
{
    display: grid;
    grid-template: 30px 30px 30px / 1fr 1fr;
    grid-gap: 10px;
}

.modal-form-wrapper-login
{
    width: 250px;
    position: absolute;
    right: 0;
    margin-top: 40px;
    padding: 25px 20px;
    border-radius: 10px;
    text-align: center;
    background-color: whitesmoke;
}

.modal-form-wrapper-find_employee
{
    width: 250px;
    position: relative;
    left: 220px;
    margin-top: 40px;
    padding: 25px 20px;
    border-radius: 10px;
    text-align: center;
    background-color: whitesmoke;
```

}

Here is views\csstable.dt:

```
:css
.table-wrapper
{
    margin: 10px 0;
    padding-bottom: 40px;
}

table
{
    padding: 10px 20px;
    border-spacing: 0;
    border-collapse: collapse;
}

table td, table th
{
    margin: 0;
    padding: 2px 10px;
    text-align: left;
}

.no-border
{
    border: 0;
}

tr:nth-child(odd)
{
    background-color: #DADADA;
}
```

## The timecard views

views\cardelete.dt:

```
extends layout
block maincontent
div.form-grid-wrapper
    h2 Delete this time card entry?
    form.form-grid(method="post", action="delete_timecard")
        label.form-grid-label Employee :
        label.form-grid-text #{e.empid} - #{e.fname} #{e.lname}
        label.form-grid-label Punched in :
        label.form-grid-text #{t.timein}
        label.form-grid-label Punched out :
        label.form-grid-text #{t.timeout}
        input(type="hidden", name="id", value="#{t.id}")
        div
            a(href="timecards")
            button.form-grid-button(type="button") Cancel
        div
            input.form-grid-button(type="submit", value="Delete")
```

views\cardedit.dt:

```
extends layout
block maincontent
div.form-grid-wrapper
    h2 Edit time card entry
    -if(error)
        div.error-div
            span.error-message #{error}
```

```
form.form-grid(method="post", action="edit_timecard")
label.form-grid-label Employee :
label.form-grid-text #{t.empid} - #{t.fullname}
label.form-grid-label Time in :
input.form-grid-input(type="text", name="t_timein", value="#{t.timein}")
label.form-grid-label Time out :
input.form-grid-input(type="text", name="t_timeout", value="#{t.timeout}")
div
input(type="hidden", name="t_id", value="#{t.id}")
input(type="hidden", name="t_empid", value="#{t.empid}")
input(type="hidden", name="t_fullname", value="#{t.fullname}")
input(type="hidden", name="t_hours", value="#{t.hours}")
div
div
a(href="timecards")
    button.form-grid-button(type="button") Cancel
div
input.form-grid-button(type="submit", value="Submit")
```

views\cardlist.dt:

```
extends layout
block maincontent
div.table-wrapper
-if(loggedIn)
    h2 All time cards
-else
    h2 #{today}
table
tr
th Emp #
th Name
```

```

th Time in
th Time out
th Hours
-if(loggedIn)
    th Action
-foreach(c; cards)
    tr
        td #{c.empid}
        td #{c.fullname}
        td #{c.timein}
        td #{c.timeout}
        td #{c.hours}
        -if(loggedIn)
            td &nbs;
            form.form-hidden(method="get", action="edit_timecard")
                input(type="hidden", name="id", value="#{c.id}")
                input(type="image", src="images/pencil.png", height="15px")
            | &nbs;
            form.form-hidden(method="get", action="delete_timecard")
                input(type="hidden", name="id", value="#{c.id}")
                input(type="image", src="images/trash.png", height="15px")
            | &nbs;

```

views\timein.dt:

```

extends layout
block maincontent
    div.main
        div.clock-wrapper
            div.clock-grid
                div.clock-heading#clock-title Lorem Ipsum Company Timekeeping System
                form#clock-in.clock-area(method="post", action="time_in")

```

```



```

views\timeout.dt:

```

extends layout
block maincontent
    div.main
        div.clock-wrapper
            div.clock-grid
                div.clock-heading#clock-title Lorem Ipsum Company Timekeeping System
                div#clock-in.clock-area
                    img(src="images/timein.png", alt="photo of a clock")

```

```
a.clock-btn(href="time_in") Punch In
form#clock-out.clock-area(method="post", action="time_out")
    div#clock-time
        select#clock-form-select(name="empidname")
            -foreach(e; emps)
                option(value="#{e.empid} #{e.fname} #{e.lname}") #{e.empid} #{e.fname} #{e.lname}
        input.clock-btn(type="submit", value="Punch Out")
    -if(error)
        div.error-div
            span.error-message #{error}
    -else if(success)
        div.success #{success}
:javascript
    const displayTime = document.querySelector("#clock-time");
    function showTime() {
        let time = new Date();
        displayTime.innerText = time.toLocaleString("en-CA", { hour12: true });
        setTimeout(showTime, 1000);
    }
    showTime();
```

## The employee views

views\ empadd.dt:

```
extends layout
block maincontent
-if(error)
  div.error-div
    span.error-message #{error}
div.form-grid-wrapper
  h2.center-align New employee details
  br
  form.form-grid(method="post", action="add_employee", enctype="multipart/form-data")
    label.form-grid-label Employee number
    input.form-grid-input(type="empid", name="e.empid", placeholder="employee number", required)
    label.form-grid-label Email address
    input.form-grid-input(type="email", name="e_email", placeholder="email@company.com", required)
    label.form-grid-label Password
    input.form-grid-input(type="password", name="e_passw", placeholder="password", required)
    label.form-grid-label First name
    input.form-grid-input(type="text", name="e_fname", placeholder="First name", required)
    label.form-grid-label Last name
    input.form-grid-input(type="text", name="e_lname", placeholder="Last name", required)
    label.form-grid-label Phone
    input.form-grid-input(type="text", name="e_phone", placeholder="Phone number")
    label.form-grid-label Department
    select#dept.form-grid-input(name="e_dept")
      -foreach(dep; departments)
        option(value="#{dep}") #{dep}
    label.form-grid-label Salary grade
    select#dept.form-grid-input(name="e_payrate")
      -foreach(pay; payrates)
```

```

    option(value="#{pay}") #{pay}
label.form-grid-label Street address (no city)
input.form-grid-input(type="text", name="e_street", placeholder="Street address", required)
label.form-grid-label City
input.form-grid-input(type="text", name="e_city", placeholder="City", required)
label.form-grid-label Province
select#province.form-grid-input(name="e_province")
-foreach(prov; provinces)
    option(value="#{prov[0]}") #{prov[1]}
label.form-grid-label Postal code
input.form-grid-input(type="text", name="e_postcode", placeholder="A1A 1A1")
label.form-grid-label ID Picture
input.form-grid-input(type="file", name="picture")
input(type="hidden", name="e_photo")
input(type="hidden", name="e_id", value="1")
input.form-grid-button(type="reset", value="Clear form")
input.form-grid-button(type="submit", value="Submit")

```

views\empdelete.dt:

```

extends layout
block maincontent
-if(error)
    div.error-div
        span.error-message #{error}
div.form-grid-wrapper
    h2.center-align Delete #{e.fname} #{e.lname}'s record?
    br
    form.form-grid(method="post", action="delete_employee")
        span.form-grid-label Employee number:
        span.form-grid-field #{e.empid}
        span.form-grid-label Email address:

```

```
span.form-grid-field #{e.email}
span.form-grid-label First name:
span.form-grid-field #{e.fname}
span.form-grid-label Last name:
span.form-grid-field #{e.lname}
span.form-grid-label Phone:
span.form-grid-field #{e.phone}
span.form-grid-label Department:
span.form-grid-field #{e.deprt}
span.form-grid-label Salary grade:
span.form-grid-field #{e.payrate}
span.form-grid-label Street address:
span.form-grid-field #{e.street}
span.form-grid-label City:
span.form-grid-field #{e.city}
span.form-grid-label Province:
span.form-grid-field #{e.province}
span.form-grid-label Postal code:
span.form-grid-field #{e.postcode}
span.form-grid-label ID Picture:

```

views\empedit.dt:

```
extends layout
block maincontent
-if(error)
    div.error-div
```

```
span.error-message #{error}
div.form-grid-wrapper
    h2.center-align Edit #{e.fname} #{e.lname} details
    br
    form.form-grid(method="post", action="edit_employee", enctype="multipart/form-data")
        label.form-grid-label Employee number
        input.form-grid-input(type="empid", name="e.empid", value="#{e.empid}")
        label.form-grid-label Email address
        input.form-grid-input(type="email", name="e_email", value="#{e.email}")
        label.form-grid-label Password
        input.form-grid-input(type="password", name="e_passw", value="#{e.passw}")
        label.form-grid-label First name
        input.form-grid-input(type="text", name="e_fname", value="#{e.fname}")
        label.form-grid-label Last name
        input.form-grid-input(type="text", name="e_lname", value="#{e.lname}")
        label.form-grid-label Phone
        input.form-grid-input(type="text", name="e_phone", value="#{e.phone}")
        label.form-grid-label Department
        select#dept.form-grid-input(name="e_dept")
            -foreach(dep; departments)
                -if(dep == e.deprt)
                    option(value="#{dep}", selected) #{dep}
                -else
                    option(value="#{dep}") #{dep}
        label.form-grid-label Salary grade
        select#paygd.form-grid-input(name="e_payrate", value="#{e.payrate}")
            -foreach(pay; payrates)
                -if(pay == e.payrate)
                    option(value="#{pay}", selected) #{pay}
                -else
                    option(value="#{pay}") #{pay}
        label.form-grid-label Street address (no city)
```

```

input.form-grid-input(type="text", name="e_street", value="#{e.street}")
label.form-grid-label City
input.form-grid-input(type="text", name="e_city", value="#{e.city}")
label.form-grid-label Province
select#province.form-grid-input(name="e_province", value="#{e.province}")
    -foreach(prov; provinces)
        -if(prov[0] == e.province)
            option(value="#{prov[0]}", selected) #{prov[1]}
        -else
            option(value="#{prov[0]}") #{prov[1]}
label.form-grid-label Postal code
input.form-grid-input(type="text", name="e_postcode", value="#{e.postcode}")
label.form-grid-label ID Picture
img(src="#{e.photo}", height="100px")
div
    input.form-grid-input(type="file", name="picture")
    input(type="hidden", name="prevPassw", value="#{e.passw}")
    input(type="hidden", name="e_photo", value="#{e.photo}")
    input(type="hidden", name="e_id", value="#{e.id}")
    input(type="hidden", name="id", value="#{e.id}")
    a(href="all_employees")
        button.form-grid-button(type="button") Cancel
    input.form-grid-button(type="submit", value="Submit")

```

views\emplist.dt:

```

extends layout
block maincontent
    h2 Lorem Ipsum Employees
    div.table-wrapper
        -if(error)
            div.error-div

```

```

.error-message #{error}
table
  tr
    th Emp #
    th Name
    th Department
    th Phone number
    th Email address
    th Action
-foreach(e; emps)
  tr
    td #{e.empid}
    td #{e.fname} #{e.lname}
    td #{e.deprt}
    td #{e.phone}
    td #{e.email}
    td &nbsp;
      form.form-hidden(method="get", action="edit_employee")
        input(type="hidden", name="id", value="#{e.id}")
        input(type="image", src="images/pencil.png", height="15px")
      | &nbsp;
      form.form-hidden(method="get", action="delete_employee")
        input(type="hidden", name="id", value="#{e.id}")
        input(type="image", src="images/trash.png", height="15px")
      | &nbsp;

```

Here is views\empshow.dt:

```

extends layout
block maincontent
  div.form-grid-wrapper
    h2.center-align #{e.fname} #{e.lname} details

```

```
br
div.form-grid
    span.form-grid-label Employee number:
    span.form-grid-field #{e.empid}
    span.form-grid-label Email address:
    span.form-grid-field #{e.email}
    span.form-grid-label First name:
    span.form-grid-field #{e.fname}
    span.form-grid-label Last name:
    span.form-grid-field #{e.lname}
    span.form-grid-label Phone:
    span.form-grid-field #{e.phone}
    span.form-grid-label Department:
    span.form-grid-field #{e.deprt}
    span.form-grid-label Salary grade:
    span.form-grid-field #{e.payrate}
    span.form-grid-label Street address:
    span.form-grid-field #{e.street}
    span.form-grid-label City:
    span.form-grid-field #{e.city}
    span.form-grid-label Province:
    span.form-grid-field #{e.province}
    span.form-grid-label Postal code:
    span.form-grid-field #{e.postcode}
    span.form-grid-label ID Picture:
    img(src="#{e.photo}", height="80px")
a(href="all_employees")
    button.form-grid-button(type="button") Close
a(href="edit_employee?id=#{e.id}")
    button.form-grid-button(type="button") Edit
```

## The layout views

views\layout.dt:

```
doctype 5
html
  head
    title Employee Timekeeping System
    include cssclock
    include cssfooter
    include cssformgrid
    include csslayout
    include cssmenu
    include csstable
  body
    div.container
      nav
        include menu
      main.content
        block maincontent
      footer
        include footer
```

views\menu.dt:

```
ul
  li
    a(href="/") Home
  li
    a(href="#") Timecards
    ul
      li
```

```
a(href="timecards") View timecards
li
    a(href="create_timesheet") Create timesheet
li
    a(href="#") Employees
    ul
        li
            a(href="all_employees") View employees
        -if(loggedIn)
            li
                a(href="#find_employee") Find an employee
        -else
            li
                a(href="#login") Find an employee
        li
            a(href="add_employee") New employee
ul.link-right
    li
        -if(loggedIn)
            a(href="logout") Logout
        -else
            a(href="#login") Login
div#find_employee.modal-form
    div.modal-form-wrapper-find_employee
        form.modal-form-grid(method="post", action="find_employee")
            input.text-control(name="fname", type="text", placeholder=" First name", required)
            input.text-control(name="lname", type="text", placeholder=" Last name", required)
            input#but-reset(type="reset", value="Clear")
            input#but-submit(type="submit", value="Find")
        br
        a.close(href="#close") Cancel
div#login.modal-form
```

```
div.modal-form-wrapper-login
  form.modal-form-grid(method="post", action="login")
    input.text-control(name="email", type="email", placeholder=" email address")
    input.text-control(name="password", type="password", placeholder=" password")
    input#but-reset(type="reset", value="Clear")
    input#but-submit(type="submit", value="Login")
  br
  a.close(href="#close") Cancel
```

views\footer.dt:

```
div.copyright Copyright &copy; The Lorem Ipsum Company 2023
```

## The home view

views\index.dt:

## The timesheet views

views\sheetcreate.dt:

```
extends layout
block maincontent
    include cssformgrid.dt
    div.form-grid-wrapper
        -if(error)
            div.error-div
                span.error-message #{error}
        form.form-grid(method="post", action="create_timesheet")
            div
                label.form-grid-label For week number
                select.form-grid-input(name="week")
                    -foreach(w; weeks)
                        option(value="#{w}") #{w}
            div
                input.form-grid-button(type="submit", value="Generate timesheet")
```

views\sheetslist.dt:

```
extends layout
block maincontent
    h2 Timesheet for week #{week}
    div.table-wrapper
        table
            tr
                th Emp #
                th Name
                th Hours
            -foreach(s; sheets)
```

```
tr
  td #{s.empid}
  td #{s.fullname}
  td.right-align #{s.hours}
```

## The project configuration file

dub.json:

```
{  
    "authors": [  
        "Owner"  
    ],  
    "copyright": "Copyright © 2023, Owner",  
    "dependencies": {  
        "mysql-native": "~>3.2.2",  
        "vibe-d": "~>0.9"  
    },  
    "description": "A simple vibe.d server application.",  
    "license": "proprietary",  
    "name": "loremtime"  
}
```

...and that's the last one.

That's all! Bye!