

Build web apps with



by learning from a learner

Second edition

Rey Valeza

Introduction

Welcome to the second edition of this tutorial. I wrote the first version in 2021 (it's now January 2023) and while reviewing it, I felt I should expand more on the database side and focus on MySQL as I know there are so many ageing deployed small apps out there that might feel the need to port to another framework. This book is by no means comprehensive since the purpose is to offer a tutorial for learners of an obscure web framework. 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 in either D or Vibe.d.

I hanker for Vibe.d and its simplicity 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.

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.

Why did Rails fail to gain traction? Because by the time it was seriously attracting potential converts, they changed drastically by using the then hot-off-the-oven buzzword REST, ignoring users who were happy with the previous ways and frustrating new converts desperately looking for updated tutorials and documentation.

Why did Meteor fail too? Because by the time it was drawing in the crowds, they decided to adopt the very popular React. For existing users, the quandary was either go the old Blaze way or go the React way. Which way will win in the end? Since there was no way of divining the future, new and old users alike took the third way, abandon ship.

You do not make drastic changes to your already revolutionary product when you haven't achieved a critical mass of users yet. You will not expand user base by constantly adopting each and every sizzling-new buzzword, you only end up losing them. Remember how Java did it? There was just so much hype at the beginning. There was just so much buzz. Java was everywhere. Everybody wanted in on the action. And Sun backed it all up with solid, updated documentation. So even if Java kept changing afterwards, it has already achieved a critical mass of followers long ago so that users have no choice but to follow changes as a captive audience. That's how you do it. Java slew the lumbering dinosaurs then. Sadly, Java is now the dinosaur.

The Vibe.d framework was created by Sönke Ludwig (there are now a bunch of competent 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.

I see that there is a dearth of tutorials on Vibe.d. 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.

Configuring your system for Vibe.d

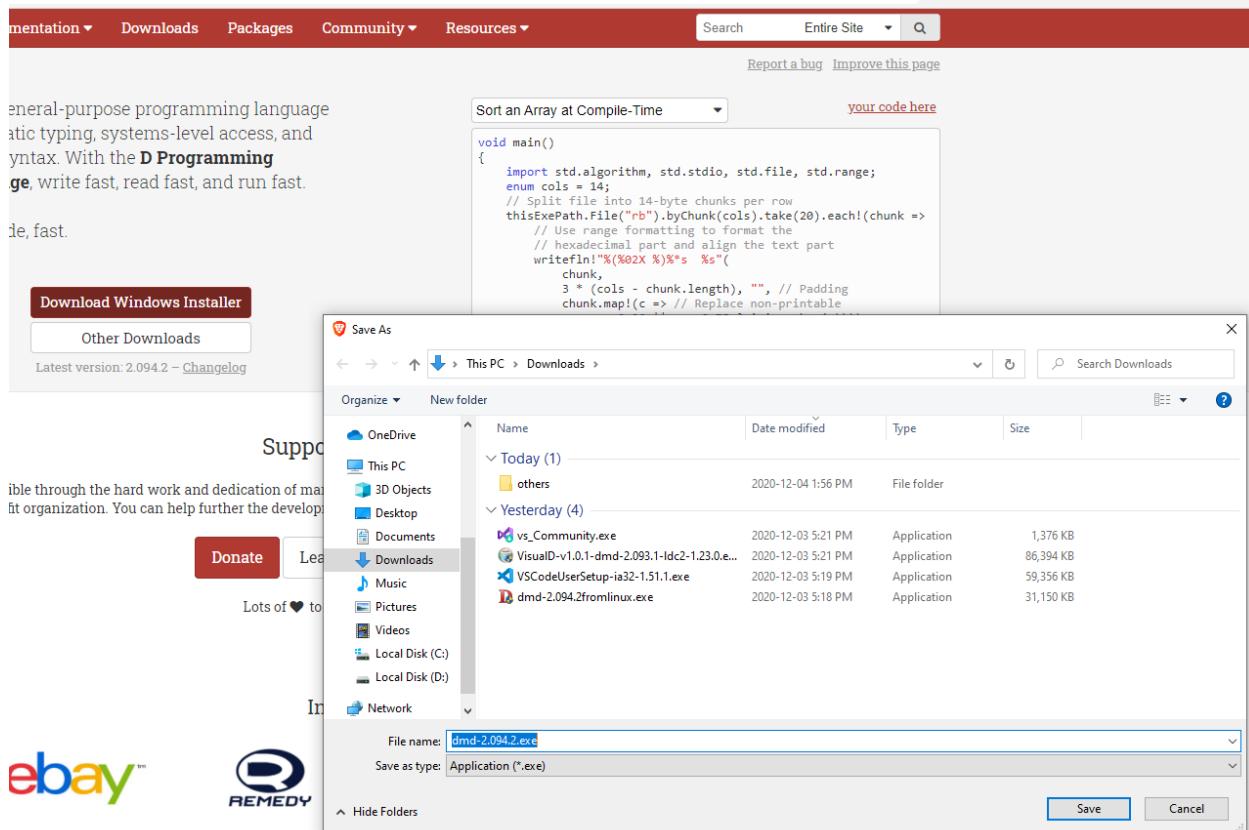
I am using Windows 10 but you should be fine with another OS you are familiar with, and I assume you know the equivalent commands in your system's shell terminal. We will be using the terminal or command window a lot. I wrote the code in my Linux box then ported them to Windows to write this book, as 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, we need at least a D compiler and a text editor.

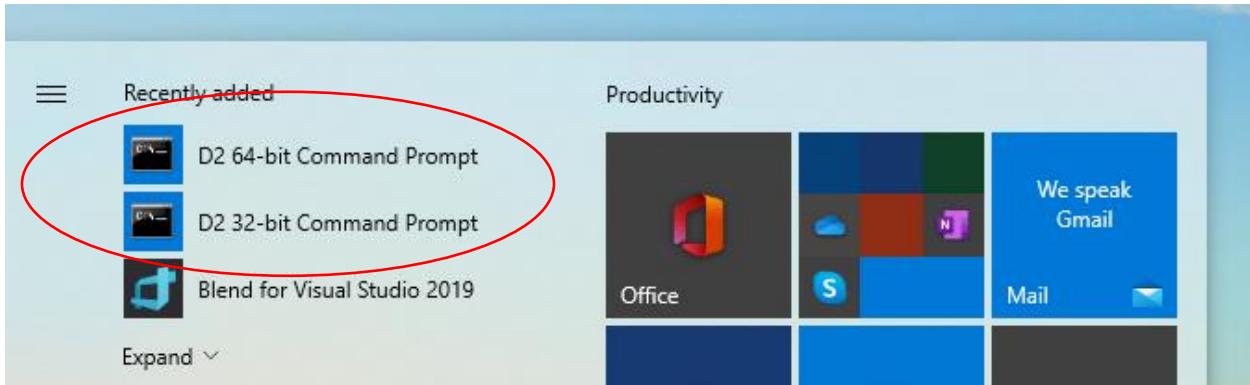
Regarding compilers, there are three choices: DMD, GDC and LDC. It is advised that, for development purposes, we should use the DMD compiler for its speed of compilation, then, when we are ready to deploy the final production version, use one of the other compilers (GDC or LDC) for the runtime speed 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 click on the ‘Download Windows Installer’ button.



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>...]`

Where:

`<file>` D source file
`<arg>` Argument to pass when running the resulting program

`<option>:`

- `@<cmdfile>` read arguments from cmdfile
- `-allinst` generate code for all template instantiations
- `-betterC` omit generating some runtime information and helper functions
- `-boundscheck=[on|safeonly|off]`
 - bounds checks on, in @safe only, or off
- `-c` compile only, do not link
- `-check=[assert|bounds|in|invariant|out|switch][=[on|off]]`
 - enable or disable specific checks
- `-check=[h|help|?]` list information on all available checks
- `-checkaction=[D|C|halt|context]`

behavior on assert/boundscheck/finalswitch failure

-checkaction=[h|help|?]
list information on all available check actions

-color turn colored console output on

-color=[on|off|auto]
force colored console output on or off, or only when not redirected (default)

-conf=<filename> use config file at filename

-cov do code coverage analysis

-cov=ctfe Include code executed during CTFE in coverage report

-cov=<nnn> require at least nnn% code coverage

-D generate documentation

-Dd<directory> write documentation file to directory

-Df<filename> write documentation file to filename

-d silently allow deprecated features and symbols

-de issue an error when deprecated features or symbols are used (halt compilation)

-dw issue a message when deprecated features or symbols are used (default)

-debug compile in debug code

-debug=<level> compile in debug code <= level

-debug=<ident> compile in debug code identified by ident

-debuglib=<name> set symbolic debug library to name

-defaultlib=<name>
set default library to name

-deps print module dependencies (imports/file/version/debug/lib)

-deps=<filename> write module dependencies to filename (only imports)

-extern-std=<standard>
set C++ name mangling compatibility with <standard>

-extern-std=[h|help|?]
list all supported standards

-g add symbolic debug info

-gf emit debug info for all referenced types

-gs always emit stack frame

-gx add stack stomp code

-H generate 'header' file

-Hd=<directory> write 'header' file to directory

-Hf=<filename> write 'header' file to filename

-HC[=[silent|verbose]]
generate C++ 'header' file

-HC=[?|h|help] list available modes for C++ 'header' file generation

-HCd=<directory> write C++ 'header' file to directory

-HCf=<filename> write C++ 'header' file to filename

--help print help and exit

-I=<directory> look for imports also in directory

-i[=<pattern>] include imported modules in the compilation

-ignore ignore unsupported pragmas

- inline do function inlining
- J=<directory> look for string imports also in directory
- L=<linkerflag> pass linkerflag to link
- lib generate library rather than object files
- lowmem enable garbage collection for the compiler
- m32 generate 32 bit code
- m32mscoff generate 32 bit code and write MS-COFF object files (deprecated use -m32)
- m32omf (deprecated) generate 32 bit code and write OMF object files
- m64 generate 64 bit code
- main add default main() if not present already (e.g. for unittesting)
- makedeps[=<filename>]
 - print dependencies in Makefile compatible format to filename or stdout.
- man open web browser on manual page
- map generate linker .map file
- mcpu=<id> generate instructions for architecture identified by 'id'
- mcpu=[h|help|?] list all architecture options
- mixin=<filename> expand and save mixins to file specified by <filename>
- mscrtlib=<libname>
 - MS C runtime library to reference from main/WinMain/DllMain
- mv=<package.module>=<filespec>
 - use <filespec> as source file for <package.module>
- noboundscheck no array bounds checking (deprecated, use -boundscheck=off)
- O optimize
- o- do not write object file
- od=<directory> write object & library files to directory
- of=<filename> name output file to filename
- op preserve source path for output files
- os=<os> sets target operating system to <os>
- preview=<name> enable an upcoming language change identified by 'name'
- preview=[h|help|?]
 - list all upcoming language changes
- profile profile runtime performance of generated code
- profile=gc profile runtime allocations
- release compile release version
- revert=<name> revert language change identified by 'name'
- revert=[h|help|?]
 - list all revertable language changes
- run <srcfile> compile, link, and run the program srcfile
- shared generate shared library (DLL)
- target=<triple> use <triple> as <arch>-[<vendor>-]<os>[-<cenv>[-<cppenv>]]
- transition=<name>
 - help with language change identified by 'name'
- transition=[h|help|?]
 - list all language changes

```
-unittest      compile in unit tests
-v           verbose
-vasm        list generated assembler for each function
-vcolumns    print character (column) numbers in diagnostics
-verror-style=[digitalmars|gnu]
              set the style for file/line number annotations on compiler messages
-verrors=<num>  limit the number of error messages (0 means unlimited)
-verrors=context show error messages with the context of the erroring source line
-verrors=spec   show errors from speculative compiles such as __traits(compiles,...)
--version     print compiler version and exit
-version=<level> compile in version code >= level
-version=<ident> compile in version code identified by ident
-vgc         list all gc allocations including hidden ones
-vtls        list all variables going into thread local storage
-vtemplates=[list-instances]
              list statistics on template instantiations
-w           warnings as errors (compilation will halt)
-wi          warnings as messages (compilation will continue)
-X           generate JSON file
-Xf=<filename> write JSON file to filename
```

C:\Users\Owner\Downloads>

Here are two helpful 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

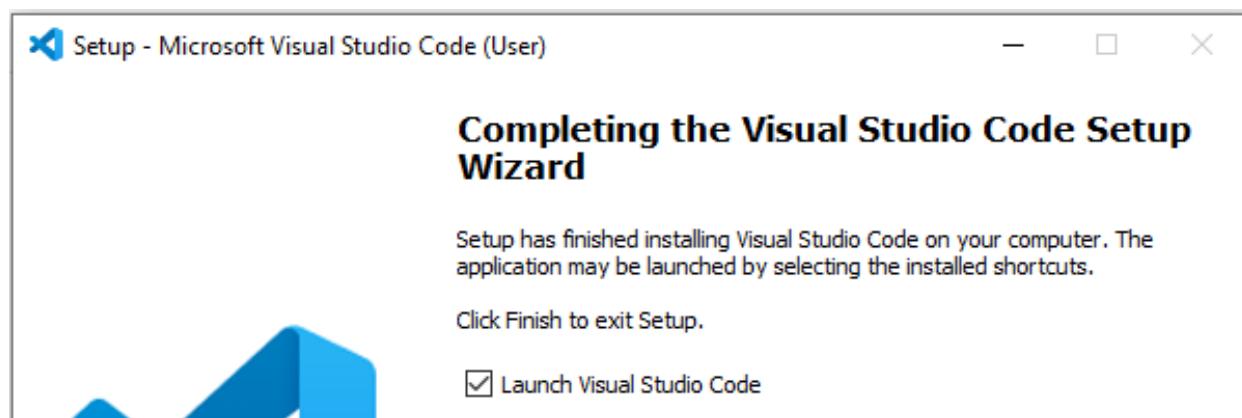
```
dub --version
```

After the compiler is installed, the next thing we 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

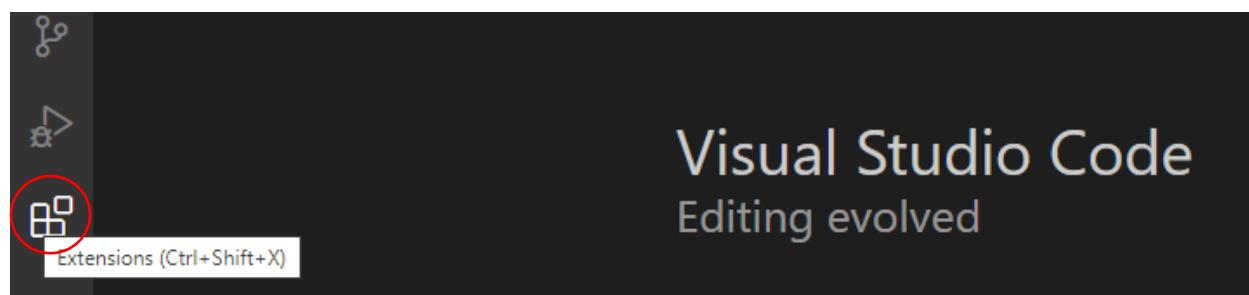
<https://code.visualstudio.com/download>

and choose the installer appropriate for your system.

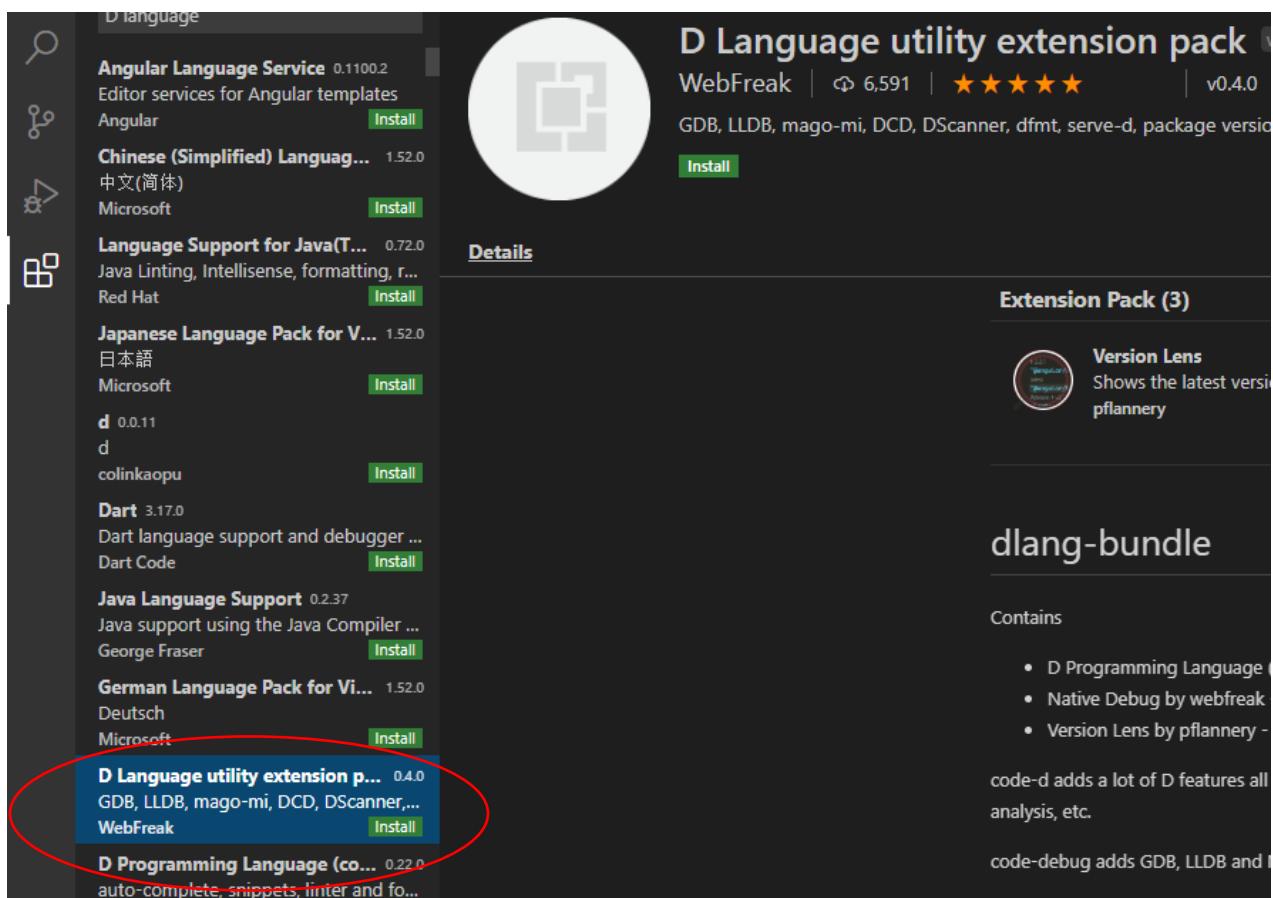
After downloading, simply double-click on the downloaded file and follow the instructions on the setup wizard.



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



Add the D language utility extension pack. With this pack, you will have features like syntax highlighting, code-completion and some error flagging.



The screenshot shows the Visual Studio Code Marketplace search results for 'D language'. On the left, there's a sidebar with icons for search, open files, recent files, and extensions. The main area displays a list of extensions:

- Angular Language Service** 0.1100.2 - Editor services for Angular templates by Angular. [Install](#)
- Chinese (Simplified) Language** 1.52.0 - 中文(简体) by Microsoft. [Install](#)
- Language Support for Java(T...)** 0.72.0 - Java Linting, Intellisense, formatting, r... by Red Hat. [Install](#)
- Japanese Language Pack for V...** 1.52.0 - 日本語 by Microsoft. [Install](#)
- d** 0.0.11 - d by colinkaopu. [Install](#)
- Dart** 3.17.0 - Dart language support and debugger ... by Dart Code. [Install](#)
- Java Language Support** 0.2.37 - Java support using the Java Compiler ... by George Fraser. [Install](#)
- German Language Pack for Vi...** 1.52.0 - Deutsch by Microsoft. [Install](#)
- D Language utility extension p...** 0.4.0 - GDB, LLDB, mago-mi, DCD, DScanner,... by WebFreak. [Install](#) (This item is circled in red)
- D Programming Language (co...** 0.22.0 - auto-complete, snippets, linter and fo... by code-d. [Install](#)

On the right, the details for the highlighted extension are shown:

D Language utility extension pack by WebFreak | ⚡ 6.591 | ★★★★★ v0.4.0

GDB, LLDB, mago-mi, DCD, DScanner, dfmt, serve-d, package versioning, etc.

[Install](#)

Details

Extension Pack (3)

Version Lens Shows the latest version of the extension across multiple releases by pflannery

dlang-bundle

Contains

- D Programming Language (code-d)
- Native Debug by webfreak
- Version Lens by pflannery -

code-d adds a lot of D features all around: auto-complete, snippets, linter and formatter, etc.

code-debug adds GDB, LLDB and DCD support.

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 4EOA-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.

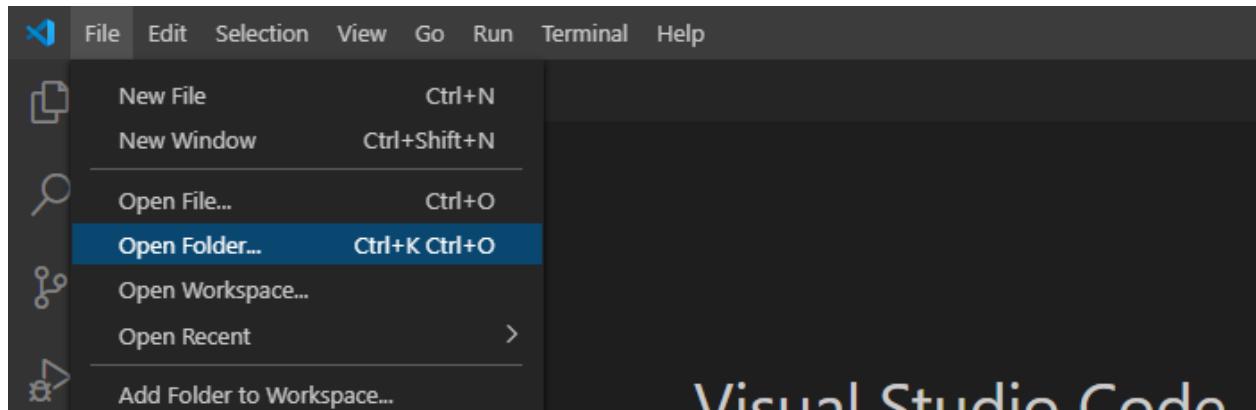
```
[00000000(----) INF] Received signal 2. Shutting down.  
[main(----) INF] Stopped to listen for HTTP requests on ::1:8080  
[main(----) INF] Stopped to listen for HTTP requests on 127.0.0.1:8080  
^C  
c:\vibeprojects\hello>
```

The common command options for dub are

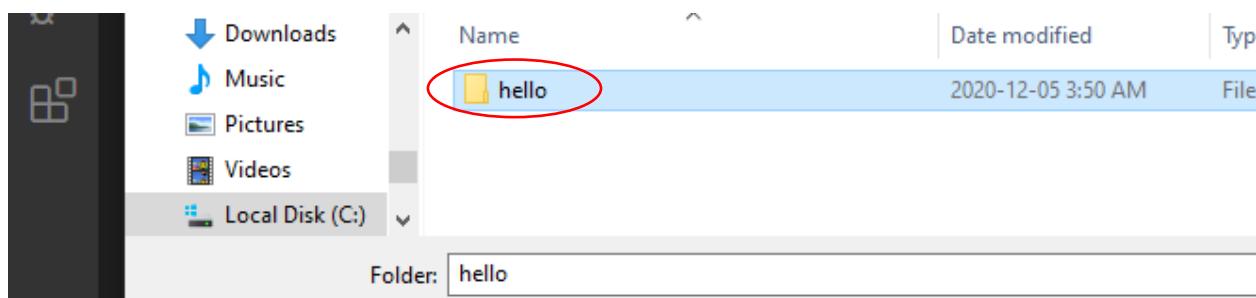
- dub init - create a new project
- dub run - build and run combined
- dub - same as dub run
- dub build - completely compile the whole project and its dependencies but don't run
- dub test - runs the unit tests

Let us take a closer look at the default application.

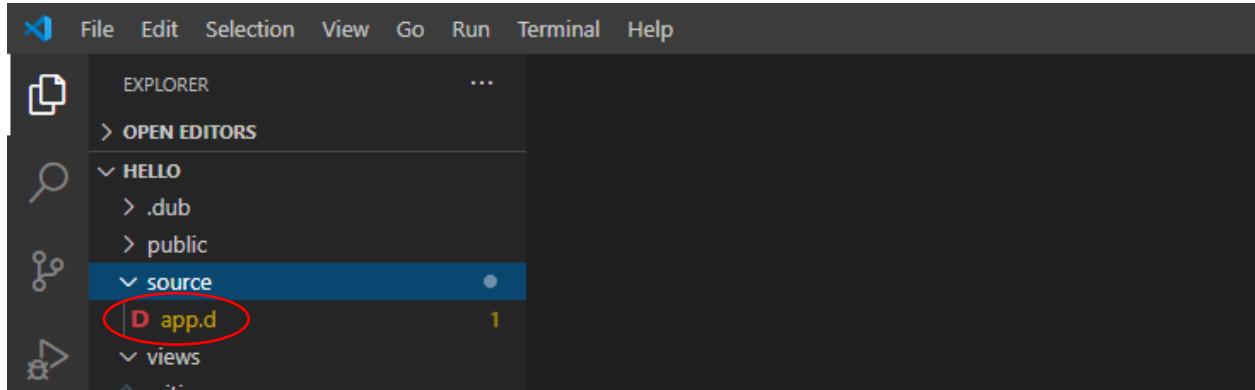
Open VS Code and go to File->Open folder...



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();
}

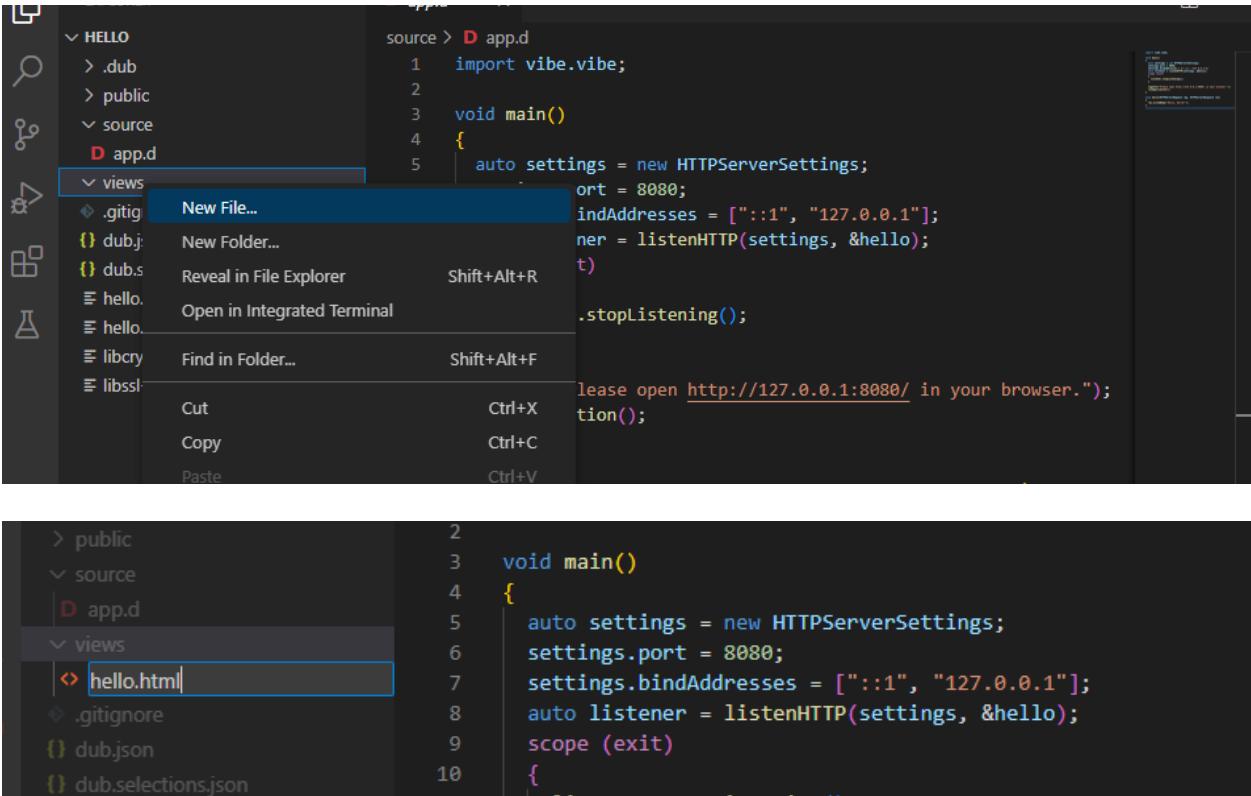
void hello(HTTPServerRequest req, HTTPServerResponse res)
{
    res.writeBody("Hello, World!");
}
```

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

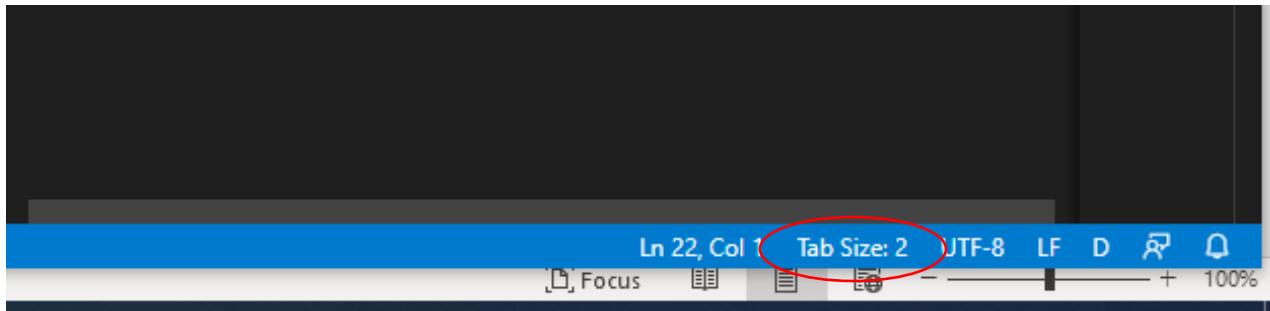
Inside the views folder, create a new HTML file named views\hello.html



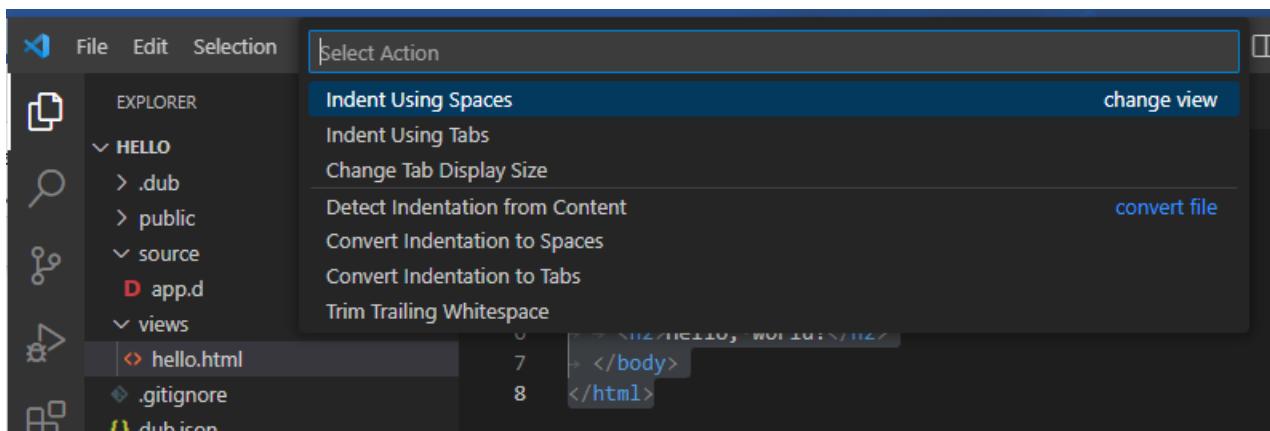
with the following contents:

```
<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 ‘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();
}
```

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 noticed, 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 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().toString();
```

Then edit source\app.d to use a router

```
import vibe.vibe;

void main()
{
    auto settings = new HTTPSServerSettings;
    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/"));
```

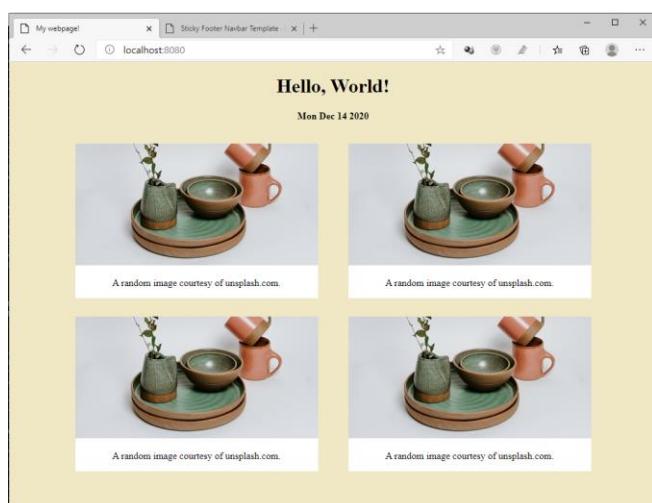
means for all other files not indicated by any route, treat them as static files and look for them inside the public/ folder.

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.

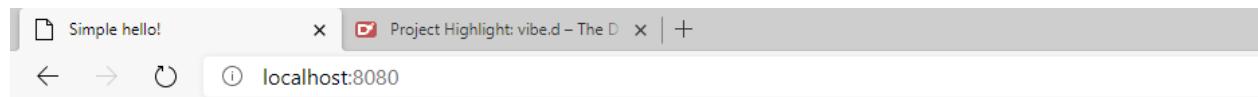
```
import vibe.vibe;

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 Jade templates (<http://jade-lang.com/>).

Here are some of the basic rules:

The 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 this

Plain text may contain HTML code
| Like this.

Nesting of elements is done by adding indentation

div Like this
 span this 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

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

Welcome to our not so simple site!

 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.

Copyright 2022 Notsosimple Company

Now we have shown that the HTML page is working.

Let's convert it into a Diet template. Save notsimplehello.html as notsimplehello.dt with the following contents:

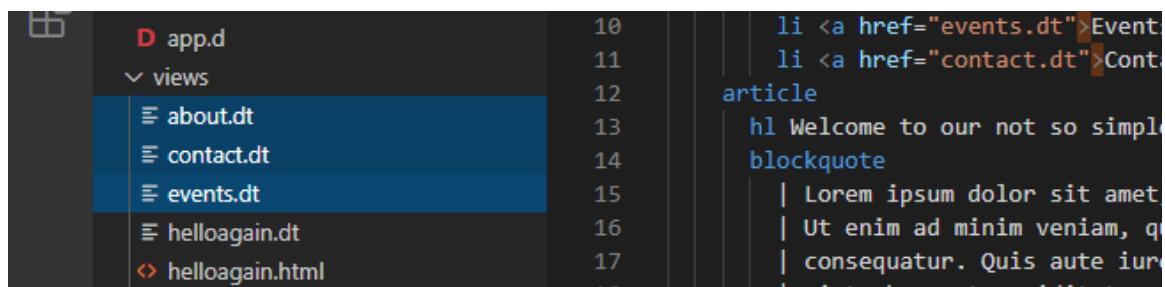
```

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 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 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, a 'views' folder containing 'about.dt', 'contact.dt', 'events.dt', 'hellogain.dt', and 'hellogain.html'. The 'events.dt' file is currently selected and its content is visible in the main editor area. The content of 'events.dt' is a partial template starting with 'li Events' followed by an 'article' section with an 'h1' header and a 'blockquote' containing placeholder text.

```

D app.d
  views
    about.dt
    contact.dt
    events.dt
    hellogain.dt
    hellogain.html

```

```

10   li <a href="events.dt">Events</a>
11   li <a href="contact.dt">Contact us</a>
12
13 article
14   h1 Welcome to our not so simple site!
15   blockquote
16     | Lorem ipsum dolor sit amet
17     | Ut enim ad minim veniam, quis nostrum exercitationem ullam
18     | corporis suscipit laboriosam, nisi ut aliquid ex ea commodi
19     | consequatur. Quis aute iure reprehenderit in voluptate velit
20     | esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
21     | obcaecat cupiditat non proident, sunt in culpa qui officia
22     | 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;

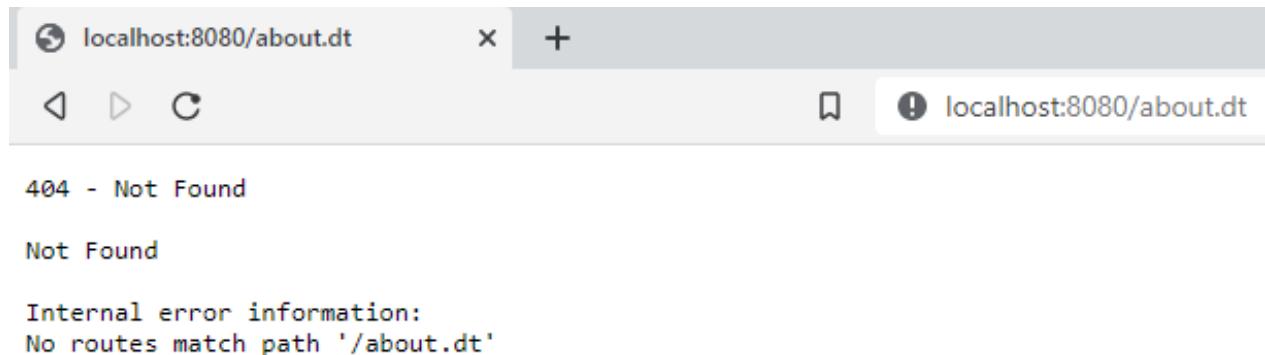
void main()
{
  auto settings = new HTTPServerSettings;
  settings.port = 8080;
  settings.bindAddresses = [":1", "127.0.0.1"];

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

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

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 main content area displays the heading "Welcome to our not so simple site" and a navigation menu with links to Home, About us, Events, and Contact us. Below the menu, a message "This is the about us page!" is circled in red. At the bottom, it says "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 main content area displays the heading "Welcome to our not so simple site" and a navigation menu with links to Home, About us, Events, and Contact us. Below the menu, a message "This is the events page!" is circled in red. At the bottom, it says "Copyright 2022 Notsosimple Company".

And the Contact us page:

A screenshot of a web browser window titled "Demo site". The address bar shows "localhost:8080/contact". The main content area displays the heading "Welcome to our not so simple site" and a navigation menu with links to Home, About us, Events, and Contact us. Below the menu, a message "This is the contact us page!" is circled in red. At the bottom, it says "Copyright 2022 Notsosimple Company".

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");
```

```

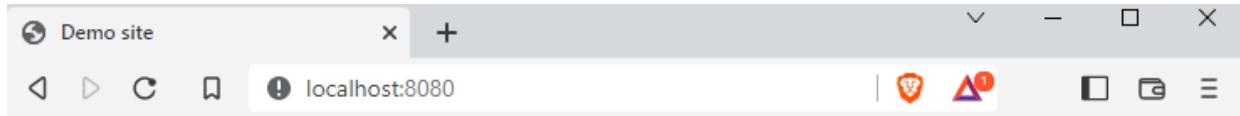
router.get("/about", staticTemplate!"about.dt");
router.get("/contact", staticTemplate!"contact.dt");
router.get("/events", staticTemplate!"events.dt");
router.get("/helloagain", &helloAgain);

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

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

```

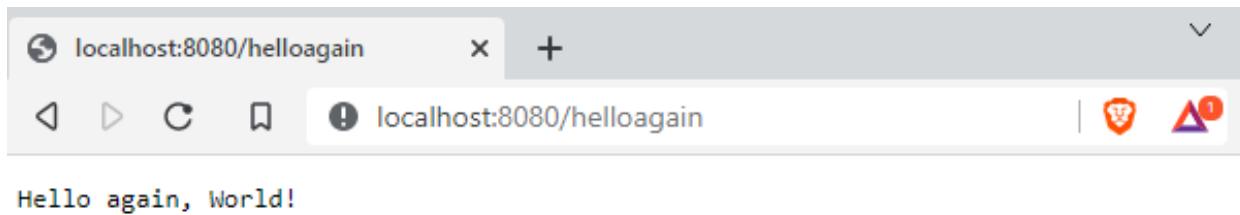
Compile and run then refresh your browser to localhost:8080/



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-powerful tool called copy-and-paste, doing manual changes to different files is still 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 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.
```

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

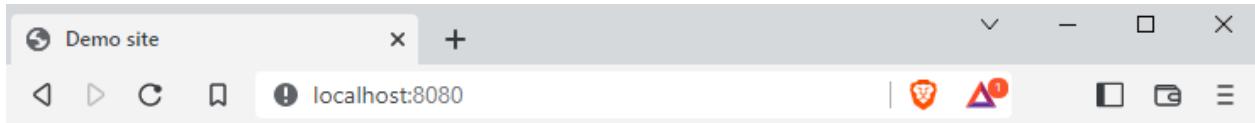
```
extends mainlayout  
block content  
    h1 This is the Contact us page!
```

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

Compile and run and refresh your browser. If you find that nothing has changed in the browser, you did good.



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.

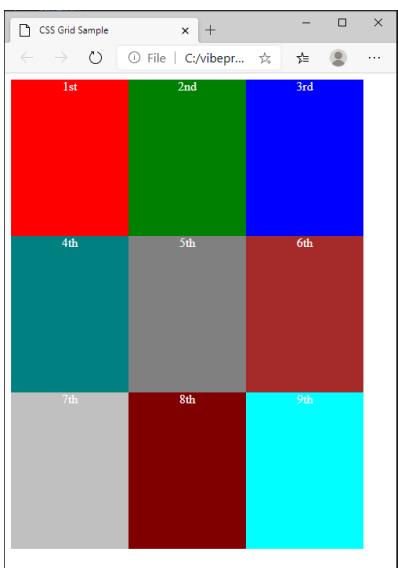
The line

```
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:

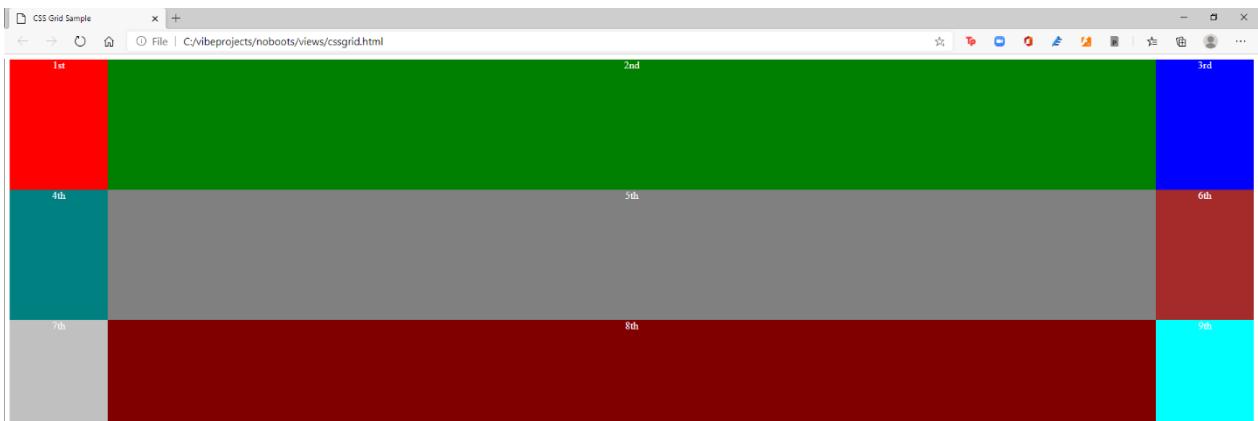
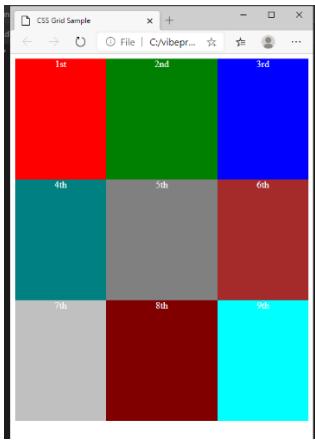


You see that when you resize the browser, the blocks remain the same sizes.

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;
}
```

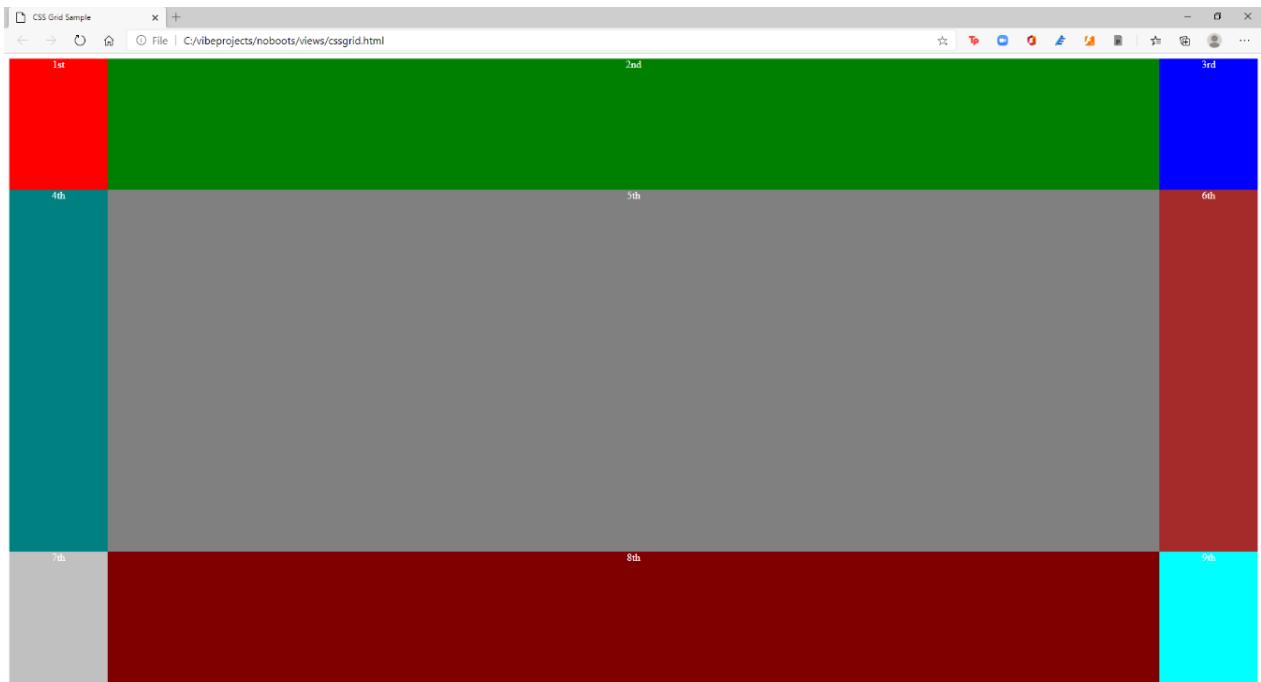
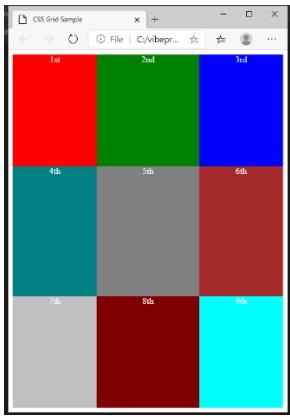
Refresh the browser. This time, when you expand or contract the browser, the middle column resizes automatically.



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 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: 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. The title bar says "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 main content area with the heading "The Lorem Ipsum Company" and a large block of placeholder text.

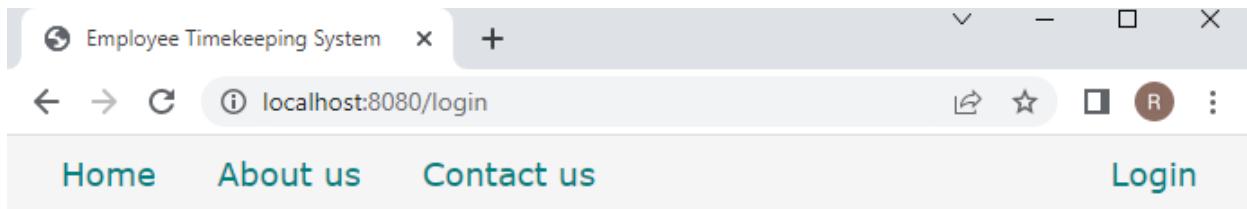
The Lorem Ipsum Company

Placeholder text (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.)

A screenshot of a web browser window. The title bar says "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 main content area with the heading "The About Us page" and the text "This is the About Us page."

Copyright © The Lorem Ipsum Company 2023

A screenshot of a web browser window. The title bar says "Employee Timekeeping System". The address bar shows "localhost:8080/contact". The page content includes a navigation bar with "Home", "About us", "Contact us", and "Login" links. Below the navigation bar is the main content area with the heading "The Contact Us page" and the text "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 using Bootstrap.

How about a modal dialogue without JavaScript?

Adding a modal dialogue to the menu

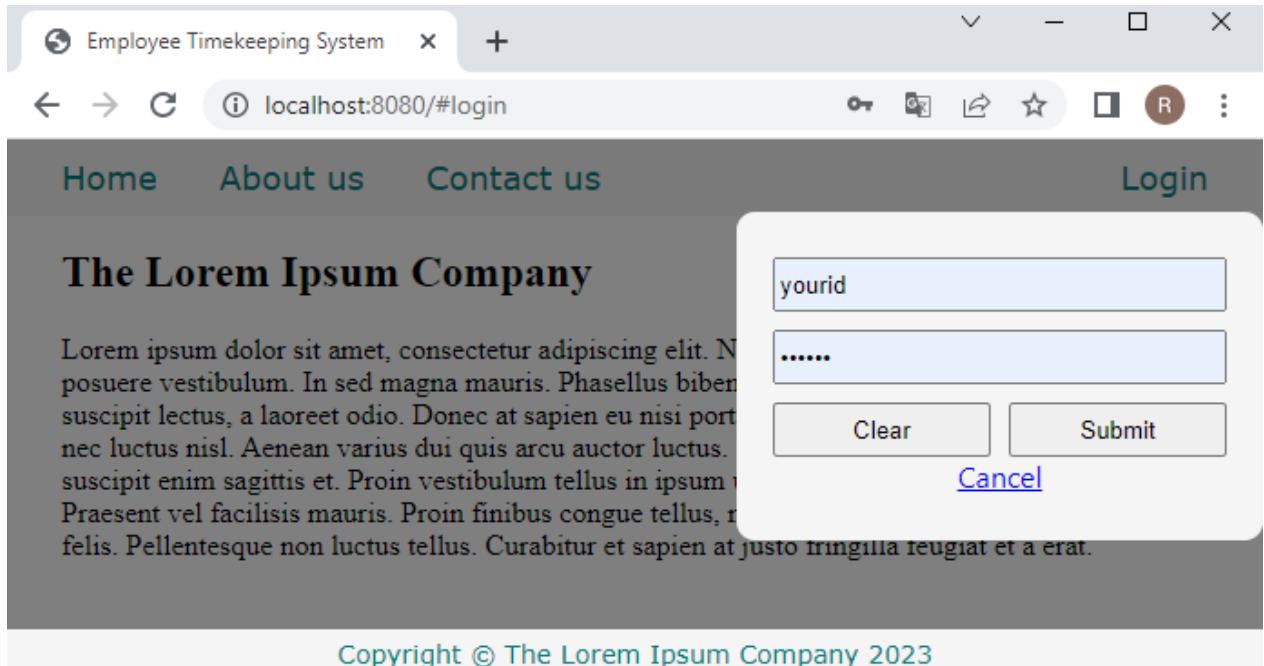
In CSS3, modal dialogues are actually part of the page but is hidden, then becomes visible and takes the focus after an event, like a click.

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:



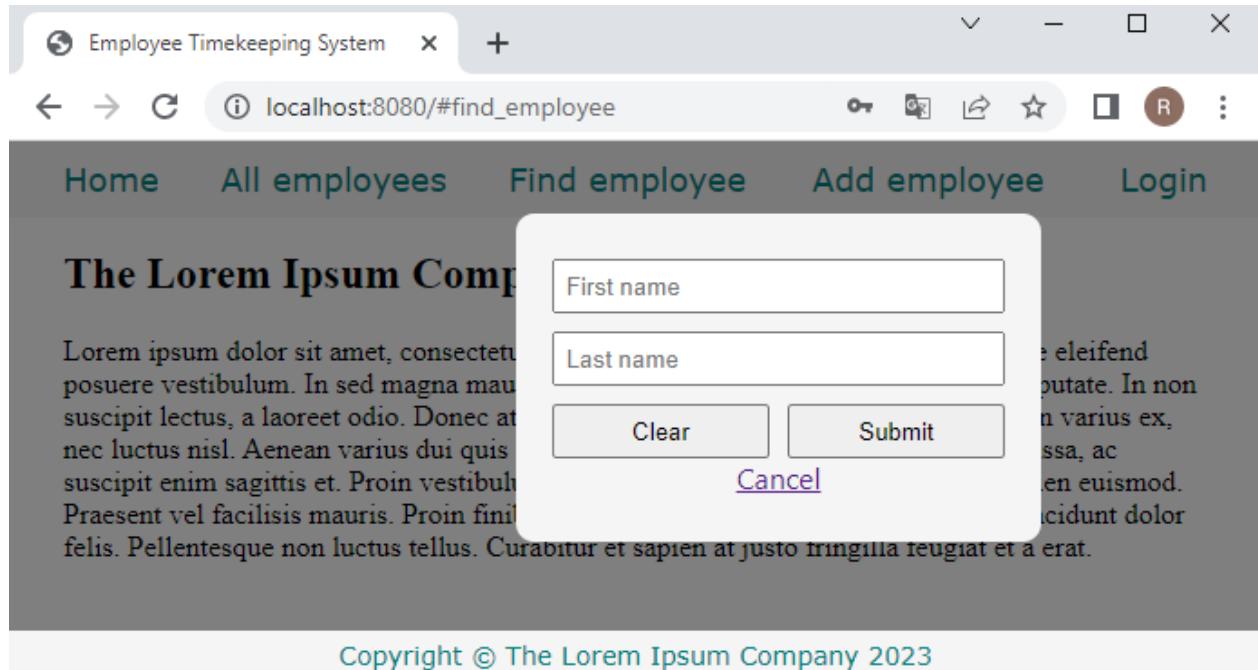
Let us edit the menu to make it more relevant to our final project.

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
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
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="Submit")
a.close(href="#close") Cancel
```

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



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

Time to talk about the web framework.

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 does not have this limitation.

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 © 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 in no_bs\views\ into the current project empapp\views so we don't have to write 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 already available to us if we need them 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 entities would like to disagree. Most sysadmins will also disagree since they are convinced 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.

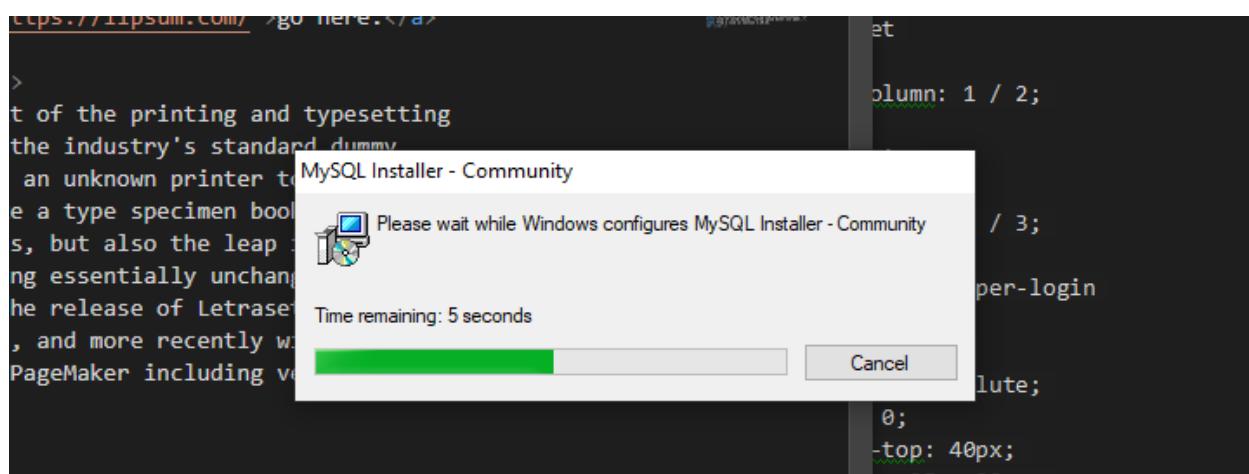
Download the MySQL community server here:

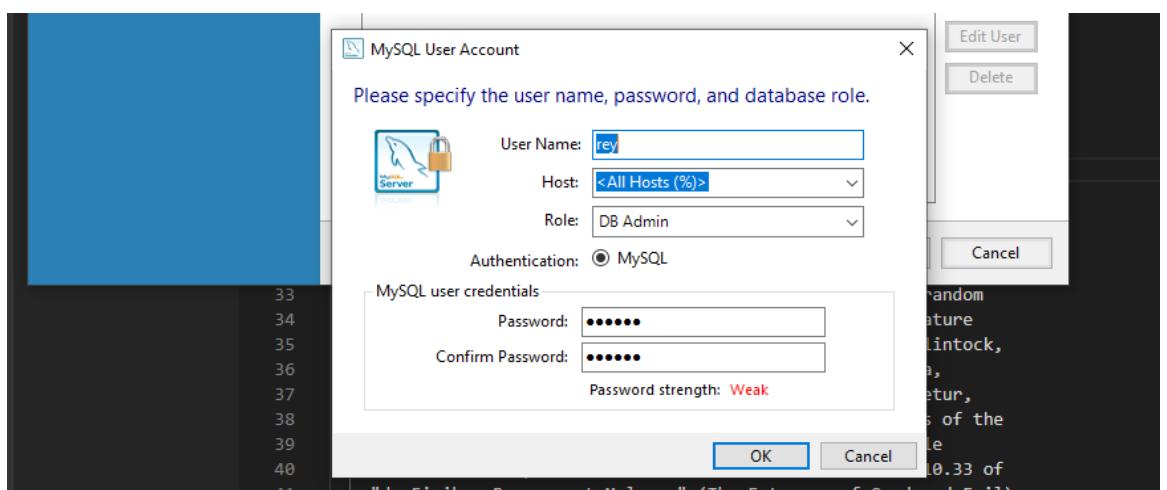
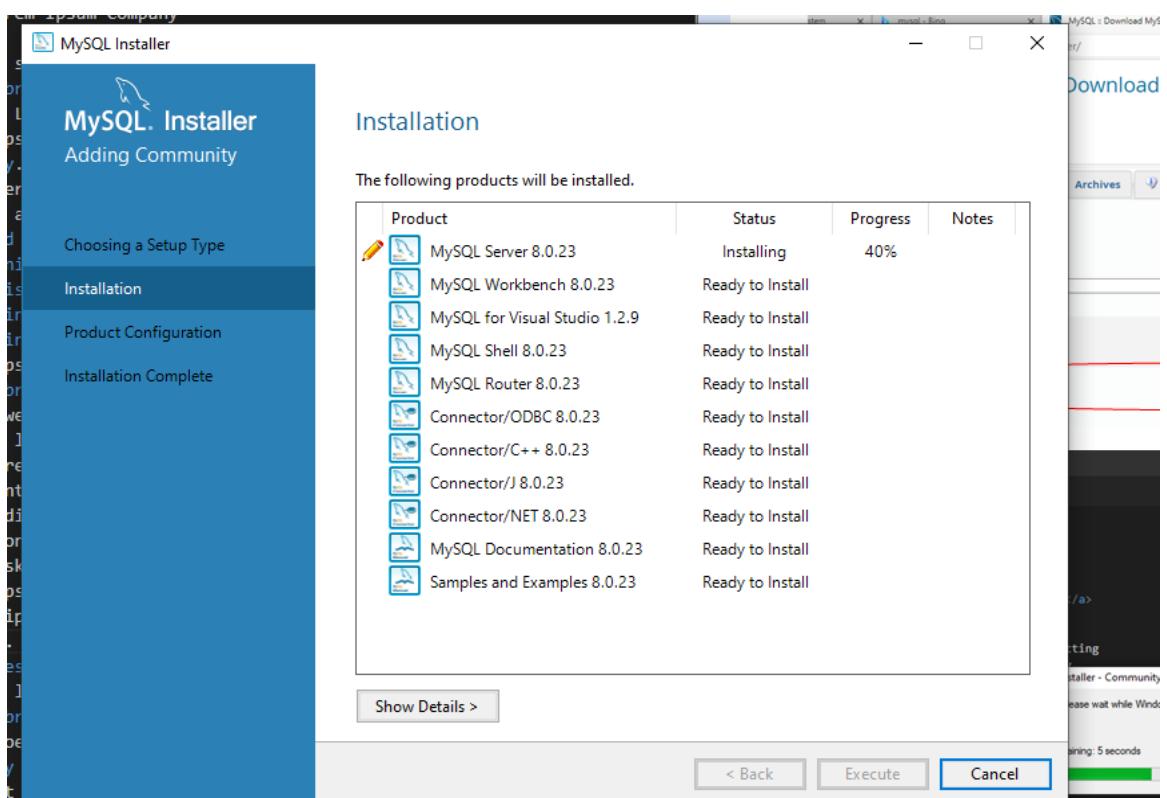
[MySQL :: Download MySQL Installer](https://dev.mysql.com/downloads/installer/)

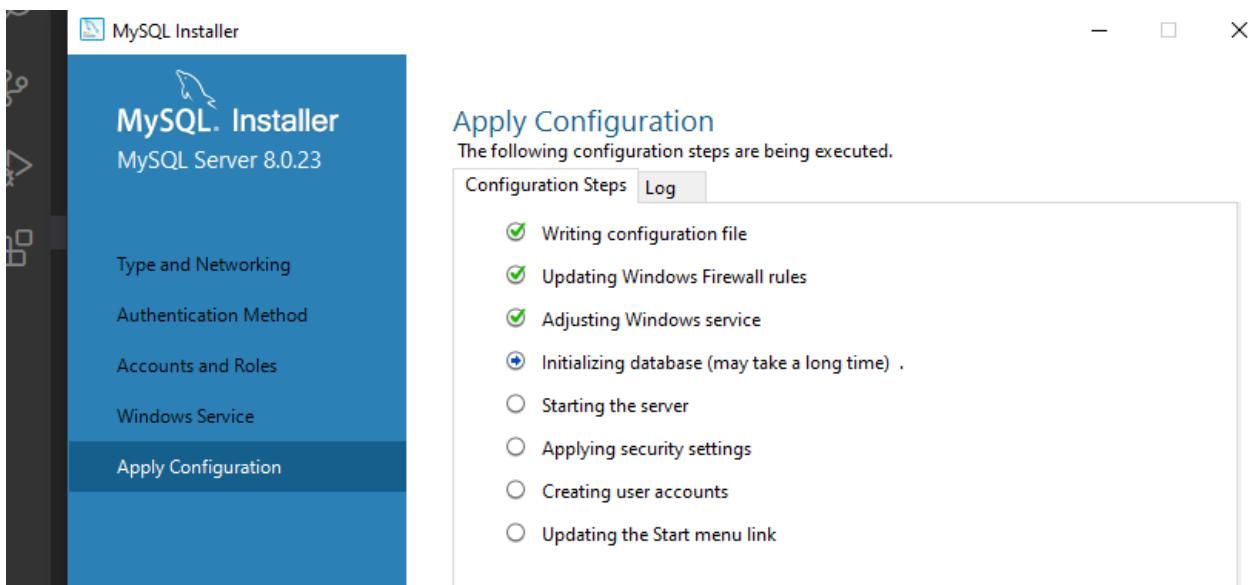
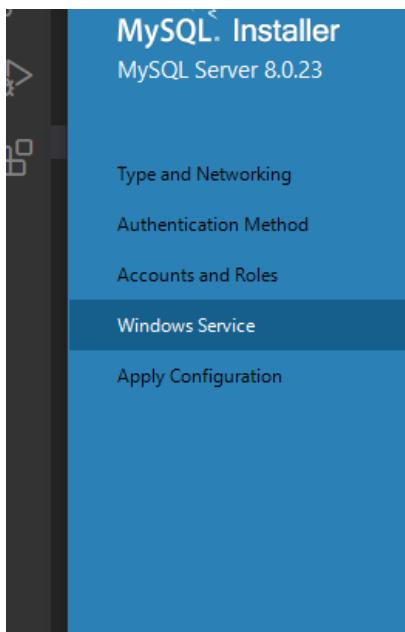
The screenshot shows the MySQL Community Downloads page. At the top, there are tabs for "General Availability (GA) Releases" (which is selected), "Archives", and "��". Below this, the title "MySQL Installer 8.0.23" is displayed. A dropdown menu "Select Operating System:" is set to "Microsoft Windows". To the right, a link "Looking for previous GA versions?" is visible. The main content area lists two download options for "Windows (x86, 32-bit), MSI Installer":

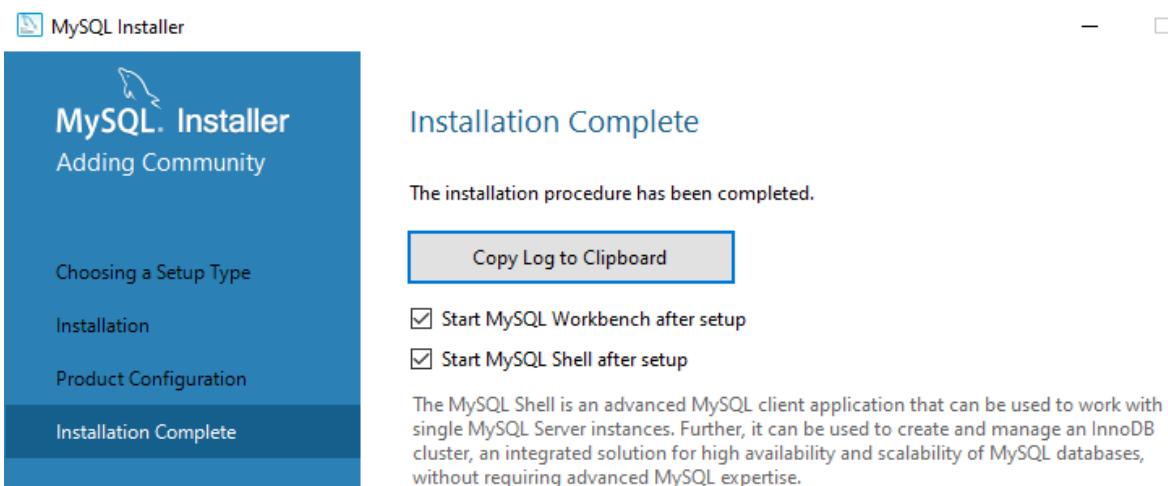
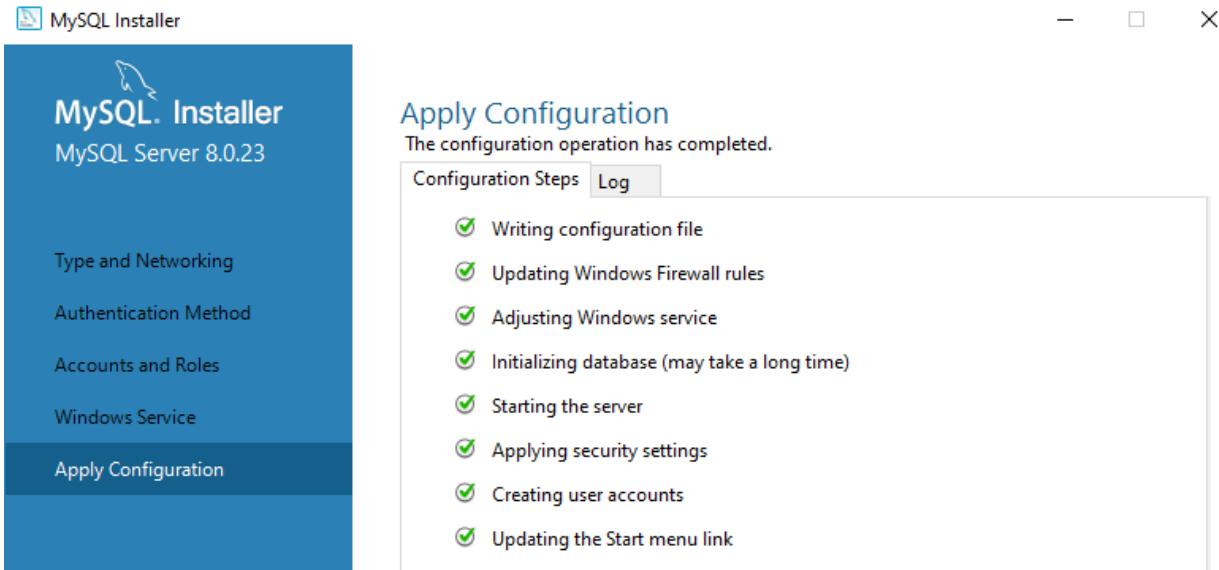
File	Version	Size	Action
mysql-installer-web-community-8.0.23.0.msi	8.0.23	2.4M	Download
mysql-installer-community-8.0.23.0.msi	8.0.23	422.4M	Download

The second row, containing the "mysql-installer-community-8.0.23.0.msi" file, is circled with a red oval.









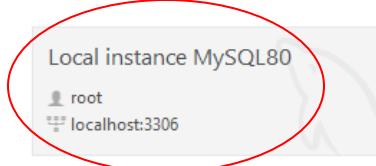
Welcome to MySQL

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 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.

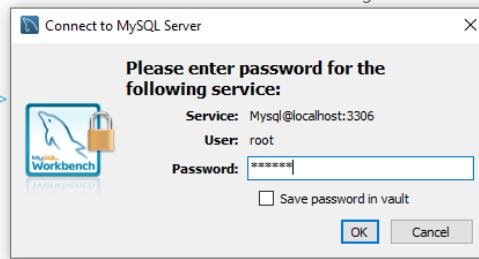
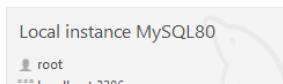
Welcome to MySQL Workbench

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

[Browse Documentation >](#)

[Ask on the Forums >](#)

MySQL Connections



 Filter connections

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
deprt	- department name, string
paygd	- salary grade, string
email	- email address, will also be used as username, string
pword	- password, string
fname	- firtstname, 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:

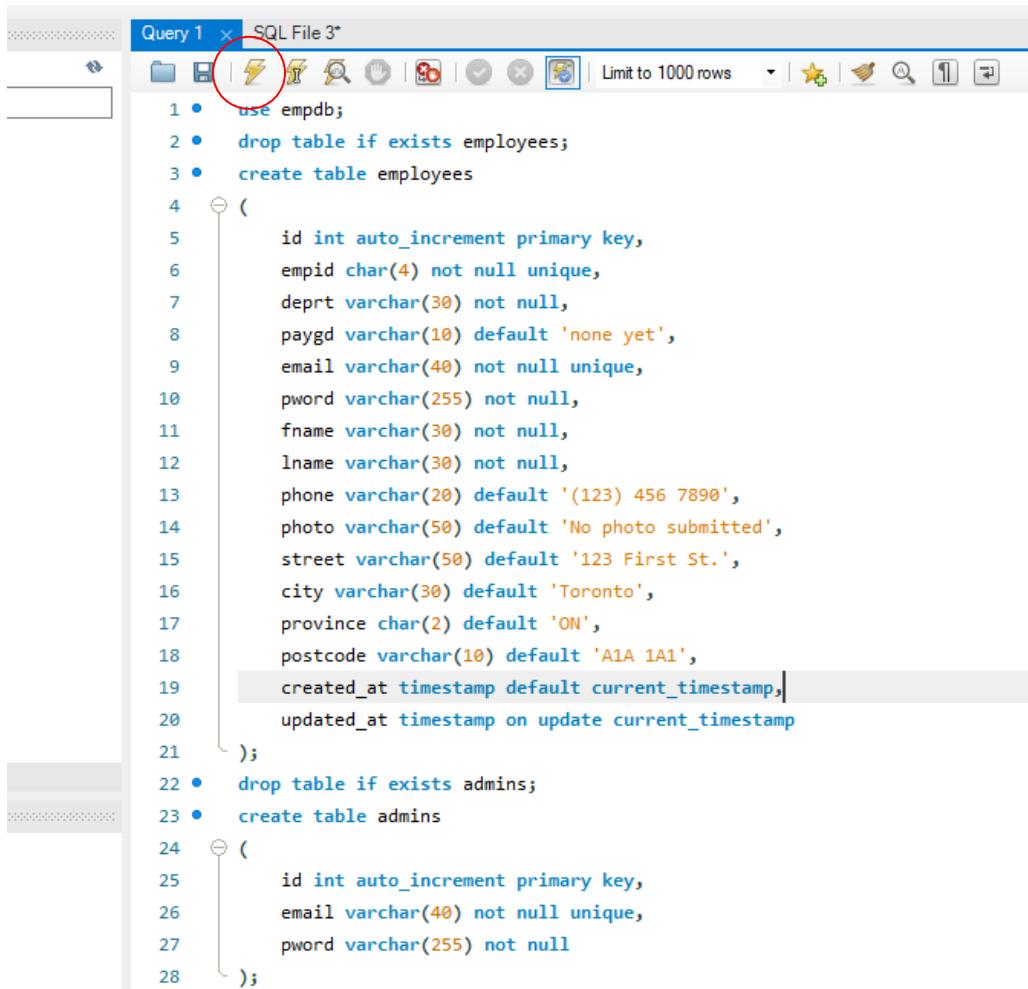
id	- the row id, the primary key incremented automatically, integer
email	- email address, which will serve as username, string
password	- password, string

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 × 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 (I wanted to use 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);
```

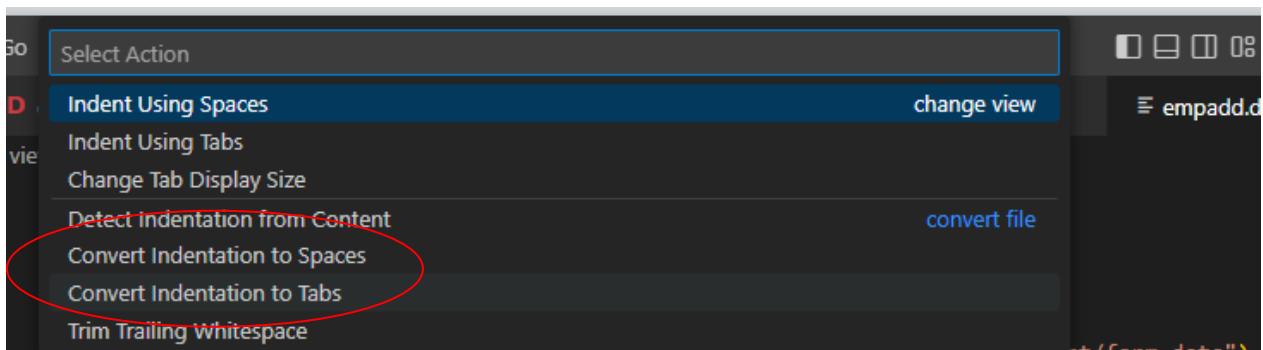
render!() has three arguments (or parameters, whichever you prefer), one string and three arrays, so we have to enclose them in parentheses.

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.



We indicated that we are going to use an EmployeeModel class which we haven't created yet, so let's do that next.

The EmployeeModel class

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 server table. It is a model that represents the employees table on the MySQL server, so we will call that class EmployeeModel.

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

Then copy the files inside empapp\views\ folder of the previous project into lorem\views to save us time.

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.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)
);
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 after the first compilation and run.**

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.dept),
```

which converts the data in e.dept (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_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")

```

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/

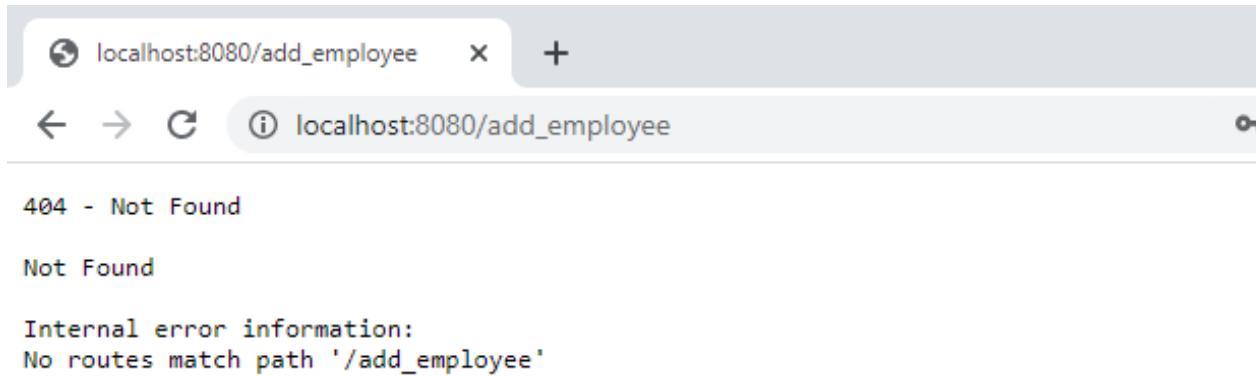
Click on the 'Add employee' link and you should see this:

The screenshot shows a web application interface. At the top, there is a navigation bar with links: Home, All employees, Find employee, Add employee, and Login. Below the navigation bar, the main content area has a title 'New employee details'. The form contains the following fields:

Department	Management and Admin
Salary grade	A100
Email address	email@company.com
Password	password
First name	First name
Last name	Last name
Phone	Phone number
Street address (no city)	Street address
City	City
Province	Alberta
Postal code	A1A 1A1
ID Picture	<input type="button" value="Choose File"/> No file chosen

Below the form, there are two buttons: 'Clear the form' and 'Submit'.

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



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

Let us add the method postAddEmployee() 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 HTTPSRequest 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 the database is hacked and 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\dm2\windows\bin\..\..\src\phobos\std\exception.d(516)
0x00007FF611495A29 in std.exception.enforce!().enforce!bool.enforce at
C:\D\dm2\windows\bin\..\..\src\phobos\std\exception.d(437)
0x00007FF6114FF20A in
vibe.data.serialization.deserializeValueImpl!(vibe.data bson.BsonSerializer,
DefaultPolicy).deserializeValueDeduced!(empmongo.Employee).deserializeValueDed-
uced at C:\D\dm2\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).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)
0x00007FF6114FEE1 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\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\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.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.

The screenshot shows a web browser window titled "Employee Timekeeping System". The address bar indicates the URL is "localhost:8080/add_employee". The page header includes links for "Home", "All employees", "Find employee", and "Add employee" (which is highlighted with a red oval), along with a "Login" link. Below the header is a form titled "New employee details". The form contains the following fields:

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="file"/> GettyImages-1173175729-1000x667.jpg

At the bottom of the form are two buttons: "Clear the form" and "Submit".

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 which corresponds to the method getAllEmployees(), which we haven't written yet, so let's write it.

Listing (or showing) 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. 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(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;
}

```

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. The Navigator pane on the left displays the database schema for 'empdb'. It shows a single table named 'employees' with columns: id, empid, dept, paygd, email, pword, fname, lname, phone, photo, street, city, province, postcode, created_at, and updated_at. The Query Editor pane on the right contains the SQL code 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 yet',
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) 456-7890',
14    photo varchar(50) default 'No photo',
15    street varchar(50) default '123 First Street',
16    city varchar(30) default 'Toronto',
17    province char(2) default 'ON',
18    postcode varchar(10) default 'A1A 1A1'

```

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 &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;
```

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.dept}

```

means display the value of e.dept 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:

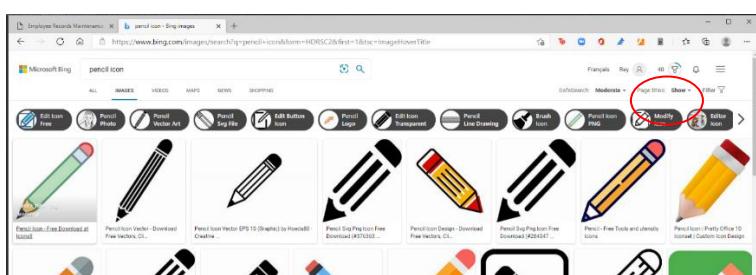


A screenshot of a web browser window titled "Employee Timekeeping System". The URL is "localhost:8080/all_employees". The page displays a table of employees with columns: First name, Last name, Department, Salary grade, Phone number, Email address, Street address, City, Province, PostCode, and Action. The data is as follows:

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\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,
                deprt, 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;
    }
}

```

And here is the full source\empcontrol.d file 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.dept}")
-foreach(dep; departments)
-if(dep == e.dept)
    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.

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

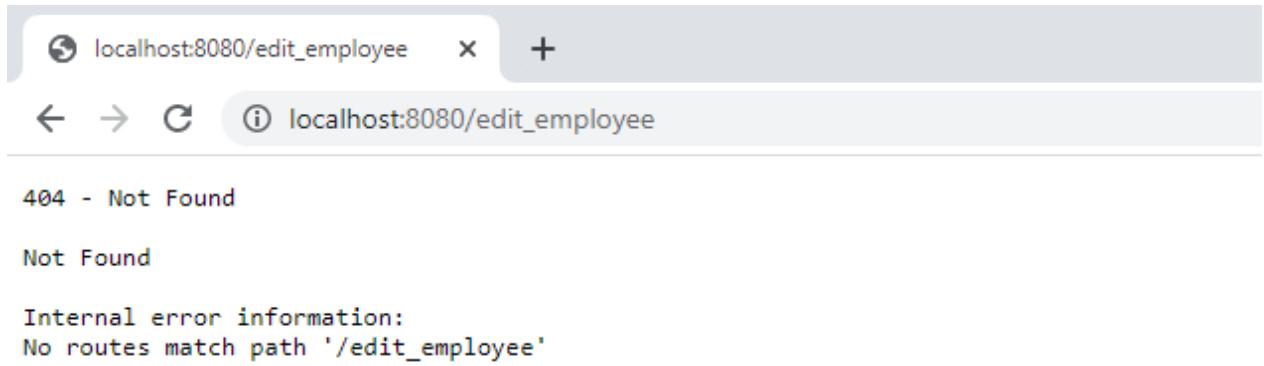
ID Picture: Choose File | No file chosen

Cancel

Submit

Great, a record was opened for editing.

But after clicking Submit, we get this error:



A screenshot of a web browser window. The address bar shows the URL `localhost:8080/edit_employee`. Below the address bar, the page content displays a 404 Not Found error. The error message includes "Internal error information" and "No routes match path '/edit_employee'".

```
404 - Not Found

Not Found

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

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

Add this method at the end of 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=?,"
```

```

deprt=?, 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(pstmt);
}

```

The SQL statement says ‘look for the record with the same value in the id field, 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	0002
Department	Accounting and Finance
Salary grade	A300
Email address	gal@gadot.com
Password	*****
First name	Gal
Last name	Gadot
Phone	0987654321
Street address (no city)	102 First St.
City	Toronto
Province	Ontario
Postal code	A1A 1A2
ID Picture	
<input type="button" value="Choose File"/> No file chosen	
<input type="button" value="Cancel"/>	
<input type="button" value="Submit"/>	

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.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);
}
}
```

Time to implement the deletion of records 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:
```

```

.form-grid-field #{e.postcode}
.form-grid-label ID Picture:


Here we are displaying a page showing 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.



Home All 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

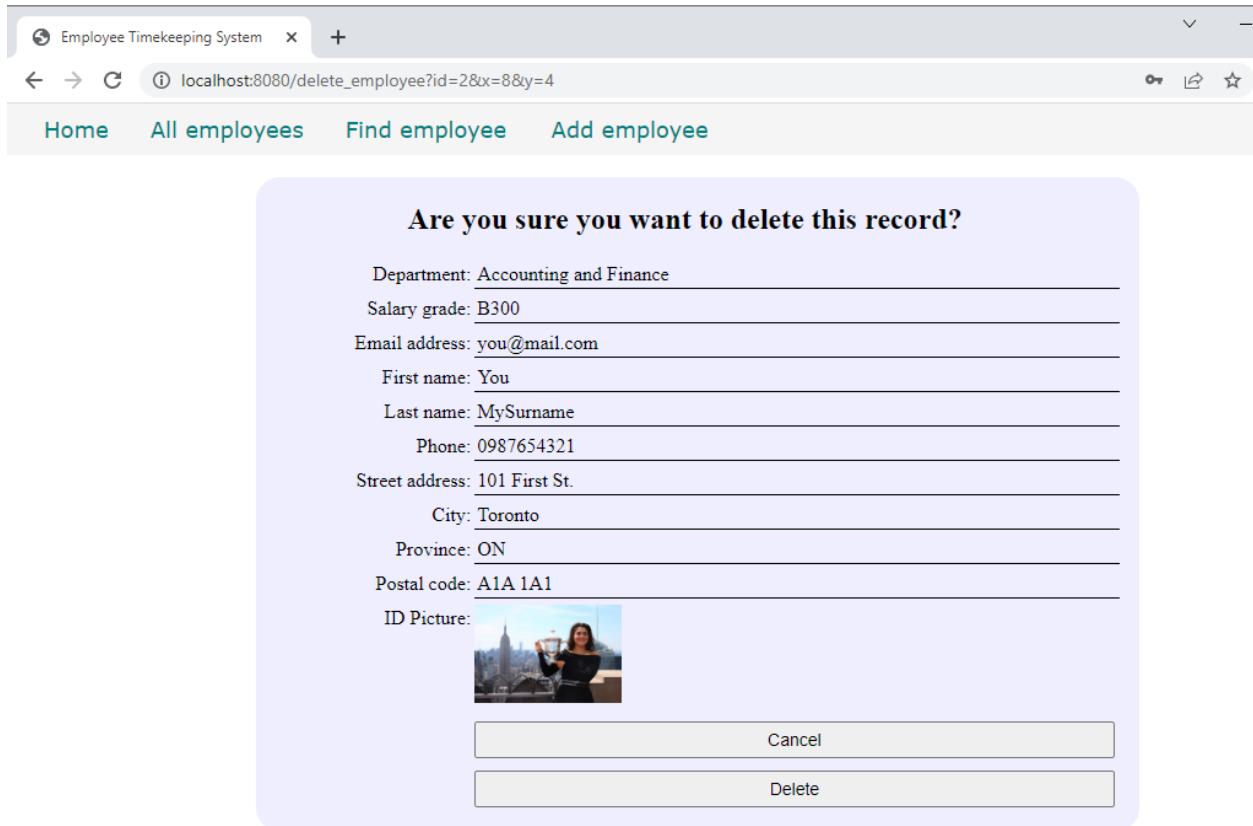


ID Picture 



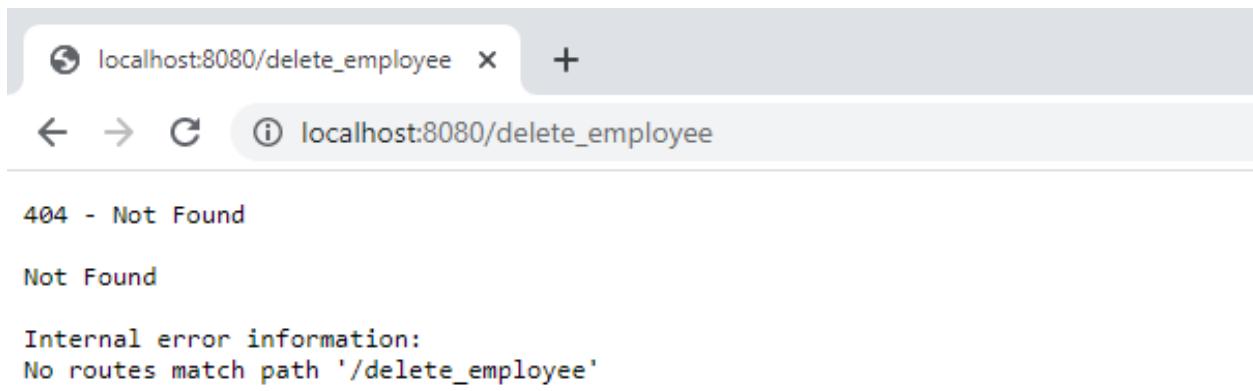
Click Cancel for now just to test if the Cancel button works. Then click the trash bin icon of another record.


```



This time click the Delete button.

And you get this error:



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",
    "G100", "G200", "G300", "G400",
    "H100", "H200", "H300", "H400",
    "I100", "I200", "I300", "I400",
    "J100", "J200", "J300", "J400",
    "K100", "K200", "K300", "K400",
    "L100", "L200", "L300", "L400",
    "M100", "M200", "M300", "M400",
    "N100", "N200", "N300", "N400",
    "O100", "O200", "O300", "O400",
    "P100", "P200", "P300", "P400",
    "Q100", "Q200", "Q300", "Q400",
    "R100", "R200", "R300", "R400",
    "S100", "S200", "S300", "S400",
    "T100", "T200", "T300", "T400",
    "U100", "U200", "U300", "U400",
    "V100", "V200", "V300", "V400",
    "W100", "W200", "W300", "W400",
    "X100", "X200", "X300", "X400",
    "Y100", "Y200", "Y300", "Y400",
    "Z100", "Z200", "Z300", "Z400"
];
```

```
"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()
{
    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.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);
}

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");
}
}

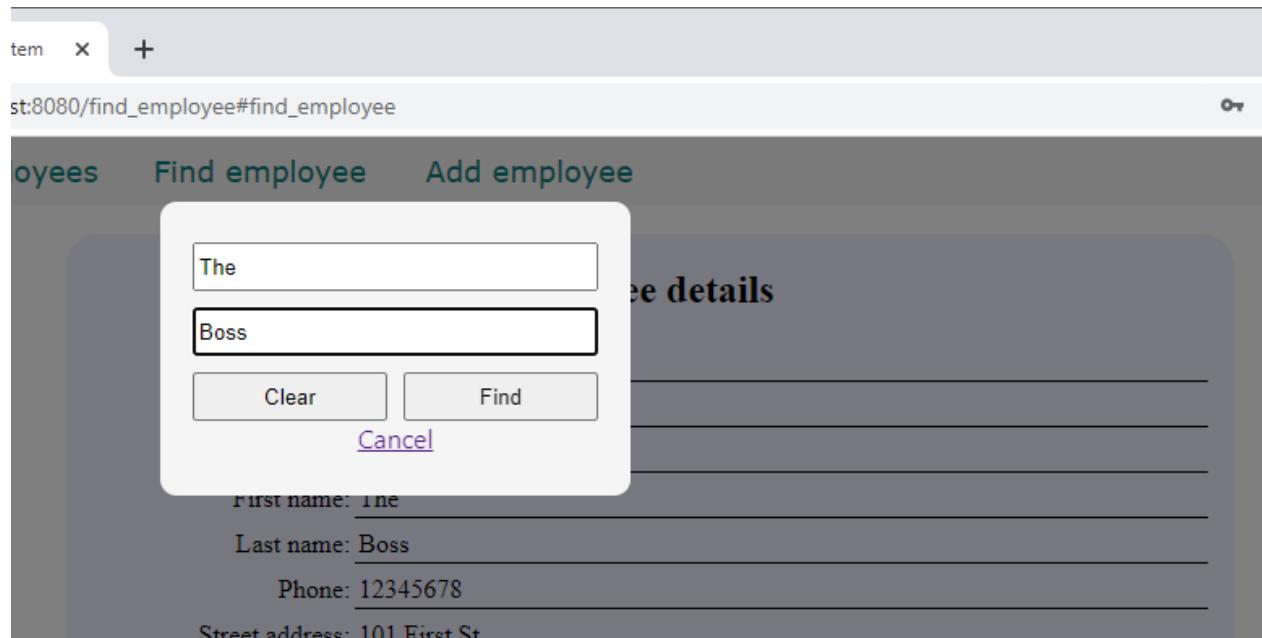
```

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

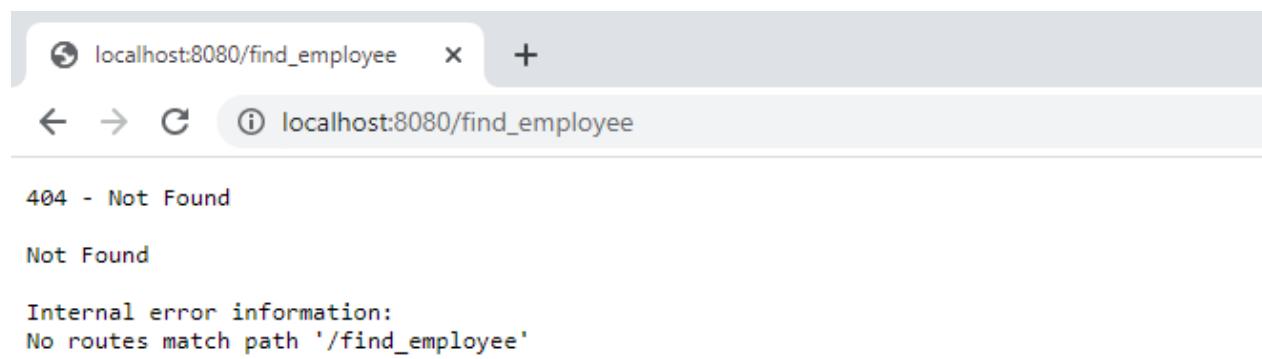
Finding an employee record by name

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.



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



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 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.prepare(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 comparing. 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:

```

```

.form-grid-field #{e.email}
.form-grid-label First name:
.form-grid-field #{e.fname}
.form-grid-label Last name:
.form-grid-field #{e.lname}
.form-grid-label Phone:
.form-grid-field #{e.phone}
.form-grid-label Street address:
.form-grid-field #{e.street}
.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:


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:     |  |


```

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.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.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 using _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 is 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 the error message is blank.

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

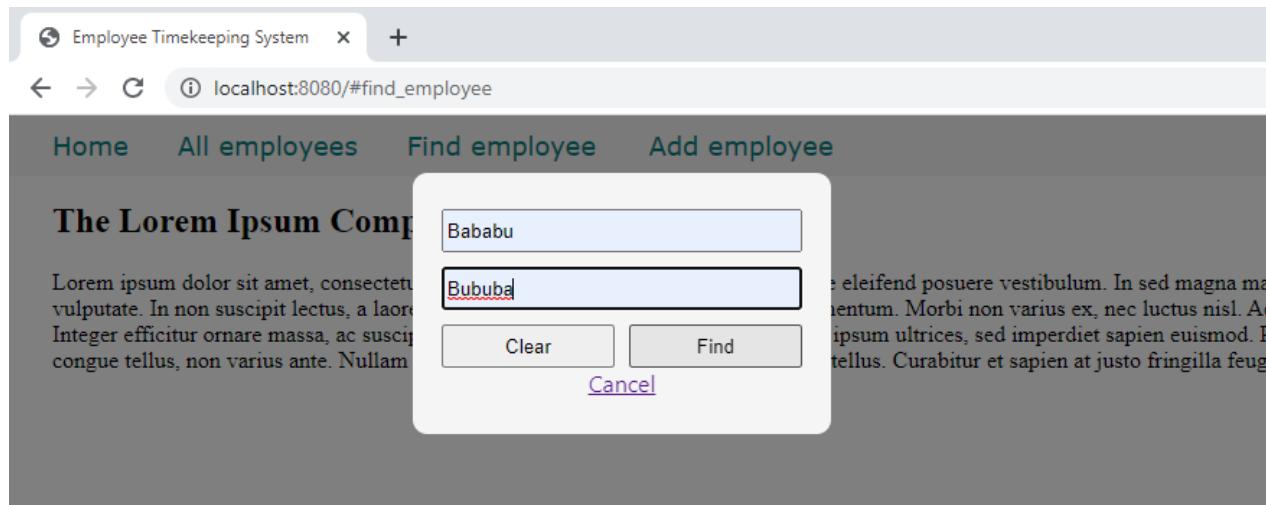
```
extends layout
block maincontent
    include csstable.dt
    -if(error)
        div.error-div
            span.error-message #{error}
    div.table-wrapper
        table
            tr
                th Employee Id
                th First name
                th Last name
                th Department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

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 screenshot of a web browser window showing a search result table. The table has a header row with columns: Employee Id, First name, Last name, Department, Phone number, Email address, Street address, City, and Province. There are three data rows: 0001 (The Boss, Management and Admin, 12345678, theboss@mail.com, 101 First St., Toronto, ON), 0002 (Gal Gadot, Accounting and Finance, 0987654321, gal@gadot.com, 102 First St., Toronto, ON), and 0003 (Eugene Bou, Accounting and Finance, 31247425645, ebouchard@mail.com, 103 First St., Toronto, ON). A red oval highlights the message "Bababu Babuba not found." located above the table.

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#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")
```

```



```

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.

Let's talk about logging in, authentication and authorization.

Logging in, authentication and authorization

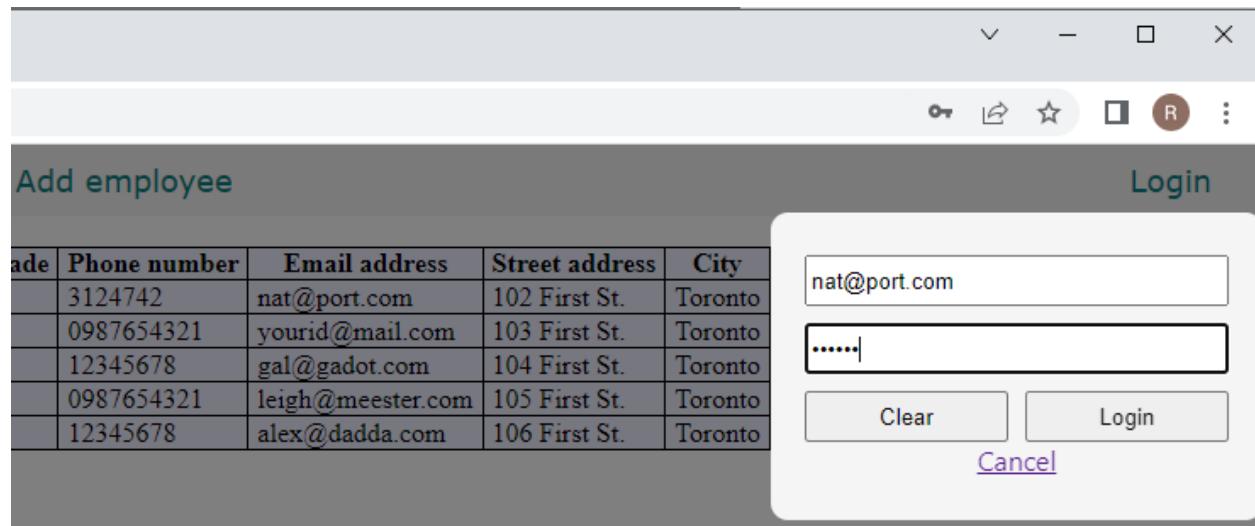
We need to restrict access to sensitive parts of our site so only authorized users (admins) can make changes to sensitive data. We need a login system to authenticate and to authorize users.

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.

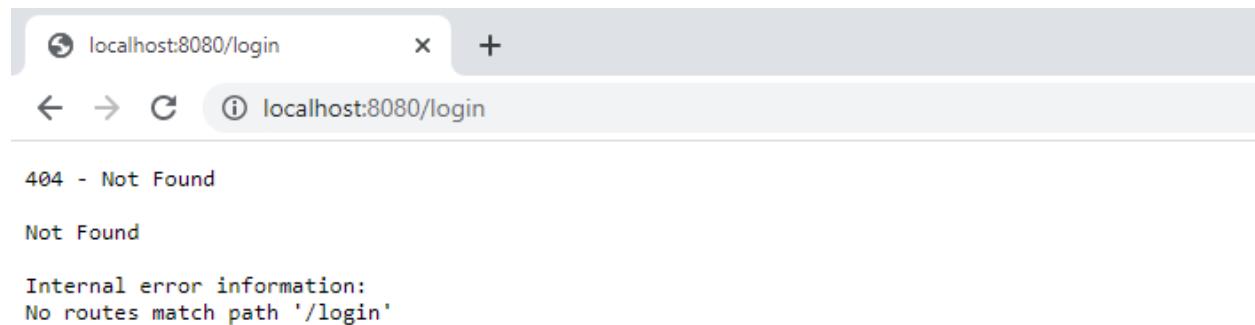
In a commercial company, only a limited number of people are supposed to access sensitive data such as employee records.

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(realm, 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.

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

Login

 [Cancel](#)

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

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

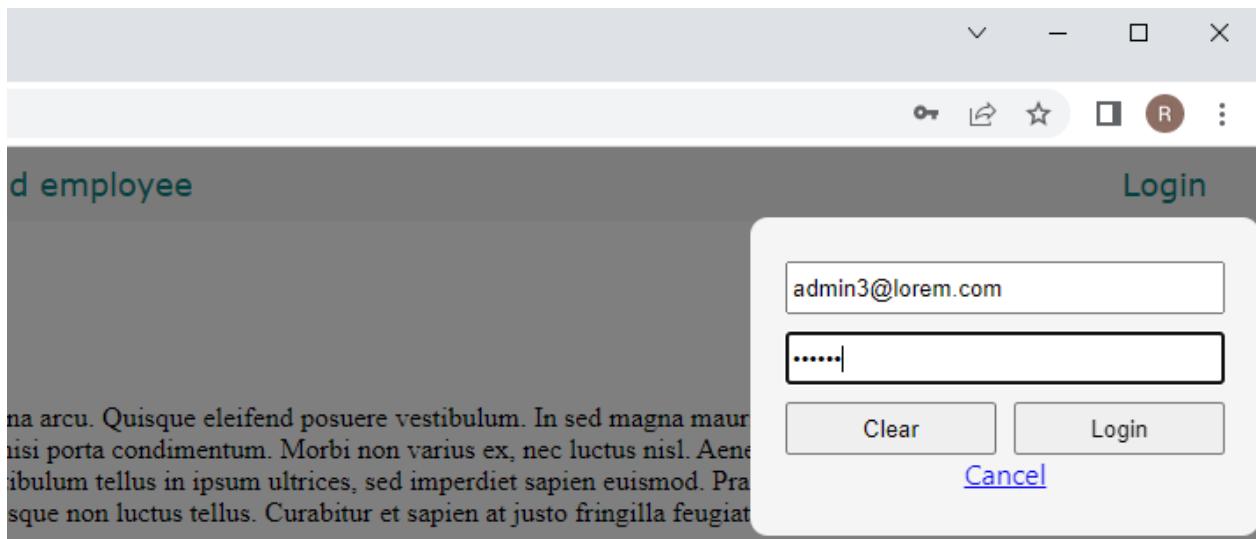
- Title Bar:** Employee Timekeeping System
- Address Bar:** localhost:8080/login
- Navigation Links:** Home, All employees, Find employee, Add employee
- Main Content:**

The Lorem Ipsum Company

Email and password combination not found.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. 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. So 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 empmode;

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 Lorem Ipsum Company

Email and password combination not found.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam nec urna arcu. Quisque eleifend posu
vulputate. In non suscipit lectus, a laoreet odio. Donec at sapien eu nisi porta condimentum. Morbi
Integer efficitur ornare massa, ac suscipit enim sagittis et. Proin vestibulum tellus in ipsum ultrices
congue tellus, non varius ante. Nullam tincidunt dolor felis. Pellentesque non luctus tellus. Curabit

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 != 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=?,
        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]);
}

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 m_user variable

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 is available and is 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 this 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 != 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(realms, 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 HTTPSServerSettings;
    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(realms, 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(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        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]);
    }



```

```

        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;
    }
}

/* I moved this here so we will not accidentally use it.
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);
}
*/

```

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:
            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 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",
encctype="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")

```

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.

The timekeeping system

Employees clock in to start work and clock out at end of day using Bundy clocks or some kind of electronic equipment to record the workers' hours. For our purposes, we will simply let users click on a button on a web page to punch in or punch out and record the time.

To record the time, we need the

- employee id
- the date
- the time in
- the time out

Other timekeeping systems require a job (or project; companies oftentimes have several projects going on simultaneously) the employee will work on so they can keep track of how many worker hours were spent on a project, but to keep things simple, we will not require a job or project.

The tables we need are

- employees table (already present)
- timecard table (for time-in/time-out)
- timesheet table (for the summary)

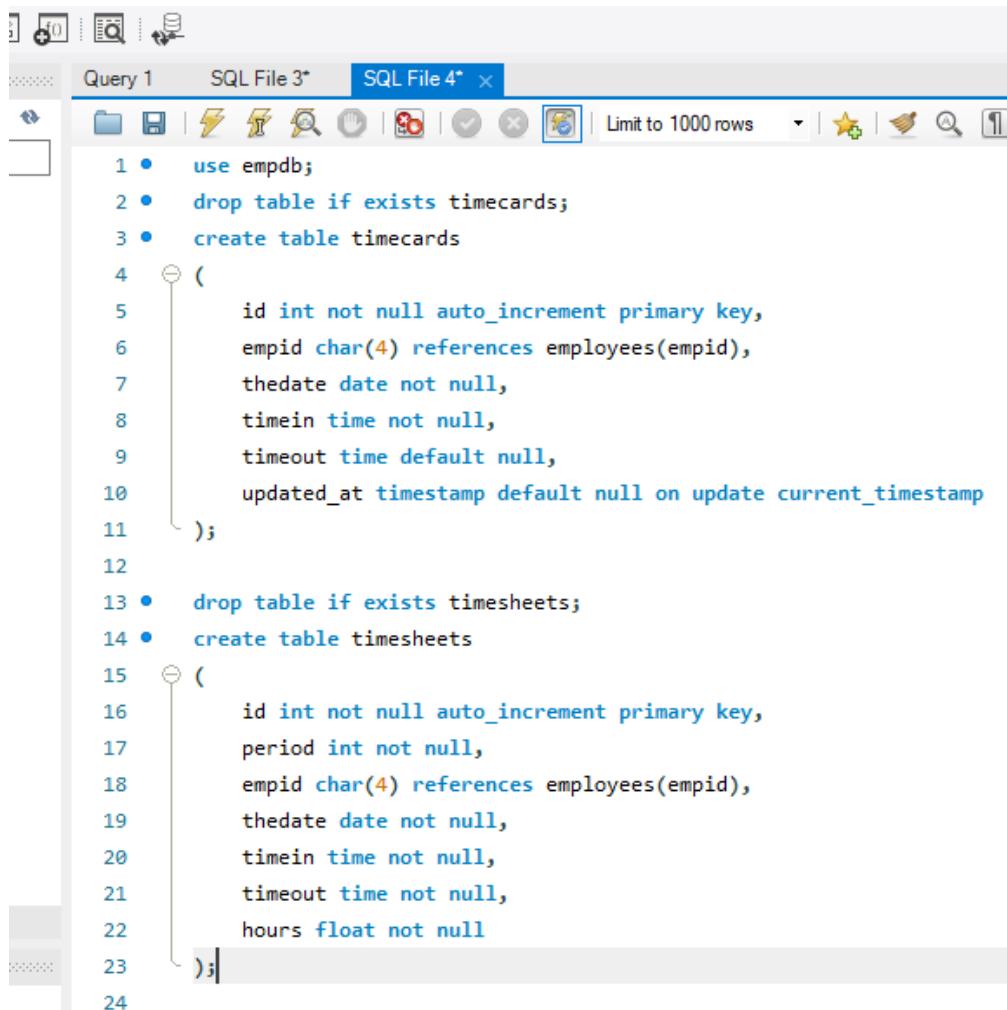
Here are the SQL statements we will use to create the timecards and the timesheets tables:

```
use empdb;
drop table if exists timecards;
create table timecards
(
    id int not null auto_increment primary key,
    empid char(4) references employees(empid),
    thedate date not null,
    timein time not null,
    timeout time default null,
    updated_at timestamp default null on update current_timestamp
);

drop table if exists timesheets;
create table timesheets
(
    id int not null auto_increment primary key,
    period int not null,
    empid char(4) references employees(empid),
    thedate date not null,
    timein time not null,
```

```
    timeout time not null,  
    hours float not null  
);
```

So here is what we will type on MySQL Workbench:



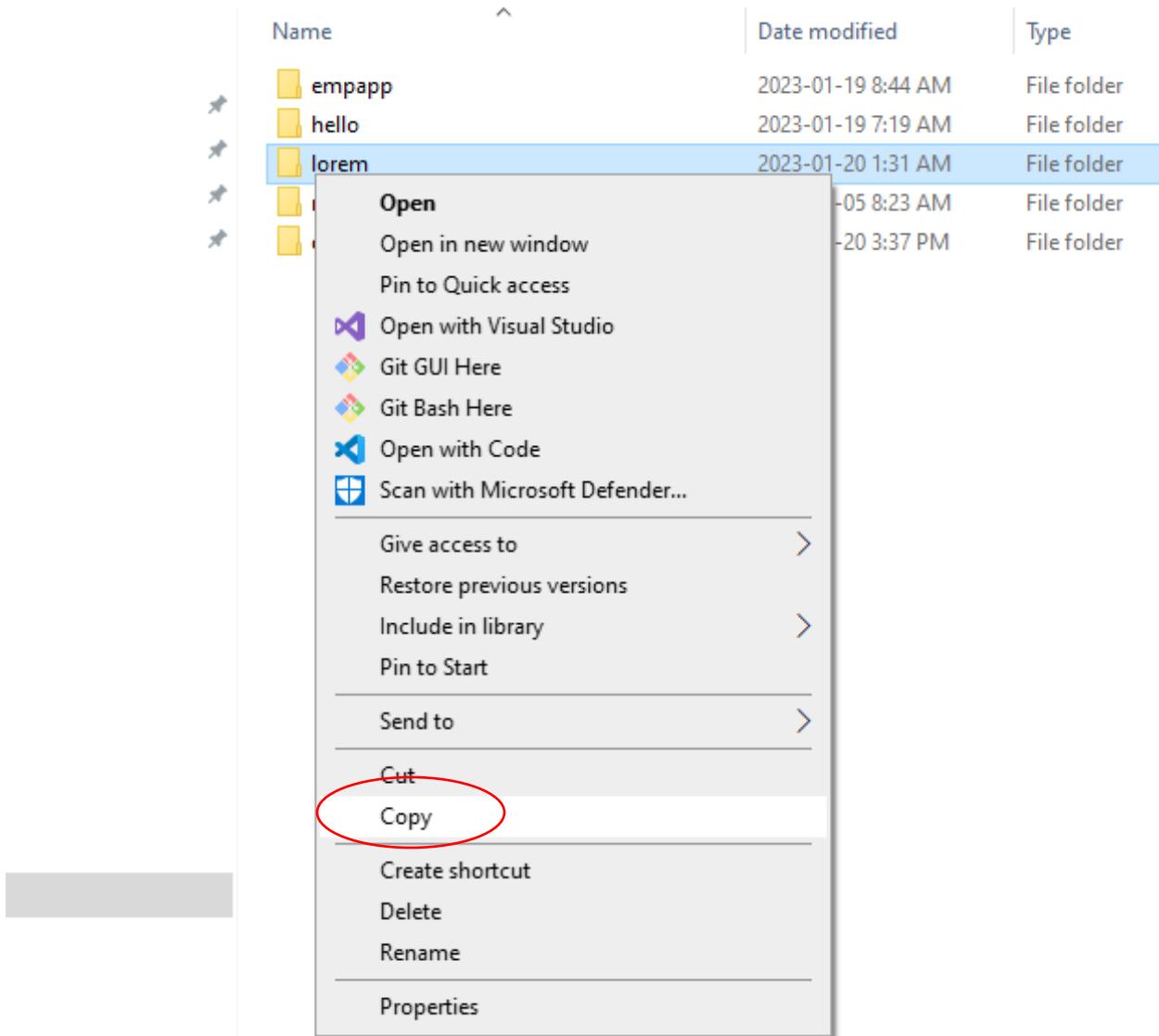
The screenshot shows the MySQL Workbench interface with three tabs: Query 1, SQL File 3*, and SQL File 4*. The SQL File 4* tab is active and contains the following SQL code:

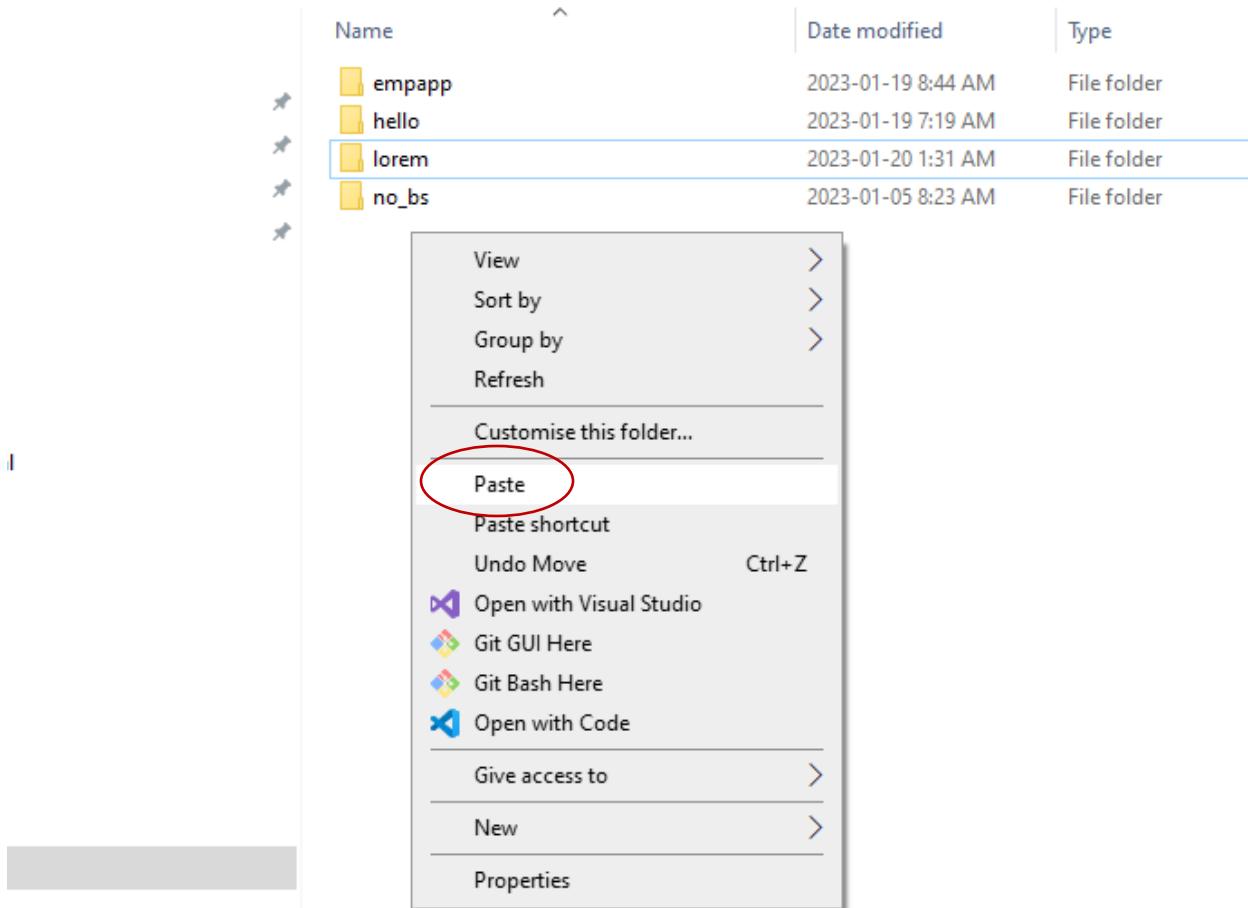
```
1 •  use empdb;  
2 •  drop table if exists timecards;  
3 •  create table timecards  
4  (    id int not null auto_increment primary key,  
5      empid char(4) references employees(empid),  
6      thedate date not null,  
7      timein time not null,  
8      timeout time default null,  
9      updated_at timestamp default null on update current_timestamp  
10     );  
11  
12  
13 •  drop table if exists timesheets;  
14 •  create table timesheets  
15  (    id int not null auto_increment primary key,  
16      period int not null,  
17      empid char(4) references employees(empid),  
18      thedate date not null,  
19      timein time not null,  
20      timeout time not null,  
21      hours float not null  
22     );  
23  
24
```

With this, we can start with the timekeeping subsystem.

Create the bundy project

Create a new project named bundy. To save us time, simply copy the whole lorem folder into a new folder called bundy in File Explorer.





Local Disk (C:) > vibeprojects >

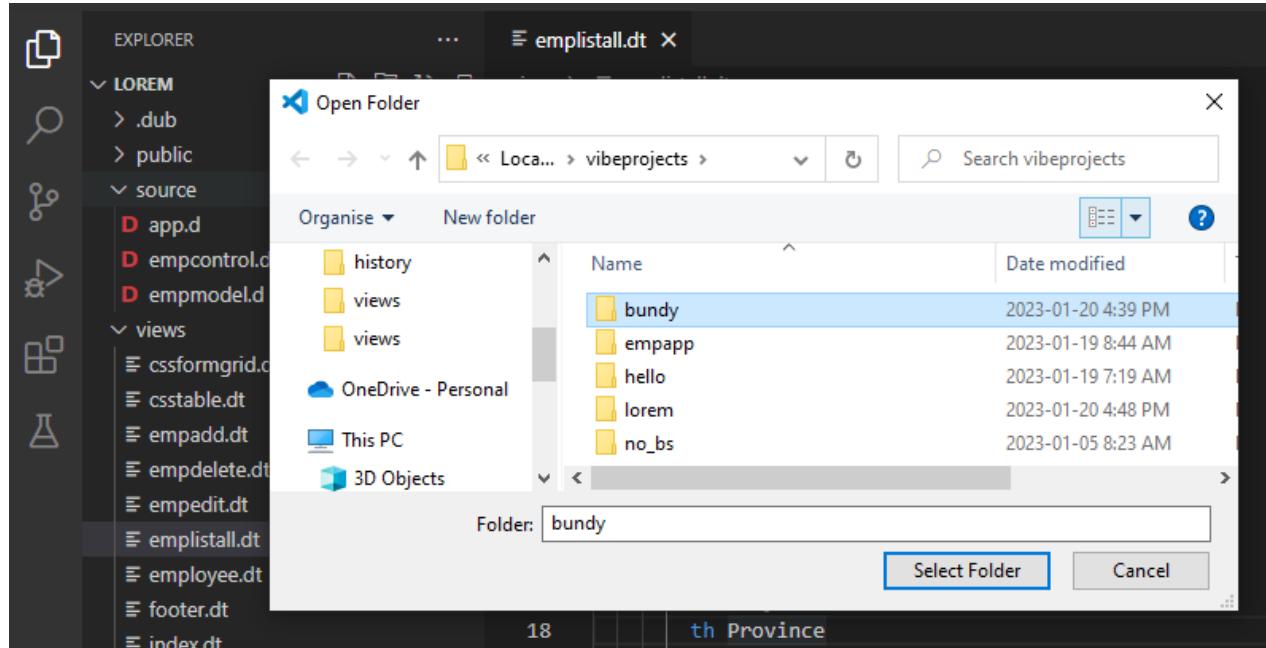
Name	Date modified	Type
empapp	2023-01-19 8:44 AM	File folder
hello	2023-01-19 7:19 AM	File folder
lorem	2023-01-20 1:31 AM	File folder
lorem - Copy	2023-01-20 4:39 PM	File folder
no_bs	2023-01-05 8:23 AM	File folder

Then simply rename lorem – Copy into bundy

al Disk (C:) > vibeprojects >

Name	Date modified	Type	Size
bundy	2023-01-20 4:39 PM	File folder	
empapp	2023-01-19 8:44 AM	File folder	
hello	2023-01-19 7:19 AM	File folder	
lorem	2023-01-20 1:31 AM	File folder	
no_bs	2023-01-05 8:23 AM	File folder	

Then open the bundy folder in VS Code.



And edit dub.json so it doesn't get confused. Rename 'lorem' to 'bundy'.

```
{
  "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": "bundy"
}
```

And compile and run to make sure everything's working.

```
c:\vibeprojects>cd bundy
```

```
c:\vibeprojects\bundy>dub
```

Now let's change the look for this project.

A new menu look

Let us have a revamped look for this project.

Edit views\layout.dt

```
doctype 5
html
  head
    title Employee Timekeeping System
    include csslayout.dt
  body
    div.container
      div.menu
        include menu.dt
      div.content
        block maincontent
      div.footer
        include footer.dt
```

Then rename views\styles.dt into views\csslayout.dt with this contents

```
:css
body
{
  margin: 0;
  padding: 0;
}
.container
{
  display: grid;
  grid-template-rows: 30px auto 30px;
  height: 100%;
  width: 100%;
  margin: 0;
  padding: 0;
}
.menu
{
  margin: 0;
  padding: 0;
}
/*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;
    font-family: Sans-serif;
}
/*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;
}
.error-div
{
    margin: 5px 0;
}
.error-message
{
    color: brown;
    font-weight: bold;
}
.center-align
```

```

{
  margin: 0 auto;
  text-align: center;
}
.right-align
{
  text-align: right;
}
.form-hidden
{
  padding: 0;
  margin: 0;
  display: inline;
}

```

Next, edit views\menu.dt.

```

include cssmenu.dt
nav
  ul
    li
      a(href="/") Home
    li
      a(href="#") Punch card
        ul
          li
            a(href="time_in") Punch in
          li
            a(href="time_out") Punch out
          li
            a(href="card_review") View cards
          li
            a(href="create_timesheet") Create timesheet
    li
      a(href="#") Employees
        ul
          li
            a(href="all_employees") Show all employees
          li
            a(href="#find_employee") Find an employee
          li
            a(href="add_employee") Add new employee
  ul.link-right
    li
      a(href="#login") Login

```

```


#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


#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


```

This template needs views\cssmenu.dt, so let's create it.

```

:css
nav
{
  margin: 0 auto;
  padding:0;
  width: 100%;
  height: auto;
  display: inline-block;
  background: teal; /*#37bc9b;*/
  font-family: Verdana, Geneva, Tahoma, sans-serif;
}
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:after { content: ' »'; } /* Change this in order to change the Dropdown symbol */
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: 200px;
}
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;
}
```

Now let's see what we did.

```
c:\vibeprojects\bundy>dub
Starting Performing "debug" build using C:\D\dm2\windows\bin\dm.exe for x86_64.
Up-to-date taggedalgebraic 0.11.22: target for configuration [library] is up to date.
Up-to-date eventcore 0.9.21: target for configuration [winapi] is up to date.
```

...

```
Compiling Diet HTML template empadd.dt...
Compiling Diet HTML template emplistall.dt...
Compiling Diet HTML template empedit.dt...
Compiling Diet HTML template empdelete.dt...
Compiling Diet HTML template employee.dt...
```

Linking bundy

Finished To force a rebuild of up-to-date targets, run again with --force

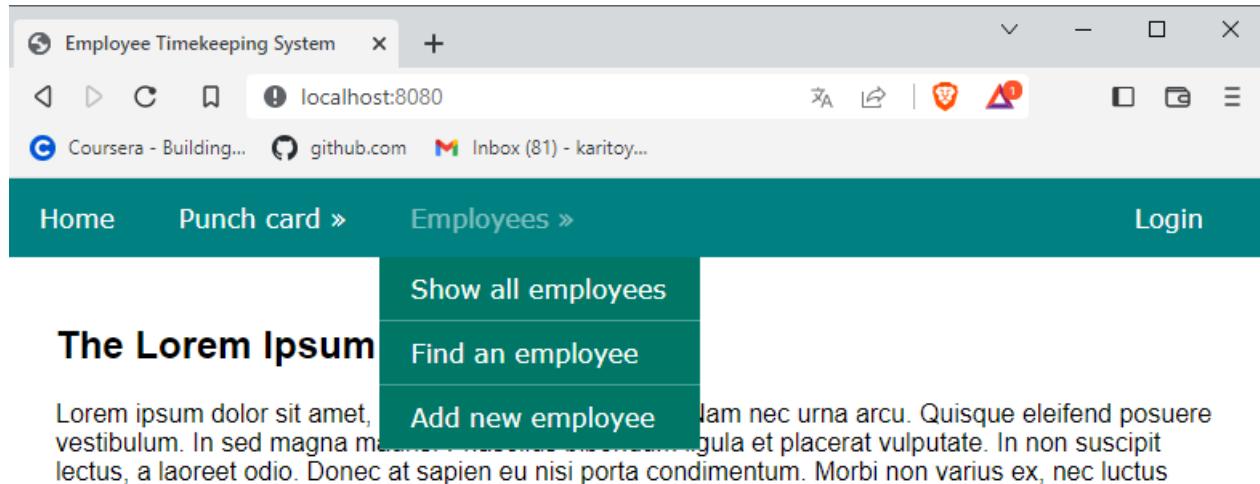
Copying files for vibe-d:tls...

Running bundy.exe

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

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

And we see this:



Okay, now let's revamp the views\emplistall.dt.

```
extends layout
block maincontent
```

```

include csstable
-if(error)
  div.error-div
    span.error-message #{error}
h2 Employees of The Lorem Ipsum Company
div.table-wrapper
  table
    tr
      th First name
      th Last name
      th Department
      th Phone number
      th Email address
      th Action
    -foreach(e; emps)
      tr
        td #{e.fname}
        td #{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.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;

```

And edit views\csstable.dt.

```

:css
.table-wrapper
{
  margin: 20px auto;
}
table
{
  padding: 10px 0;
  border-spacing: 0;
  border-collapse: collapse;
}
table td, table th

```

```

{
  margin: 0;
  padding: 2px 10px;
}
.no-border
{
  border: 0;
}
tr:nth-child(odd)
{
  background: whitesmoke;
}

```

Now let's review what we just did again. Compile and run and refresh the browser. Remember to login to view the employees.

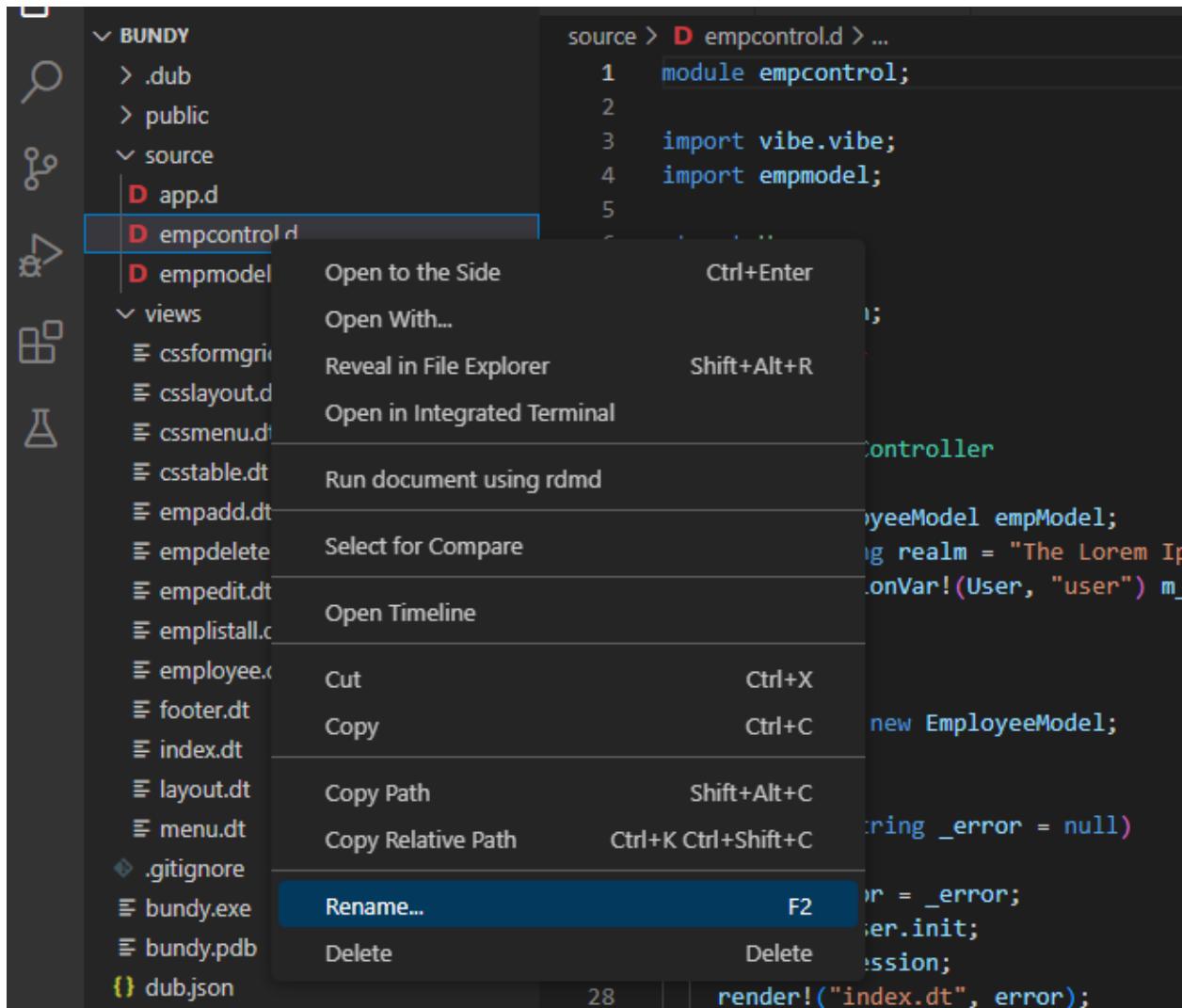
The screenshot shows a web application interface. At the top, there is a navigation bar with links: "Home", "Punch card >", "Employees >", and "Login". Below the navigation bar, the main content area has a title "The Lorem Ipsum Company" and a large block of placeholder text (Lorem ipsum). A modal dialog box is overlaid on the page. The dialog contains input fields for "Email" (with value "admin1@lorem.com") and "Password" (with value "....."). It also includes "Clear", "Login", and "Cancel" buttons. The "Login" button is highlighted in blue, indicating it is the active button.

And we get this.

The screenshot shows the same web application interface after logging in. The navigation bar remains the same. The main content area now displays a table titled "Employees of The Lorem Ipsum Company" with ten rows of employee data. The table columns are: First name, Last name, Department, Phone number, Email address, and Action. The "Action" column contains icons for edit and delete. The data in the table is as follows:

First name	Last name	Department	Phone number	Email address	Action
The	Boss	Management	12345678	theboss@mail.com	
Gal	Gadot	Accounting	0987654321	gal@gadot.com	
Eugene	Bou	Shipping	31247425645	ebouchard@mail.com	
Jess	Alba	Personnel	31247425645	jess@mail.com	
Leigh	Mees	Production	0987654321	leigh@mees.com	
Ale	Dada	Finance	1234567890	ale@dada.com	
Ariel	Win	Production	1234567890	ariel@win.com	
Sien	Benoit	Accounting	0987654321	sien@benoit.com	
Camille	Bell	Production	1234567890	cami@bell.com	
Hedi	Lamar	Production	0987654321	hedi@lamar.com	

Let's rename source\empcontrol.d to source\bundycontrol.d.



And change the class name to BundyController

```
module bundycontrol;

import vibe.vibe;
import empmodel;

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

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

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

And edit source\app.d to instantiate BundyController instead, after importing bundycontrol.

```
import bundycontrol;
```

then

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

and compile and run to test the whole thing.

So here's the whole source\app.d for now:

```
import vibe.vibe;
import bundycontrol;

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 BundyController());

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

    runApplication();
}
```

So here's all of source\bundycontrol.d for now:

```
module bundycontrol;

import vibe.vibe;
import empmodeL;

struct User
{
    bool loggedIn;
    string email;
}

class BundyController
{
    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, first ~ " " ~ last ~ " 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;
}

```

Now we can start with the timekeeping part.

Creating the time-in form

Create views\timein.dt.

```
extends layout
block maincontent
    include cssformgrid.dt
    div.form-grid-wrapper
        br
        h1#time.center-align
        br
        form.form-grid(method="post", action="time_in")
            label.form-grid-label Employee:
            select#empid.form-grid-input(name="empid")
                -foreach(e; emps)
                    option(value="#{e.empid}") #{e.empid} - #{e.fname} #{e.lname}
            div
            div
                input.form-grid-button(type="submit", value="Punch in!")
:javascript
    const displayTime = document.querySelector("#time");
    function showTime() {
        let time = new Date();
        displayTime.innerText = time.toLocaleString("en-CA", { hour12: true });
        setTimeout(showTime, 1000);
    }
    showTime();
```

and edit source\bundycontrol.d and add this method:

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

Now compile, run, refresh the browser and access the Punch in page

Home Punch card » Employees » Login

	Punch in	Employee Ipsum Company					
Employee	Punch out	First name	View cards	Department	Phone number	Email address	Action
The Boss		The		Management	12345678	theboss@mail.com	
Gal Gadot	Create timesheet	Gal		Marketing	0987654321	gal@gadot.com	
Eugene Bouchard		Eugene	Bou	Shipping	31247425645	ebouchard@mail.com	

And this should show

Home Punch card » Employees » Login

2023-01-21, 11:08:58 a.m.

Employee:

But when you press ‘Punch in!’, you get this message

localhost:8080/time_in

404 - Not Found

Not Found

Internal error information:
No routes match path '/time_in'

Time to create the timecards and timesheets models.

Meanwhile, here is the complete source\bundycontrol.d so far.

```
module BundyControl;

import vibe.vibe;
import empmodel;
```

```

struct User
{
    bool loggedIn;
    string email;
}

class BundyController
{
    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);
    }

/* start of employee subsystem *****/
@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 fullname = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
        try moveFile(pic.tempPath, NativePath(fullname));
        catch (Exception ex) copyFile(pic.tempPath, NativePath(fullname), 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, first ~ " " ~ last ~ " not found.");
        render!("employee.dt", e);
    }
/* end of employee subsystem *****/

@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;

/* start of timekeeping subsystem *****/
void getTimeIn()
{
    Employee[] emps = empModel.getEmployees;
    render!("timein.dt", emps);
}

/* end of timekeeping subsystem *****/
}

```

The timecard model

This is the schema we defined for our punch card system:

```
create table timecards
(
    id int not null auto_increment primary key,
    empid char(4) references employees(empid),
    thedate date not null,
    timein time not null,
    timeout time default null,
    updated_at timestamp default null on update current_timestamp
);

create table timesheets
(
    id int not null auto_increment primary key,
    period int not null,
    empid char(4) references employees(empid),
    thedate date not null,
    timein time not null,
    timeout time not null,
    hours float not null
);
```

So here is source\cardmodel.d

```
module cardmodel;

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

struct Timecard
{
    int id;
    string empid;
    string thedate;
    string timein;
    string timeout;
}

class TimecardModel
```

```
{  
    Connection conn;  
  
    this()  
    {  
        string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";  
        conn = new Connection(url);  
        scope(exit) conn.close;  
    }  
  
    bool timeIn(string empid)  
    {  
        string sql = "insert into timecards(empid, thedate, timein) values(?,  
curdate(), curtime())";  
        Prepared pstmt = conn.prepare(sql);  
        pstmt.setArgs(empid);  
        conn.exec(pstmt);  
        return true;  
    }  
  
    bool noTimeInYet(string empid)  
    {  
        string sql = "select * from timecards where empid=? and thedate=?";  
        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[] getCards()  
    {  
        Timecard[] cards;  
        string sql = "select * from timecards order by thedate desc, timein desc";  
        Row[] rows = conn.query(sql).array;  
        if(rows.length == 0) return cards;  
        foreach(row; rows)  
        {
```

```

        Timecard c;
        c.id = to!int(to!string(row[0]));
        c.empid = to!string(row[1]);
        c.thedate = to!string(row[2]);
        c.timein = to!string(row[3]);
        c.timeout = to!string(row[4]);
        cards ~= c;
    }
    return cards;
}
}

```

And here is source\bundycontrol.d

```

module Bundycontrol;

import vibe.vibe;
import empmodeL;
import cardmodeL;

struct User
{
    bool loggedIn;
    string email;
}

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

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

    void index(string _error = null)
    {
        string error = _error;
        m_user = User.init();
        terminateSession();
        render!("index.dt", error);
    }
}

```

```
}

/* start of employee subsystem *****/
@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(realms, 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, first ~ " " ~ last ~ " not found.");
    render!("employee.dt", e);
}
/* end of employee subsystem *****/
@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(HTTPRequest req, HTTPServerResponse res)
{
    if(!m_user.loggedIn) redirect("/");
    return m_user.email;
}
mixin PrivateAccessProxy;

/* start of timekeeping subsystem *****/
void getTimeIn()
{
    Employee[] emps = empModel.getEmployees();
    render!("timein.dt", emps);
}

@errorDisplay!getAllCards
void postTimeIn(string empid)
{
    string fullname = getFullscreen(empid);
    //the record should not exist, otherwise employee already punched in earlier
    enforce(cardModel.noTimeIn(empid), fullname ~ " already punched in today");
    cardModel.timeIn(empid);
    redirect("all_cards");
}

string getFullscreen(string empid)
{
    Employee e = empModel.getEmployee(empid);
    string fullname = to!string(e.fname) ~ " " ~ to!string(e.lname);
    return fullname;
}

void getAllCards(string _error)
{
    string error = _error;
    bool loggedIn = m_user.loggedIn;
    string today = getToday();
    Timecard[] cards = cardModel.getCards();
    render!("cardlistall.dt", cards, today, error, loggedIn);
}

string getToday()
{
```

```

import std.datetime.date;

auto time = Clock.currTime;
return to!string(Date(time.year, time.month, time.day));
}

/* end of timekeeping subsystem *****/
}

```

And here is views\timein.dt

```

extends layout
block maincontent
    include cssformgrid.dt
    div.form-grid-wrapper
        br
        h1#time.center-align
        br
        form.form-grid(method="post", action="time_in")
            label.form-grid-label Employee:
            select#empid.form-grid-input(name="empid")
                -foreach(e; emps)
                    option(value="#{e.empid}") #{e.empid} - #{e.fname} #{e.lname}
            div
            div
                input.form-grid-button(type="submit", value="Punch in!")
:javascript
    const displayTime = document.querySelector("#time");
    function showTime() {
        let time = new Date();
        displayTime.innerText = time.toLocaleString("en-CA", { hour12: true });
        setTimeout(showTime, 1000);
    }
    showTime();

```

And here is views\cardlistall.dt

```

extends layout
block maincontent
    include csstable.dt
    h2 Time cards as of #{today}
    div.table-wrapper
        -if(error)
            div.error-div
                span.error-message #{error}

```

```


| Date                      | Emp #                                                           | Time in                                                            | Time out              |
|---------------------------|-----------------------------------------------------------------|--------------------------------------------------------------------|-----------------------|
| <b>-if(loggedIn)</b>      | <b>th Action</b>                                                |                                                                    |                       |
| <b>-foreach(c; cards)</b> | <b>tr</b>                                                       |                                                                    |                       |
|                           | <b>td #{c.thedate}</b>                                          | <b>td #{c.empid}</b>                                               | <b>td #{c.timein}</b> |
|                           | <b>td #{c.timeout}</b>                                          |                                                                    |                       |
| <b>-if(loggedIn)</b>      | <b>td &amp;nbsp;</b>                                            |                                                                    |                       |
|                           | <b>form.form-hidden(method="get", action="edit_timecard")</b>   |                                                                    |                       |
|                           | <b>input(type="hidden", name="id", value="#{c.id}")</b>         | <b>input(type="image", src="images/pencil.ico", height="15px")</b> |                       |
|                           | <b>  &amp;nbsp;</b>                                             |                                                                    |                       |
|                           | <b>form.form-hidden(method="get", action="delete_timecard")</b> |                                                                    |                       |
|                           | <b>input(type="hidden", name="id", value="#{c.id}")</b>         | <b>input(type="image", src="images/trash.ico", height="15px")</b>  |                       |
|                           | <b>  &amp;nbsp;</b>                                             |                                                                    |                       |


```

Those code should cover the actions from time-in data entry to showing the list of all who punched in.

Compile, run and refresh your browser. There is no need to login for the punch card system.



2023-01-21, 1:30:09 p.m.

Employee: **0001 - The Boss** ▼

Punch in!

After you punched in some employees, you should see this.

[Home](#)[Punch card »](#)[Employees »](#)[Login](#)

Time cards as of 2023-Jan-21

Date	Emp #	Time in	Time out
2023-Jan-21	0002	13:31:23	null
2023-Jan-21	0001	12:31:33	null
2023-Jan-21	1004	12:31:10	null
2023-Jan-21	0007	12:18:13	null

And if you punched in someone who already punched in for the day, you got this:

[Home](#)[Punch card »](#)[Employees »](#)[Login](#)

Time cards as of 2023-Jan-21

Gal Gadot already punched in today

Date	Emp #	Time in	Time out
2023-Jan-21	0002	13:31:23	null
2023-Jan-21	0001	12:31:33	null
2023-Jan-21	1004	12:31:10	null
2023-Jan-21	0007	12:18:13	null

Now let's do the punch-out system.

Punching out

We should make sure that a person who punches in should be able to punch out, but the person who failed to punch in earlier should not be able to punch out, meaning, we check if the punch-in record is already there before we save the punch out time.

Add this code to source\bundycontrol.d

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

@errorDisplay!getAllCards
void postTimeOut(string empid)
{
    string fullname = getfullname(empid);
    //the record should exist, otherwise employee did not punch in earlier
    enforce(!cardModel.noTimeIn(empid), fullname ~ " did not punched in today");
    //the timeout field should be null, otherwise employee already punched out
    enforce(cardModel.noTimeOut(empid), fullname ~ " already punched out today");
    cardModel.timeOut(empid);
    redirect("all_cards");
}
```

It is calling cardModel.noTimeOut(), as well as the cardModel.timeOut(), so let's implement them.

Add this code to source\cardmodel.d

```
bool noTimeOut(string empid)
{
    string sql = "select * from timecards where timeout is null and empid=? and
    thedate=?";
    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;
}

void timeOut(string empid)
{
```

```

        string sql = "update timecards set timeout=curtime() where empid=? and
        thedate=?";
        Prepared pstmt = conn.prepareStatement(sql);
        string today = getToday;
        pstmt.setArgs(empid, today);
        conn.exec(pstmt);
    }

```

That takes care of the database side.

Now let's implement the business logic side.

Add this code to source\bundycontrol.d

```

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

@errorDisplay!getAllCards
void postTimeOut(string empid)
{
    string fullname = getfullname(empid);
    //the record should exist, otherwise employee did not punch in earlier
    enforce(!cardModel.noTimeIn(empid), fullname ~ " did not punched in today");
    //the timeout field should be null, otherwise employee already punched out
    enforce(cardModel.noTimeOut(empid), fullname ~ " already punched out today");
    cardModel.timeOut(empid);
    redirect("all_cards");
}

```

We are making sure that our requirements for a timeout are met.

Compile, run and refresh the browser. Try to timeout someone who did not time in, and this should show:

[Home](#)[Punch card »](#)[Employees »](#)[Login](#)**2023-01-21, 3:24:20 p.m.**

Employee:

0001 - The Boss
0002 - Gal Gadot
0003 - Eugene Bou
0005 - Jess Alba
1004 - Leigh Mees
0007 - Ale Dada
0010 - Ariel Win
0011 - Sien Benoit
0012 - Camille Bell
0013 - Hedi Lamar

[Home](#)[Punch card »](#)[Employees »](#)[Login](#)**2023-01-21, 3:28:48 p.m.**

Employee:

[Home](#)[Punch card »](#)[Employees »](#)[Login](#)

Time cards as of 2023-Jan-21

Gal Gadot did not punched in today

Date	Emp #	Time in	Time out
2023-Jan-21	0013	15:01:55	null
2023-Jan-21	0012	15:01:49	null
2023-Jan-21	0011	15:01:42	null
2023-Jan-21	0010	15:01:38	null
2023-Jan-21	0005	15:01:20	null
2023-Jan-21	0003	15:01:14	null
2023-Jan-21	0001	12:31:33	14:21:14
2023-Jan-21	1004	12:31:10	15:23:12

And if you try to punch out someone who already punched out, you also get this:

[Home](#)[Punch card »](#)[Employees »](#)[Login](#)

Time cards as of 2023-Jan-21

Leigh Mees already punched out today

Date	Emp #	Time in	Time out
2023-Jan-21	0013	15:01:55	null
2023-Jan-21	0012	15:01:49	null
2023-Jan-21	0011	15:01:42	null
2023-Jan-21	0010	15:01:38	null
2023-Jan-21	0005	15:01:20	null
2023-Jan-21	0003	15:01:14	null
2023-Jan-21	0001	12:31:33	14:21:14
2023-Jan-21	1004	12:31:10	15:23:12

But if our conditions are met, everything's good.

[Home](#)[Punch card »](#)[Employees »](#)[Login](#)

2023-01-21, 3:38:44 p.m.

Employee:

[Home](#)[Punch card »](#)[Employees »](#)[Login](#)

Time cards as of 2023-Jan-21

Date	Emp #	Time in	Time out
2023-Jan-21	0013	15:01:55	null
2023-Jan-21	0012	15:01:49	null
2023-Jan-21	0011	15:01:42	15:39:46
2023-Jan-21	0010	15:01:38	null
2023-Jan-21	0005	15:01:20	null
2023-Jan-21	0003	15:01:14	15:38:23
2023-Jan-21	0001	12:31:33	14:21:14
2023-Jan-21	1004	12:31:10	15:23:12
2023-Jan-21	0007	12:18:13	null

The time-in and time-out actions were covered.

So here is the full source\bundycontrol.d

```
module BundyControl;

import vibe.vibe;
import empmodel;
import cardmodel;

struct User
{
    bool loggedIn;
    string email;
}

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

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

    void index(string _error = null)
    {
        string error = _error;
        m_user = User.init;
        terminateSession;
        render!("index.dt", error);
    }

/* start of employee subsystem *****/
@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, first ~ " " ~ last ~ " not found.");
    render!("employee.dt", e);
}
/* end of employee subsystem *****/

```

```

@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;

```

```

/* start of timekeeping subsystem *****/
void getTimeIn()
{
    Employee[] emps = empModel.getEmployees;
    render!("timein.dt", emps);
}

@errorDisplay!getAllCards
void postTimeIn(string empid)
{
    string fullname = getFullscreen(empid);
    //the record should not exist, otherwise employee already punched in earlier
    enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in
today");
    cardModel.timeIn(empid);
    redirect("all_cards");
}

string getFullscreen(string empid)
{
    Employee e = empModel.getEmployee(empid);
    string fullname = to!string(e.fname) ~ " " ~ to!string(e.lname);
    return fullname;
}

void getAllCards(string _error)
{
    string error = _error;
    bool loggedIn = m_user.loggedIn;
    string today = getToday();
    Timecard[] cards = cardModel.getCards();
    render!("cardlistall.dt", cards, today, error, loggedIn);
}

string getToday()
{
    import std.datetime.date;

    auto time = Clock.currTime;
    return to!string(Date(time.year, time.month, time.day));
}

void getTimeOut()
{
    Employee[] emps = empModel.getEmployees;
}

```

```

        render!("timeout.dt", emps);
    }

@errorDisplay!getAllCards
void postTimeOut(string empid)
{
    string fullname = getfullname(empid);
    //the record should exist, otherwise employee did not punch in earlier
    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");
    cardModel.timeOut(empid);
    redirect("all_cards");
}

/* end of timekeeping subsystem *****/
}

```

And here is the full source\cardmodel.d

```

module cardmodel;

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

struct Timecard
{
    int id;
    string empid;
    string thedate;
    string timein;
    string timeout;
}

class TimecardModel
{
    Connection conn;

    this()
    {
        string url = "host=localhost;port=3306;user=owner;pwd=qwerty;db=empdb";

```

```
conn = new Connection(url);
scope(exit) conn.close;
}

void timeIn(string empid)
{
    string sql = "insert into timecards(empid, thedate, timein) values(?, curdate(), curtime())";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(empid);
    conn.exec(pstmt);
}

bool noTimeIn(string empid)
{
    string sql = "select * from timecards where empid=? and thedate=?";
    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[] getCards()
{
    Timecard[] cards;
    string sql = "select * from timecards order by thedate desc, timein desc";
    Row[] rows = conn.query(sql).array;
    if(rows.length == 0) return cards;
    foreach(row; rows)
    {
        Timecard c;
        c.id = to!int(to!string(row[0]));
        c.empid = to!string(row[1]);
        c.thedate = to!string(row[2]);
        c.timein = to!string(row[3]);
        c.timeout = to!string(row[4]);
        cards ~= c;
    }
}
```

```

    }
    return cards;
}

bool noTimeOut(string empid)
{
    string sql = "select * from timecards where timeout is null and empid=? and thedate=?";
    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;
}

void timeOut(string empid)
{
    string sql = "update timecards set timeout=curtime() where empid=? and thedate=?";
    Prepared pstmt = conn.prepare(sql);
    string today = getToday();
    pstmt.setArgs(empid, today);
    conn.exec(pstmt);
}
}

```

How about editing of the timecards so that those who failed to punch in can still punch out?

If you login and choose ‘View cards’ on the menu, you should have a slightly different view of the cards list.

The screenshot shows a web application interface. At the top, there is a navigation bar with links for 'Home', 'Punch card >', 'Employees >', and 'Login'. Below the navigation bar, the main content area displays a table titled 'Time cards as of 2023-Jan-21'. The table has columns for Date, Emp #, Time in, and Time out. The data in the table is:

Date	Emp #	Time in	Time out
2023-Jan-21	0013	15:01:55	null
2023-Jan-21	0012	15:01:49	null
2023-Jan-21	0011	15:01:42	15:39:46

To the right of the table, a modal dialog box is open, prompting for login information. The dialog contains two input fields: one for email ('admin1@lorem.com') and one for password ('*****'). Below the password field are three buttons: 'Clear', 'Login', and 'Cancel'.

The screenshot shows a web-based application interface. At the top, there's a navigation bar with links for "Home", "Punch card »", "Employees »", and "Login". Below the navigation bar, there's a sidebar with sections for "Punch in" and "Punch out". The main content area displays a table titled "Employee Ipsum Company" with three rows of data. The first row has columns for First name, Last name, Department, Phone number, Email address, and Action. The second row has columns for First name, Last name, Department, Phone number, Email address, and Action. The third row has columns for First name, Last name, Department, Phone number, Email address, and Action. A red oval highlights the "View cards" link in the sidebar.

First name	Last name	Department	Phone number	Email address	Action
The Boss	Administrator	12345678	theboss@mail.com		
Gal Gadot	Marketing	0987654321	gal@gadot.com		
Eugene Bouchard	Shipping	31247425645	ebouchard@mail.com		

Time cards as of 2023-Jan-21

Date	Emp #	Time in	Time out	Action
2023-Jan-21	0013	15:01:55	null	
2023-Jan-21	0012	15:01:49	null	
2023-Jan-21	0011	15:01:42	15:39:46	
2023-Jan-21	0010	15:01:38	null	
2023-Jan-21	0005	15:01:20	null	
2023-Jan-21	0003	15:01:14	15:38:23	
2023-Jan-21	0001	12:31:33	14:21:14	
2023-Jan-21	1004	12:31:10	15:23:12	
2023-Jan-21	0007	12:18:13	null	

So if logged in (meaning, you are an administrator), you should be able to edit the timecards.

But if you do click on the pencil icon, you get this:

The screenshot shows a browser window with a 404 Not Found error page. The URL in the address bar is "localhost:8080/edit_timecard?id=43&x=6&y=7". The page content includes standard browser navigation icons (back, forward, search, etc.) and a message "404 - Not Found". Below the message, there is some internal error information: "Not Found" and "Internal error information: No routes match path '/edit_timecard?id=43&x=6&y=7'".

Because we haven't written the code for the editing of timecards.

Let's do that next.

Editing the timecards

Add this code to the end of source\bundycontrol.d

```
void getEditTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getCard(id);
    Employee e = empModel.getEmployeeByEmpid(t.empid);
    render!("cardedit.dt", t, e, error);
}

@errorDisplay!getEditTimecard
void postEditTimecard(Timecard t)
{
    cardModel.editTimecard(t);
    redirect("all_cards");
}
```

The getEditTimecard() method is using cardedit.dt, so let's create it.

Create views\cardedit.dt

```
extends layout
block maincontent
    include cssformtimecard
    h2 Time card of #{e.fname} #{e.lname} for #{t.thedate}
    div.form-grid-wrapper
        -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 #{e.empid} - #{e.fname} #{e.lname}
            label.form-grid-label Punch in :
            input.form-grid-input(type="text", name="t_timein", value="#{t.timein}")
            label.form-grid-label Punch 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="id", value="#{t.id}")
                input(type="hidden", name="t_empid", value="#{t.empid}")
                input(type="hidden", name="t_thedate", value="#{t.thedate}")
            div
            div
```

```

a\(href="all\_cards"\)
    button.form-grid-button(type="button") Cancel
  div
    input.form-grid-button(type="submit", value="Submit")

```

The getEditTimecard() method is also calling cardModel.getCard(), so let's write it.

Add this code to source\cardmodel.d

```

Timecard getCard(int id)
{
  Timecard c;
  string sql = "select * from timecards where id=?";
  Prepared pstmt = conn.prepare(sql);
  pstmt.setArgs(id);
  Row[] rows = conn.query(pstmt).array;
  if(rows.length == 0) return c;
  Row row = rows[0];
  c.id = to!int(to!string(row[0]));
  c.empid = to!string(row[1]);
  c.thedate = to!string(row[2]);
  c.timein = to!string(row[3]);
  c.timeout = to!string(row[4]);
  return c;
}

```

The getEditTimecard() method is also calling empModel.getEmployeeByEmpid(), so let's add that to the EmployeeModel class.

Add this to source\empmodel.d

```

Employee getEmployeeByEmpid(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]);
}

```

cardedit.dt needs cssformtimecard.dt, so let's create it.

Create views\cssformtimecard.dt

```
:css
.form-grid-wrapper
{
    width: 450px;
    height: auto;
    margin: 20px 0;
    padding: 10px;
    border-radius: 10px;
    background-color: #eef;
}
.form-grid
{
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 5px;
}
.form-grid-label
{
    width: 100%;
    font-size: 16px;
    text-align: right;
    padding: 2px;
}
.form-grid-text
{
    width: 100%;
    font-size: 16px;
    font-weight: bold;
    padding: 2px;
}
.form-grid-input
{
    width: 95%;
    font-size: 14px;
    padding: 2px 0 2px 2px;
}
.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: 0;
    padding: 0;
}
```

And the postEditTimecard() method is calling cardModel.editTimecard(), so let's write it.

```
ulong editTimecard(Timecard t)
{
    Prepared pstmt;
    if(to!string(t.timeout) == "null")
    {
        string sql = "update timecards set timein=?, timeout=null where id=?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.id);
    }
    else
    {
        string sql = "update timecards set timein=?, timeout=? where id=?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.timeout, t.id);
    }
    return conn.exec(pstmt);
}
```

Now compile, run and refresh the browser. Be sure to login first so you can edit any record in the list of timecards.

[Home](#)[Punch card »](#)[Employees »](#)[Login](#)

Time cards as of 2023-Jan-21

Date	Emp #	Time in	Time out	Action
2023-Jan-21	0013	15:01:55	null	
2023-Jan-21	0012	15:01:49	null	
2023-Jan-21	0011	15:01:42	15:39:46	
2023-Jan-21	0010	15:01:38	null	
2023-Jan-21	0005	15:01:20	null	
2023-Jan-21	0003	15:01:14	15:38:23	
2023-Jan-21	0001	12:31:33	14:21:15	
2023-Jan-21	1004	12:31:10	15:23:12	
2023-Jan-21	0007	12:18:13	null	

Choose any record to open the card editing page

[Home](#)[Punch card »](#)[Employees »](#)[Login](#)

Time card of Ariel Win for 2023-Jan-21

Employee : 0010 - Ariel Win

Punch in :

Punch out :

Try both the 'Cancel' and the 'Submit' buttons.

Now, how does an employee who forgot to time in can still recover?

1. The employee should time in even though it's late.
2. The employee can now time out
3. The admin can then edit the time-in time

So here's the full source\bundycontrol so far

```
module BundyControl;

import vibe.vibe;
import empmodel;
```

```
import cardmodel;

struct User
{
    bool loggedIn;
    string email;
}

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

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

    void index(string _error = null)
    {
        string error = _error;
        m_user = User.init;
        terminateSession;
        render!("index.dt", error);
    }

/* start of employee subsystem *****/
@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 != 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, first ~ " " ~ last ~ " not found.");
    render!("employee.dt", e);
}
/* end of employee subsystem *****/

@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;

/* start of timekeeping subsystem *****/
void getTimeIn()
{
    Employee[] emps = empModel.getEmployees;
    render!("timein.dt", emps);
}

@errorDisplay!getAllCards

```

```

void postTimeIn(string empid)
{
    string fullname = getFullscreen(empid);
    //the record should not exist, otherwise employee already punched in earlier
    enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in
today");
    cardModel.timeIn(empid);
    redirect("all_cards");
}

string getFullscreen(string empid)
{
    Employee e = empModel.getEmployee(empid);
    string fullname = to!string(e.fname) ~ " " ~ to!string(e.lname);
    return fullname;
}

void getAllCards(string _error)
{
    string error = _error;
    bool loggedIn = m_user.loggedIn;
    string today = getToday();
    Timecard[] cards = cardModel.getCards();
    render!("cardlistall.dt", cards, today, error, loggedIn);
}

string getToday()
{
    import std.datetime.date;

    auto time = Clock.currTime;
    return to!string(Date(time.year, time.month, time.day));
}

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

@errorDisplay!getAllCards
void postTimeOut(string empid)
{
    string fullname = getFullscreen(empid);
    //the record should exist, otherwise employee did not punch in earlier
}

```

```

    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");
    cardModel.timeOut(empid);
    redirect("all_cards");
}

void getEditTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getCard(id);
    Employee e = empModel.getEmployeeByEmpid(t.empid);
    render!("cardedit.dt", t, e, error);
}

@errorDisplay!getEditTimecard
void postEditTimecard(Timecard t)
{
    cardModel.editTimecard(t);
    redirect("all_cards");
}

/* end of timekeeping subsystem *****/
}

```

And here's source\cardmodel.d so far

```

module cardmodel;

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

struct Timecard
{
    int id;
    string empid;
    string thedate;
    string timein;
    string timeout;
}

```

```
class TimecardModel
{
    Connection conn;

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

    void timeIn(string empid)
    {
        string sql = "insert into timecards(empid, thedate, timein) values(?, curdate(), curtime())";
        Prepared pstmt = conn.prepare(sql);
        pstmt.setArgs(empid);
        conn.exec(pstmt);
    }

    bool noTimeIn(string empid)
    {
        string sql = "select * from timecards where empid=? and thedate=?";
        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[] getCards()
    {
        Timecard[] cards;
        string sql = "select * from timecards order by thedate desc, timein desc";
        Row[] rows = conn.query(sql).array;
        if(rows.length == 0) return cards;
        foreach(row; rows)
        {
            Timecard card;
            card.empid = row["empid"];
            card.thedate = row["thedate"];
            card.timein = row["timein"];
            card.hours = row["hours"];
            card.minutes = row["minutes"];
            card.seconds = row["seconds"];
            cards ~= card;
        }
        return cards;
    }
}
```

```

Timecard c;
c.id = to!int(to!string(row[0]));
c.empid = to!string(row[1]);
c.thedate = to!string(row[2]);
c.timein = to!string(row[3]);
c.timeout = to!string(row[4]);
cards ~= c;
}
return cards;
}

bool noTimeOut(string empid)
{
    string sql = "select * from timecards where timeout is null and empid=? and thedate=?";
    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;
}

void timeOut(string empid)
{
    string sql = "update timecards set timeout=curtime() where empid=? and thedate=?";
    Prepared pstmt = conn.prepare(sql);
    string today = getToday;
    pstmt.setArgs(empid, today);
    conn.exec(pstmt);
}

Timecard getCard(int id)
{
    Timecard c;
    string sql = "select * from timecards where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return c;
    Row row = rows[0];
    c.id = to!int(to!string(row[0]));
    c.empid = to!string(row[1]);
    c.thedate = to!string(row[2]);
}

```

```
c.timein = to!string(row[3]);
c.timeout = to!string(row[4]);
return c;
}

ulong editTimecard(Timecard t)
{
    Prepared pstmt;
    if(to!string(t.timeout) == "null")
    {
        string sql = "update timecards set timein=?, timeout=null where id=?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.id);
    }
    else
    {
        string sql = "update timecards set timein=?, timeout=? where id=?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.timeout, t.id);
    }
    return conn.exec(pstmt);
}
```

Now, how about deleting a timecard?

Deleting timecards

Add this to source\bundycontrol.d

```
void getDeleteTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getCard(id);
    Employee e = empModel.getEmployeeByEmpid(t.empid);
    render!("carddelete.dt", t, e, error);
}

@errorDisplay!getDeleteTimecard
void postDeleteTimecard(int id)
{
    cardModel.deleteTimecard(id);
    redirect("all_cards");
}
```

The method getDeleteTimecard() needs carddelete.dt, so let's create it.

Create views\carddelete.dt

```
extends layout
block maincontent
    include cssformtimecard
    h2 Sure you want to delete #{e.fname} #{e.lname}'s punch card?
    div.form-grid-wrapper
        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 Date :
            label.form-grid-text #{t.thedate}
            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="t_id", value="#{t.id}")
            input(type="hidden", name="id", value="#{t.id}")
            input(type="hidden", name="t_empid", value="#{t.empid}")
            input(type="hidden", name="t_thedate", value="#{t.thedate}")
            input(type="hidden", name="t_timein", value="#{t.timein}")
            input(type="hidden", name="t_timeout", value="#{t.timeout}")
        div
        div
```

```


a(href="all_cards")
    button.form-grid-button(type="button") Cancel



```

The method getDeleteTimecard() is calling cardModel.deleteTimecard(), so let's write it.

Add 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 refresh the browser. Login again to show the pencil and trash icons, and try to delete a record.

The screenshot shows a web application interface. At the top, there is a navigation bar with links for 'Home', 'Punch card >', 'Employees >', and 'Login'. Below the navigation bar, a modal dialog box is open. It contains two input fields: one for email ('admin1@lorem.com') and one for password ('.....'). Below the password field are three buttons: 'Clear', 'Login', and 'Cancel'. In the background, there is a table titled 'Time cards as of 2023-Jan-21' with the following data:

Date	Emp #	Time in	Time out
2023-Jan-21	0013	15:01:55	null
2023-Jan-21	0012	15:01:49	null
2023-Jan-21	0011	15:01:42	15:39:46

The screenshot shows a web application interface. At the top, there is a navigation bar with links for 'Home', 'Punch card >', 'Employees >', and 'Login'. Below the navigation bar, there is a dropdown menu with options: 'Punch in', 'Punch out', and 'View cards'. A red oval highlights the 'View cards' option. To the right of the dropdown, there is a table with the following data:

Employee List

First name	Last name	Department	Phone number	Email address	Action
Theo	Gal	Management	12345678	theboss@mail.com	
Gwen	Smith	Management	0987654321	gwen.smith@gadot.com	

Time cards as of 2023-Jan-21

Date	Emp #	Time in	Time out	Action
2023-Jan-21	0013	15:01:55	null	
2023-Jan-21	0012	15:01:49	null	
2023-Jan-21	0011	15:01:42	15:39:46	
2023-Jan-21	0010	15:01:38	null	
2023-Jan-21	0005	15:01:20	null	
2023-Jan-21	0003	15:01:14	15:38:23	
2023-Jan-21	0001	12:31:33	14:21:15	
2023-Jan-21	1004	12:31:10	15:23:12	
2023-Jan-21	0007	12:18:13	null	

Choose a record

Sure you want to delete Ariel Win's punch card?

Employee : 0010 - Ariel Win

Date : 2023-Jan-21

Punched in : 15:01:38

Punched out : null

[Cancel](#)[Delete](#)

Try the 'Delete' button and the deleted record should be gone.

Time cards as of 2023-Jan-21

Date	Emp #	Time in	Time out	Action
2023-Jan-21	0013	15:01:55	null	
2023-Jan-21	0012	15:01:49	null	
2023-Jan-21	0011	15:01:42	15:39:46	
2023-Jan-21	0005	15:01:20	null	
2023-Jan-21	0003	15:01:14	15:38:23	
2023-Jan-21	0001	12:31:33	14:21:15	
2023-Jan-21	1004	12:31:10	15:23:12	
2023-Jan-21	0007	12:18:13	null	

With that, we've done all the CRUD stuff with the timecards.

So here's source\bundycontrol.d so far

```
module Bundycontrol;

import vibe.vibe;
import empmodel;
import cardmodel;

struct User
{
    bool loggedIn;
    string email;
}

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

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

    void index(string _error = null)
    {
        string error = _error;
        m_user = User.init;
        terminateSession;
        render!("index.dt", error);
    }

/* start of employee subsystem *****/
@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, first ~ " " ~ last ~ " not found.");
    render!("employee.dt", e);
}
/* end of employee subsystem *****/

```

```

@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;

```

```

/* start of timekeeping subsystem *****/
void getTimeIn()
{
    Employee[] emps = empModel.getEmployees;
    render!("timein.dt", emps);
}

@errorDisplay!getAllCards
void postTimeIn(string empid)
{
    string fullname = getFullscreen(empid);
    //the record should not exist, otherwise employee already punched in earlier
    enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in
today");
    cardModel.timeIn(empid);
    redirect("all_cards");
}

string getFullscreen(string empid)
{
    Employee e = empModel.getEmployee(empid);
    string fullname = to!string(e.fname) ~ " " ~ to!string(e.lname);
    return fullname;
}

void getAllCards(string _error)
{
    string error = _error;
    bool loggedIn = m_user.loggedIn;
    string today = getToday();
    Timecard[] cards = cardModel.getCards();
    render!("cardlistall.dt", cards, today, error, loggedIn);
}

string getToday()
{
    import std.datetime.date;

    auto time = Clock.currTime;
    return to!string(Date(time.year, time.month, time.day));
}

void getTimeOut()
{
    Employee[] emps = empModel.getEmployees;
}

```

```

        render!("timeout.dt", emps);
    }

@errorDisplay!getAllCards
void postTimeOut(string empid)
{
    string fullname = getfullname(empid);
    //the record should exist, otherwise employee did not punch in earlier
    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");
    cardModel.timeOut(empid);
    redirect("all_cards");
}

void getEditTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getCard(id);
    Employee e = empModel.getEmployeeByEmpid(t.empid);
    render!("cardedit.dt", t, e, error);
}

@errorDisplay!getEditTimecard
void postEditTimecard(Timecard t)
{
    cardModel.editTimecard(t);
    redirect("all_cards");
}

void getDeleteTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getCard(id);
    Employee e = empModel.getEmployeeByEmpid(t.empid);
    render!("carddelete.dt", t, e, error);
}

@errorDisplay!getDeleteTimecard
void postDeleteTimecard(int id)
{
    cardModel.deleteTimecard(id);
    redirect("all_cards");
}

```

```

    }

/* end of timekeeping subsystem *****/
}

```

And here's source\cardmodel.d so far

```

module cardmodel;

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

struct Timecard
{
    int id;
    string empid;
    string thedate;
    string timein;
    string timeout;
}

class TimecardModel
{
    Connection conn;

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

    void timeIn(string empid)
    {
        string sql = "insert into timecards(empid, thedate, timein) values(?, curdate(), curtime())";
        Prepared pstmt = conn.prepare(sql);
        pstmt.setArgs(empid);
        conn.exec(pstmt);
    }

    bool noTimeIn(string empid)
    {

```

```
string sql = "select * from timecards where empid=? and thedate=?";
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[] getCards()
{
    Timecard[] cards;
    string sql = "select * from timecards order by thedate desc, timein desc";
    Row[] rows = conn.query(sql).array;
    if(rows.length == 0) return cards;
    foreach(row; rows)
    {
        Timecard c;
        c.id = to!int(to!string(row[0]));
        c.empid = to!string(row[1]);
        c.thedate = to!string(row[2]);
        c.timein = to!string(row[3]);
        c.timeout = to!string(row[4]);
        cards ~= c;
    }
    return cards;
}

bool noTimeOut(string empid)
{
    string sql = "select * from timecards where timeout is null and empid=? and
theday=?";
    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;
}
```

```
void timeOut(string empid)
{
    string sql = "update timecards set timeout=curtime() where empid=? and
thedate=?";
    Prepared pstmt = conn.prepare(sql);
    string today = getToday;
    pstmt.setArgs(empid, today);
    conn.exec(pstmt);
}

Timecard getCard(int id)
{
    Timecard c;
    string sql = "select * from timecards where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return c;
    Row row = rows[0];
    c.id = to!int(to!string(row[0]));
    c.empid = to!string(row[1]);
    c.thedate = to!string(row[2]);
    c.timein = to!string(row[3]);
    c.timeout = to!string(row[4]);
    return c;
}

ulong editTimecard(Timecard t)
{
    Prepared pstmt;
    if(to!string(t.timeout) == "null")
    {
        string sql = "update timecards set timein=?, timeout=null where id=?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.id);
    }
    else
    {
        string sql = "update timecards set timein=?, timeout=? where id?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.timeout, t.id);
    }
    return 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);
}
}

```

We added one method to EmployeeModel, so here's source\empmodel.d now

```

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",

```

```
"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"]
];

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]);
}

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]);
}

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.prepareStatement(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.prepareStatement(sql);
        pstmt.setArgs(email, password);
        Row[] rows = conn.query(pstmt).array;
        if(rows.length == 0) return false;
        return true;
    }

    Employee getEmployeeByEmpid(string empid)
    {
        string sql = "select * from employees where empid=?";
        Prepared 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 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);
}
*/

```

Now let's start creating the timesheet.

Timesheets

Timesheets get their data from timecards but have additional info and some summarization, and then can be further used for salary calculations, although that part we will not do.

What we will do is simply copy the contents of timecards for an indicated period into the timesheets table with calculations for the number of hours worked for each worker per day. Every time the timesheet is generated, it is erased before being filled with new data specific to the period requested, but the timecards remain growing.

Create source\sheetmodel.d

```
module sheetmodel;

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

struct Timesheet
{
    int id;
    int period;
    string empid;
    string thedate;
    string timein;
    string timeout;
    double hours;
}

class TimesheetModel
{
    Connection conn;

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

Next, add this code to source\bundycontrol.d

```
void getCreateTimesheet(string _error = null)
{
    string error = _error;
```

```

    string[] dates = cardModel.getThedates;
    render!("sheetcreate.dt", dates, error);
}

```

This code is calling cardModel.getThedates() method and rendering a form, so let's write them.

Add this code to source\cardmodel.d

```

string[] getThedates()
{
    string sql = "select distinct thedate from timecards order by thedate";
    Row[] rows = conn.query(sql).array;
    string[] dates;
    foreach (row; rows) dates ~= to!string(row[0]);
    return dates;
}

```

Next, create views\sheetscreate.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")
            label.form-grid-label From:
            select#from.form-grid-input(name="from")
                -foreach(d; dates)
                    option(value="#{d}") #{d}
            div
            div
            label.form-grid-label Up to:
            select#from.form-grid-input(name="upto")
                -foreach(d; dates)
                    option(value="#{d}") #{d}
            div
            div
            input.form-grid-button(type="submit", value="Generate timesheet")

```

Then compile and run and refresh the browser to test what we just did.

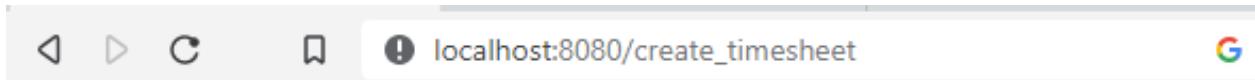
When you click ‘Create timesheet’ on the menu, you should see this:



From:

Up to:

But when you click on ‘Generate timesheet’, you see this:



404 - Not Found

Not Found

Internal error information:
No routes match path '/create_timesheet'

So let's write the code for creating the timesheet.

But first, here's the whole source\bundycontrol.d as of now:

```
module Bundycontrol;

import vibe.vibe;
import empmodeL;
import cardmodeL;
import sheetmodeL;

struct User
{
    bool loggedIn;
    string email;
}

class BundyController
{
    private EmployeeModel empModel;
    private TimecardModel cardModel;
    private TimesheetModel sheetModel;
    private string realm = "The Lorem Ipsum Company";
    private SessionVar!(User, "user") m_user;
```

```

this()
{
    empModel = new EmployeeModel;
    cardModel = new TimecardModel;
    sheetModel = new TimesheetModel;
}

void index(string _error = null)
{
    string error = _error;
    m_user = User.init;
    terminateSession;
    render!("index.dt", error);
}

/* start of employee subsystem *****/
@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(fullfile), 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 fullname = dir ~ e.fname ~ "_" ~ e.lname ~ ext;
        try moveFile(pic.tempPath, NativePath(fullname));
        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, first ~ " " ~ last ~ " not found.");
    render!("employee.dt", e);
}
/* end of employee subsystem *****/

```

```

@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;

/* start of timekeeping subsystem *****/
void getTimeIn()
{
    Employee[] emps = empModel.getEmployees;
    render!("timein.dt", emps);
}

@errorDisplay!getAllCards
void postTimeIn(string empid)
{
    string fullname = getfullname(empid);
    //the record should not exist, otherwise employee already punched in earlier
    enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in
today");
    cardModel.timeIn(empid);
    redirect("all_cards");
}

string getfullname(string empid)
{
    Employee e = empModel.getEmployee(empid);

```

```

    string fullname = to!string(e.fname) ~ " " ~ to!string(e.lname);
    return fullname;
}

void getAllCards(string _error)
{
    string error = _error;
    bool loggedIn = m_user.loggedIn;
    string today = getToday();
    Timecard[] cards = cardModel.getCards();
    render!("cardlistall.dt", cards, today, error, loggedIn);
}

string getToday()
{
    import std.datetime.date;

    auto time = Clock.currTime;
    return to!string(Date(time.year, time.month, time.day));
}

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

@errorDisplay!getAllCards
void postTimeOut(string empid)
{
    string fullname = getFullscreen(empid);
    //the record should exist, otherwise employee did not punch in earlier
    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");
    cardModel.timeOut(empid);
    redirect("all_cards");
}

void getEditTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getCard(id);
}

```

```

Employee e = empModel.getEmployeeByEmpid(t.empid);
render!("cardedit.dt", t, e, error);
}

@errorDisplay!getEditTimecard
void postEditTimecard(Timecard t)
{
    cardModel.editTimecard(t);
    redirect("all_cards");
}

void getDeleteTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getCard(id);
    Employee e = empModel.getEmployeeByEmpid(t.empid);
    render!("carddelete.dt", t, e, error);
}

@errorDisplay!getDeleteTimecard
void postDeleteTimecard(int id)
{
    cardModel.deleteTimecard(id);
    redirect("all_cards");
}

void getCreateTimesheet(string _error = null)
{
    string error = _error;
    string[] dates = cardModel.getThedates;
    render!("sheetcreate.dt", dates, error);
}

/* end of timekeeping subsystem *****/
}

```

And here's source\sheetmodel.d as of now:

```

module sheetmodel;

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

struct Timesheet

```

```

{
    int id;
    int period;
    string empid;
    string thedate;
    string timein;
    string timeout;
    double hours;
}

class TimesheetModel
{
    Connection conn;

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

```

We added one method to the TimecardModel class so here's source\cardmodel.d as of now.

```

module cardmodel;

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

struct Timecard
{
    int id;
    string empid;
    string thedate;
    string timein;
    string timeout;
}

class TimecardModel
{
    Connection conn;

    this()

```

```

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

void timeIn(string empid)
{
    string sql = "insert into timecards(empid, thedate, timein) values(?,,
curdate(), curtime())";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(empid);
    conn.exec(pstmt);
}

bool noTimeIn(string empid)
{
    string sql = "select * from timecards where empid=? and thedate=?";
    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]);
}

string[] getThedates()
{
    string sql = "select distinct thedate from timecards order by thedate";
    Row[] rows = conn.query(sql).array;
    string[] dates;
    foreach (row; rows) dates ~= to!string(row[0]);
    return dates;
}

Timecard[] getCards()
{
    Timecard[] cards;
    string sql = "select * from timecards order by thedate desc, timein desc";
}

```

```

Row[ ] rows = conn.query(sql).array;
if(rows.length == 0) return cards;
foreach(row; rows)
{
    Timecard c;
    c.id = to!int(to!string(row[0]));
    c.empid = to!string(row[1]);
    c.thedate = to!string(row[2]);
    c.timein = to!string(row[3]);
    c.timeout = to!string(row[4]);
    cards ~= c;
}
return cards;
}

bool noTimeOut(string empid)
{
    string sql = "select * from timecards where timeout is null and empid=? and thedate=?";
    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;
}

void timeOut(string empid)
{
    string sql = "update timecards set timeout=curtime() where empid=? and thedate=?";
    Prepared pstmt = conn.prepare(sql);
    string today = getToday;
    pstmt.setArgs(empid, today);
    conn.exec(pstmt);
}

Timecard getCard(int id)
{
    Timecard c;
    string sql = "select * from timecards where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Row[ ] rows = conn.query(pstmt).array;
    if(rows.length == 0) return c;
}

```

```

Row row = rows[0];
c.id = to!int(to!string(row[0]));
c.empid = to!string(row[1]);
c.thedate = to!string(row[2]);
c.timein = to!string(row[3]);
c.timeout = to!string(row[4]);
return c;
}

ulong editTimecard(Timecard t)
{
    Prepared pstmt;
    if(to!string(t.timeout) == "null")
    {
        string sql = "update timecards set timein=?, timeout=null where id=?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.id);
    }
    else
    {
        string sql = "update timecards set timein=?, timeout=? where id=?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.timeout, t.id);
    }
    return 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);
}

```

Now let's go on to creating the timesheet.

Creating the timesheet

We are going to extract data from the timecards table to the timesheets table, using the dates passed from the form as basis for copying.

Add this code to source\bundycontrol.d

```
@errorDisplay!getCreateTimesheet
void postCreateTimesheet(string from, string upto)
{
    sheetModel.createTimesheets(from, upto);
    redirect("all_timesheets");
}

void getAllTimesheets()
{
    Timesheet[] sheets = sheetModel.getAllTimesheets;
    Employee[] emps = empModel.getEmployees;
    string[] dates = sheetModel.getThedates;
    render!("sheetlistall.dt", sheets, emps, dates);
}
```

Each of these methods are calling methods from another class. But since they are related, we opted to simply write them together as these steps look familiar now: we open a form to get input data, we use that input data to process the database data, then output the result to a web page.

The code is calling createTimsheets(), getAllTimesheets() and getTheDates() methods from TimesheetModel, so let's write them.

Add this code to source\sheetsmodel.d

```
string[] getThedates()
{
    string sql = "select distinct thedate from timesheets order by thedate";
    Row[] rows = conn.query(sql).array;
    string[] dates;
    foreach (row; rows) dates ~= to!string(row[0]);
    return dates;
}

void createTimesheets(string from, string upto)
{
    string sql = "delete from timesheets";
    conn.exec(sql);
```

```

        sql = "insert into timesheets(period, empid, thedate, timein, timeout,
hours)
            select week(thedate), empid, thedate, timein, timeout,
round((time_to_sec(timeout)-time_to_sec(timein))/3600,2)
            from timecards where timeout is not null
            and thedate between str_to_date(?, '%Y-%b-%d') and
str_to_date(?, '%Y-%b-%d')
            order by thedate desc, empid";
        Prepared pstmt = conn.prepare(sql);
        pstmt.setArgs(from, upto);
        conn.exec(pstmt);
    }

Timesheet[] getAllTimesheets()
{
    string sql = "select * from timesheets";
    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.thedate = to!string(row[3]);
        t.timein = to!string(row[4]);
        t.timeout = to!string(row[5]);
        t.hours = to!double(to!string(row[6]));
        sheets ~= t;
    }
    return sheets;
}

```

The shitlistall.dt template is being rendered, so let's create it.

Create views\sheetlistall.dt

```

extends layout
block maincontent
    include csstable.dt
    h2 Timesheets
    div.form-grid-wrapper
        form.form-grid(method="post", action="by_empid")
            input.form-grid-input(type="submit", value="By employee")
            select.form-grid-input(name="empid")

```

```

-foreach(e; emps)
    option(value="#{e.empid}") #{e.fname} #{e.lname} - #{e.empid}
br
br
form.form-grid(method="post", action="by_date")
    input.form-grid-input(type="submit", value="Arrange by date")
br
br
    input(type="hidden", name="empid", value="0")
div.table-wrapper
table
    tr
        th Week
        th Emp #
        th Date
        th Time in
        th Time out
        th Hours
    -foreach(s; sheets)
        tr
            td #{s.period}
            td #{s.empid}
            td #{s.thedate}
            td #{s.timein}
            td #{s.timeout}
            td.right-align #{s.hours}

```

Now compile, run, refresh your browser and click on ‘Create timesheet’.

The screenshot shows a web application interface. At the top is a dark green header bar with white text. From left to right, it contains: 'Home', 'Punch card >', 'Employees >', and 'Login'. Below the header is a light blue search form. It has two dropdown menus: 'From' set to '2023-Jan-21' and 'Up to' set to '2023-Jan-22'. Below these is a large, light gray rectangular button with a thin black border, containing the text 'Generate timesheet' in a small, dark font.

Pick a start date and an end date for the time period and click on ‘Generate timesheet’ and you should see something like this:

Timesheets

▾

Week	Emp #	Date	Time in	Time out	Hours
4	0001	2023-Jan-22	08:20:41	10:17:01	1.94
3	0001	2023-Jan-21	12:31:33	14:21:15	1.83
3	0003	2023-Jan-21	15:01:14	15:38:23	0.62
3	0011	2023-Jan-21	15:01:42	15:39:46	0.63
3	1004	2023-Jan-21	12:31:10	15:23:12	2.87

So we were able to generate a timesheet from a given time period.

So here's the whole source\bundycontrol.d so far

```
module Bundycontrol;

import vibe.vibe;
import empmodel;
import cardmodel;
import sheetmodel;

struct User
{
    bool loggedIn;
    string email;
}

class BundyController
{
    private EmployeeModel empModel;
    private TimecardModel cardModel;
    private TimesheetModel sheetModel;
    private string realm = "The Lorem Ipsum Company";
    private SessionVar!(User, "user") m_user;

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

```

sheetModel = new TimesheetModel;
}

void index(string _error = null)
{
    string error = _error;
    m_user = User.init;
    terminateSession;
    render!("index.dt", error);
}

/* start of employee subsystem *****/
@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 != 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, first ~ " " ~ last ~ " not found.");
    render!("employee.dt", e);
}
/* end of employee subsystem *****/
@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;

/* start of timekeeping subsystem *****/
    void getTimeIn()
    {
        Employee[] emps = empModel.getEmployees;
        render!("timein.dt", emps);
    }

    @errorDisplay!getAllCards
    void postTimeIn(string empid)
    {
        string fullname = getfullname(empid);
        //the record should not exist, otherwise employee already punched in earlier
        enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in
today");
        cardModel.timeIn(empid);
        redirect("all_cards");
    }

    string getfullname(string empid)
    {
        Employee e = empModel.getEmployee(empid);
        string fullname = to!string(e.fname) ~ " " ~ to!string(e.lname);
        return fullname;
    }

    void getAllCards(string _error)

```

```

{
    string error = _error;
    bool loggedIn = m_user.loggedIn;
    string today = getToday();
    Timecard[] cards = cardModel.getCards();
    render!("cardlistall.dt", cards, today, error, loggedIn);
}

string getToday()
{
    import std.datetime.date;

    auto time = Clock.currTime;
    return to!string(Date(time.year, time.month, time.day));
}

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

@errorDisplay!getAllCards
void postTimeOut(string empid)
{
    string fullname = getFullscreen(empid);
    //the record should exist, otherwise employee did not punch in earlier
    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");
    cardModel.timeOut(empid);
    redirect("all_cards");
}

void getEditTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getCard(id);
    Employee e = empModel.getEmployeeByEmpid(t.empid);
    render!("cardedit.dt", t, e, error);
}

@errorDisplay!getEditTimecard

```

```

void postEditTimecard(Timecard t)
{
    cardModel.editTimecard(t);
    redirect("all_cards");
}

void getDeleteTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getCard(id);
    Employee e = empModel.getEmployeeByEmpid(t.empid);
    render!("carddelete.dt", t, e, error);
}

@errorDisplay!getDeleteTimecard
void postDeleteTimecard(int id)
{
    cardModel.deleteTimecard(id);
    redirect("all_cards");
}

void getCreateTimesheet(string _error = null)
{
    string error = _error;
    string[] dates = cardModel.getThedates;
    render!("sheetcreate.dt", dates, error);
}

@errorDisplay!getCreateTimesheet
void postCreateTimesheet(string from, string upto)
{
    sheetModel.createTimesheets(from, upto);
    redirect("all_timesheets");
}

void getAllTimesheets()
{
    Timesheet[] sheets = sheetModel.getAllTimesheets;
    Employee[] emps = empModel.getEmployees;
    string[] dates = sheetModel.getThedates;
    render!("sheetlistall.dt", sheets, emps, dates);
}

/* end of timekeeping subsystem *****/
}

```

And here's source\sheetmodel.d so far

```
module sheetmodel;

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

struct Timesheet
{
    int id;
    int period;
    string empid;
    string thedate;
    string timein;
    string timeout;
    double hours;
}

class TimesheetModel
{
    Connection conn;

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

    string[] getThedates()
    {
        string sql = "select distinct thedate from timesheets order by thedate";
        Row[] rows = conn.query(sql).array;
        string[] dates;
        foreach (row; rows) dates ~= to!string(row[0]);
        return dates;
    }

    void createTimesheets(string from, string upto)
    {
        string sql = "delete from timesheets";
        conn.exec(sql);
```

```

        sql = "insert into timesheets(period, empid, thedate, timein, timeout,
hours)
            select week(thedate), empid, thedate, timein, timeout,
round((time_to_sec(timeout)-time_to_sec(timein))/3600,2)
            from timecards where timeout is not null
            and thedate between str_to_date(?,'%Y-%b-%d') and
str_to_date(?,'%Y-%b-%d')
            order by thedate desc, empid";
        Prepared pstmt = conn.prepareStatement(sql);
        pstmt.setArgs(from, upto);
        conn.exec(pstmt);
    }

Timesheet[] getAllTimesheets()
{
    string sql = "select * from timesheets";
    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.thedate = to!string(row[3]);
        t.timein = to!string(row[4]);
        t.timeout = to!string(row[5]);
        t.hours = to!double(to!string(row[6]));
        sheets ~= t;
    }
    return sheets;
}
}

```

Now let's see about custom arrangement of the timesheet data.

Rearranging the timesheet data

We provided two buttons on the list of timesheets, one for filtering the data so only the data for a given employee is shown, and the other for arranging the data by date. These operations are just samples of rearranging data and are somewhat contrived.

Add this code to source\bundycontrol.d

```
void postByEmpid(string empid)
{
    Timesheet[] sheets = sheetModel.getByEmpid(empid);
    Employee e = empModel.getEmployee(empid);
    string fullname = to!string(e.fname) ~ " " ~ to!string(e.lname);
    render!("sheetbyempid.dt", sheets, empid, fullname);
}

void postByDate()
{
    Timesheet[] sheets = sheetModel.getByDate;
    string today = getToday;
    render!("sheetbydate.dt", sheets, today);
}
```

So we defined the two methods in one go, meant for the actions when clicking the two buttons.

Create views\sheetsbyempid.dt

```
extends layout
block maincontent
    include csstable.dt
    h2 Timesheet for #{fullname} - #{empid}
    div.form-grid-wrapper
        form.form-grid(method="get", action="all_timesheets")
            input.form-grid-input(type="submit", value="Back")
    div.table-wrapper
        table
            tr
                th Week
                th Date
                th Time in
                th Time out
                th Hours
                -double totalhours = 0.0;
                -foreach(s; sheets)
                    -totalhours += s.hours;
```

```

tr
td #{s.period}
td #{s.thedate}
td #{s.timein}
td #{s.timeout}
td.right-align #{s.hours}
tr
td.right-align(colspan="5") -----
tr
td.right-align(colspan="4") Total hours:
td.right-align #{totalhours}

```

and create views\sheetbydate.dt

```

extends layout
block maincontent
include csstable.dt
h2 Timesheet as of #{today}
div.form-grid-wrapper
form.form-grid(method="get", action="all_timesheets")
input.form-grid-input(type="submit", value="Back")
div.table-wrapper
table
tr
th Date
th Week
th Emp #
th Time in
th Time out
th Hours
-foreach(s; sheets)
tr
td #{s.thedate}
td #{s.period}
td #{s.empid}
td #{s.timein}
td #{s.timeout}
td.right-align #{s.hours}

```

and add this code to source\sheetsmodel.d

```

Timesheet[] getByEmpid(string empid)
{
    string sql = "select * from timesheets where empid=?";
    Prepared pstmt = conn.prepareStatement(sql);

```

```

 pstmt.setArgs(empid);
 Row[] rows = conn.query(pstmt).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.thedate = to!string(row[3]);
    t.timein = to!string(row[4]);
    t.timeout = to!string(row[5]);
    t.hours = to!double(to!string(row[6]));
    sheets ~= t;
 }
 return sheets;
}

Timesheet[] getByDate()
{
    string sql = "select * from timesheets order by thedate";
    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.thedate = to!string(row[3]);
        t.timein = to!string(row[4]);
        t.timeout = to!string(row[5]);
        t.hours = to!double(to!string(row[6]));
        sheets ~= t;
    }
    return sheets;
}

```

Now compile, run and refresh the browser.

Choose one employee:

Timesheets

[By employee](#) [The Boss - 0001](#)

[Arrange by date](#)

Week	Emp #	Date	Time in	Time out	Hours
4	0001	2023-Jan-22	08:20:41	10:17:01	1.94
3	0001	2023-Jan-21	12:31:33	14:21:15	1.83
3	0003	2023-Jan-21	15:01:14	15:38:23	0.62
3	0011	2023-Jan-21	15:01:42	15:39:46	0.63
3	1004	2023-Jan-21	12:31:10	15:23:12	2.87

And you should see this:

Timesheet for The Boss - 0001

[Back](#)

Week	Date	Time in	Time out	Hours
4	2023-Jan-22	08:20:41	10:17:01	1.94
3	2023-Jan-21	12:31:33	14:21:15	1.83

Total hours: 3.77				

And if you choose ‘Arrange by date’, you should see this

Timesheet as of 2023-Jan-22

[Back](#)

Date	Week	Emp #	Time in	Time out	Hours
2023-Jan-21	3	0001	12:31:33	14:21:15	1.83
2023-Jan-21	3	0003	15:01:14	15:38:23	0.62
2023-Jan-21	3	0011	15:01:42	15:39:46	0.63
2023-Jan-21	3	1004	12:31:10	15:23:12	2.87
2023-Jan-22	4	0001	08:20:41	10:17:01	1.94

And with that, we are done!

All the source code

Here is source\app.d

```
import vibe.vibe;
import bundycontrol;

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 BundyController);

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

    runApplication();
}
```

Here is source\bundycontrol.d

```
module bundycontrol;

import vibe.vibe;
import empmodel;
import cardmodel;
import sheetmodel;

struct User
{
    bool loggedIn;
    string email;
}

class BundyController
{
    private EmployeeModel empModel;
    private TimecardModel cardModel;
    private TimesheetModel sheetModel;
    private string realm = "The Lorem Ipsum Company";
```

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

this()
{
    empModel = new EmployeeModel;
    cardModel = new TimecardModel;
    sheetModel = new TimesheetModel;
}

void index(string _error = null)
{
    string error = _error;
    m_user = User.init;
    terminateSession;
    render!("index.dt", error);
}

/* start of employee subsystem *****/
@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, first ~ " " ~ last ~ " not found.");
    render!("employee.dt", e);
}
/* end of employee subsystem *****/

```

```

@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;

/* start of timekeeping subsystem *****/
void getTimeIn()
{
    Employee[] emps = empModel.getEmployees;
    render!("timein.dt", emps);
}

@errorDisplay!getAllCards
void postTimeIn(string empid)
{
    string fullname = getfullname(empid);
    //the record should not exist, otherwise employee already punched in earlier
    enforce(cardModel.noTimeIn(empid), fullname ~ ", you already punched in
today");
    cardModel.timeIn(empid);
    redirect("all_cards");
}

string getfullname(string empid)
{

```

```

Employee e = empModel.getEmployee(empid);
string fullname = to!string(e.fname) ~ " " ~ to!string(e.lname);
return fullname;
}

void getAllCards(string _error)
{
    string error = _error;
    bool loggedIn = m_user.loggedIn;
    string today = getToday();
    Timecard[] cards = cardModel.getCards();
    render!("cardlistall.dt", cards, today, error, loggedIn);
}

string getToday()
{
    import std.datetime.date;

    auto time = Clock.currTime;
    return to!string(Date(time.year, time.month, time.day));
}

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

@errorDisplay!getAllCards
void postTimeOut(string empid)
{
    string fullname = getfullname(empid);
    //the record should exist, otherwise employee did not punch in earlier
    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");
    cardModel.timeOut(empid);
    redirect("all_cards");
}

void getEditTimecard(int id, string _error = null)
{
    string error = _error;
}

```

```

Timecard t = cardModel.getCard(id);
Employee e = empModel.getEmployeeByEmpid(t.empid);
render!("cardedit.dt", t, e, error);
}

@errorDisplay!getEditTimecard
void postEditTimecard(Timecard t)
{
    cardModel.editTimecard(t);
    redirect("all_cards");
}

void getDeleteTimecard(int id, string _error = null)
{
    string error = _error;
    Timecard t = cardModel.getCard(id);
    Employee e = empModel.getEmployeeByEmpid(t.empid);
    render!("carddelete.dt", t, e, error);
}

@errorDisplay!getDeleteTimecard
void postDeleteTimecard(int id)
{
    cardModel.deleteTimecard(id);
    redirect("all_cards");
}

void getCreateTimesheet(string _error = null)
{
    string error = _error;
    string[] dates = cardModel.getThedates;
    render!("sheetcreate.dt", dates, error);
}

@errorDisplay!getCreateTimesheet
void postCreateTimesheet(string from, string upto)
{
    sheetModel.createTimesheets(from, upto);
    redirect("all_timesheets");
}

void getAllTimesheets()
{
    Timesheet[] sheets = sheetModel.getAllTimesheets;
    Employee[] emps = empModel.getEmployees;
}

```

```

        string[] dates = sheetModel.getTheDates;
        render!("sheetlistall.dt", sheets, emps, dates);
    }

    void postByEmpid(string empid)
    {
        Timesheet[] sheets = sheetModel.getByEmpid(empid);
        Employee e = empModel.getEmployee(empid);
        string fullname = to!string(e.fname) ~ " " ~ to!string(e.lname);
        render!("sheetbyempid.dt", sheets, empid, fullname);
    }

    void postByDate()
    {
        Timesheet[] sheets = sheetModel.getByDate;
        string today = getToday;
        render!("sheetbydate.dt", sheets, today);
    }
/* end of timekeeping subsystem *****/
}

```

Here is source\cardmodel.d

```

module cardmodel;

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

struct Timecard
{
    int id;
    string empid;
    string thedate;
    string timein;
    string timeout;
}

class TimecardModel
{
    Connection conn;

    this()
    {

```

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

void timeIn(string empid)
{
    string sql = "insert into timecards(empid, thedate, timein) values(?, curdate(), curtime())";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(empid);
    conn.exec(pstmt);
}

bool noTimeIn(string empid)
{
    string sql = "select * from timecards where empid=? and thedate=?";
    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]);
}

string[] getThedates()
{
    string sql = "select distinct thedate from timecards order by thedate";
    Row[] rows = conn.query(sql).array;
    string[] dates;
    foreach (row; rows) dates ~= to!string(row[0]);
    return dates;
}

Timecard[] getCards()
{
    Timecard[] cards;
    string sql = "select * from timecards order by thedate desc, timein desc";
    Row[] rows = conn.query(sql).array;
```

```

if(rows.length == 0) return cards;
foreach(row; rows)
{
    Timecard c;
    c.id = to!int(to!string(row[0]));
    c.empid = to!string(row[1]);
    c.thedate = to!string(row[2]);
    c.timein = to!string(row[3]);
    c.timeout = to!string(row[4]);
    cards ~= c;
}
return cards;
}

bool noTimeOut(string empid)
{
    string sql = "select * from timecards where timeout is null and empid=? and thedate=?";
    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;
}

void timeOut(string empid)
{
    string sql = "update timecards set timeout=curtime() where empid=? and thedate=?";
    Prepared pstmt = conn.prepare(sql);
    string today = getToday;
    pstmt.setArgs(empid, today);
    conn.exec(pstmt);
}

Timecard getCard(int id)
{
    Timecard c;
    string sql = "select * from timecards where id=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(id);
    Row[] rows = conn.query(pstmt).array;
    if(rows.length == 0) return c;
    Row row = rows[0];
}

```

```

c.id = to!int(to!string(row[0]));
c.empid = to!string(row[1]);
c.thedate = to!string(row[2]);
c.timein = to!string(row[3]);
c.timeout = to!string(row[4]);
return c;
}

ulong editTimecard(Timecard t)
{
    Prepared pstmt;
    if(to!string(t.timeout) == "null")
    {
        string sql = "update timecards set timein=?, timeout=null where id=?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.id);
    }
    else
    {
        string sql = "update timecards set timein=?, timeout=? where id=?";
        pstmt = conn.prepare(sql);
        pstmt.setArgs(t.timein, t.timeout, t.id);
    }
    return 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);
}
}

```

Here 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",
    "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"]
];

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]);
}

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]);
}

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);
}

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;
}

Employee getEmployeeByEmpid(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 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);
}
*/

```

Here is source\sheetmodel.d

```

module sheetmodel;

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

struct Timesheet

```

```

{
    int id;
    int period;
    string empid;
    string thedate;
    string timein;
    string timeout;
    double hours;
}

class TimesheetModel
{
    Connection conn;

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

    string[] getThedates()
    {
        string sql = "select distinct thedate from timesheets order by thedate";
        Row[] rows = conn.query(sql).array;
        string[] dates;
        foreach (row; rows) dates ~= to!string(row[0]);
        return dates;
    }

    void createTimesheets(string from, string upto)
    {
        string sql = "delete from timesheets";
        conn.exec(sql);
        sql = "insert into timesheets(period, empid, thedate, timein, timeout,
hours)
                select week(thedate), empid, thedate, timein, timeout,
round((time_to_sec(timeout)-time_to_sec(timein))/3600,2)
                from timecards
                where timeout is not null
                and thedate between str_to_date(?, '%Y-%b-%d') and
str_to_date(?, '%Y-%b-%d')
                order by thedate desc, empid";
        Prepared pstmt = conn.prepare(sql);
        pstmt.setArgs(from, upto);
    }
}

```

```
conn.exec(pstmt);
}

Timesheet[] getAllTimesheets()
{
    string sql = "select * from timesheets";
    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.thedate = to!string(row[3]);
        t.timein = to!string(row[4]);
        t.timeout = to!string(row[5]);
        t.hours = to!double(to!string(row[6]));
        sheets ~= t;
    }
    return sheets;
}

Timesheet[] getByEmpid(string empid)
{
    string sql = "select * from timesheets where empid=?";
    Prepared pstmt = conn.prepare(sql);
    pstmt.setArgs(empid);
    Row[] rows = conn.query(pstmt).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.thedate = to!string(row[3]);
        t.timein = to!string(row[4]);
        t.timeout = to!string(row[5]);
        t.hours = to!double(to!string(row[6]));
        sheets ~= t;
    }
    return sheets;
}
```

```

Timesheet[] getByDate()
{
    string sql = "select * from timesheets order by thedate";
    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.thedate = to!string(row[3]);
        t.timein = to!string(row[4]);
        t.timeout = to!string(row[5]);
        t.hours = to!double(to!string(row[6]));
        sheets ~= t;
    }
    return sheets;
}
}

```

Here is views\carddelete.dt

```

extends layout
block maincontent
    include cssformtimecard
    h2 Sure you want to delete #{e.fname} #{e.lname}'s punch card?
    div.form-grid-wrapper
        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 Date :
            label.form-grid-text #{t.thedate}
            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="t_id", value="#{t.id}")
            input(type="hidden", name="id", value="#{t.id}")
            input(type="hidden", name="t_empid", value="#{t.empid}")
            input(type="hidden", name="t_thedate", value="#{t.thedate}")
            input(type="hidden", name="t_timein", value="#{t.timein}")
            input(type="hidden", name="t_timeout", value="#{t.timeout}")
            div
            div

```

```

div
  a(href="all_cards")
    button.form-grid-button(type="button") Cancel
div
  input.form-grid-button(type="submit", value="Delete")

```

Here is views\cardedit.dt

```

extends layout
block maincontent
  include cssformtimecard
  h2 Time card of #{e.fname} #{e.lname} for #{t.thedate}
  div.form-grid-wrapper
    -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 #{e.empid} - #{e.fname} #{e.lname}
      label.form-grid-label Punch in :
      input.form-grid-input(type="text", name="t_timein", value="#{t.timein}")
      label.form-grid-label Punch 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="id", value="#{t.id}")
        input(type="hidden", name="t_empid", value="#{t.empid}")
        input(type="hidden", name="t_thedate", value="#{t.thedate}")
      div
      div
        a(href="all_cards")
          button.form-grid-button(type="button") Cancel
      div
        input.form-grid-button(type="submit", value="Submit")

```

Here is views\cardlistall.dt

```

extends layout
block maincontent
  include csstable.dt
  h2 Time cards as of #{today}
  div.table-wrapper
    -if(error)
      div.error-div
        span.error-message #{error}

```

```

table
  tr
    th Date
    th Emp #
    th Time in
    th Time out
    -if(loggedIn)
      th Action
    -foreach(c; cards)
      tr
        td #{c.thedate}
        td #{c.empid}
        td #{c.timein}
        td #{c.timeout}
        -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.ico", 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.ico", height="15px")
          | &nbs;

```

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\cssformtimecard.dt

```
:css
.form-grid-wrapper
{
    width: 450px;
    height: auto;
    margin: 20px 0;
    padding: 10px;
    border-radius: 10px;
    background-color: #eef;
}
.form-grid
{
    display: grid;
```

```
    grid-template-columns: 1fr 1fr;
    gap: 5px;
}
.form-grid-label
{
    width: 100%;
    font-size: 16px;
    text-align: right;
    padding: 2px;
}
.form-grid-text
{
    width: 100%;
    font-size: 16px;
    font-weight: bold;
    padding: 2px;
}
.form-grid-input
{
    width: 95%;
    font-size: 14px;
    padding: 2px 0 2px 2px;

}
.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: 0;
    padding: 0;
}
```

Here views\csslayout.dt

```
:css
body
{
```

```
    margin: 0;
    padding: 0;
}
.container
{
    display: grid;
    grid-template-rows: 30px auto 30px;
    height: 100%;
    width: 100%;
    margin: 0;
    padding: 0;
}
.menu
{
    margin: 0;
    padding: 0;
}
/*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;
    font-family: Sans-serif;
}
/*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;
}
.error-div
{
    margin: 5px 0;
}
.error-message
{
    color: brown;
    font-weight: bold;
}
.center-align
{
    margin: 0 auto;
    text-align: center;
}
.right-align
{
    text-align: right;
}
.form-hidden
{
    padding: 0;
    margin: 0;
    display: inline;
}
```

Here is views\cssmenu.dt

```
:css
nav
{
    margin: 0 auto;
    padding: 0;
    width: 100%;
    height: auto;
    display: inline-block;
```

```
background: teal; /*#37bc9b;*/
font-family: Verdana, Geneva, Tahoma, sans-serif;
}
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:after { content: ' »'; } /* Change this in order to change the Dropdown symbol */
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: 200px;
}
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;  
}
```

Here is views\csstable.dt

```
:css  
.table-wrapper  
{  
margin: 20px auto;  
}  
table  
{  
padding: 10px 0;  
border-spacing: 0;  
border-collapse: collapse;  
}  
table td, table th  
{  
margin: 0;  
padding: 2px 10px;  
}  
.no-border  
{  
border: 0;  
}  
tr:nth-child(odd)  
{  
background: whitesmoke;  
}
```

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:
            span.form-grid-field #{e.city}
            span.form-grid-label Province:
            span.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.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")

```

Here is views\emplistall.dt

```

extends layout
block maincontent
    include csstable
    -if(error)
        div.error-div
            span.error-message #{error}
    h2 Employees of The Lorem Ipsum Company
    div.table-wrapper
        table
            tr
                th First name
                th Last name

```

```

th Department
th Phone number
th Email address
th Action
-foreach(e; emps)
  tr
    td #{e.fname}
    td #{e.lname}
    td #{e.deprt}
    td #{e.phone}
    td #{e.email}
    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;

```

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}

```

```

.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\footer.dt



```
div.copyright Copyright © The Lorem Ipsum Company 2023
```



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.


```



Here is views\layout.dt



```

doctype 5
html
 head

```


```

```

title Employee Timekeeping System
include csslayout.dt
body
  div.container
    div.menu
      include menu.dt
    div.content
      block maincontent
    div.footer
      include footer.dt

```

Here is views\menu.dt

```

include cssmenu.dt
nav
  ul
    li
      a(href="/") Home
    li
      a(href="#") Punch card
      ul
        li
          a(href="time_in") Punch in
        li
          a(href="time_out") Punch out
        li
          a(href="all_cards") View cards
        li
          a(href="create_timesheet") Create timesheet
    li
      a(href="#") Employees
      ul
        li
          a(href="all_employees") Show all employees
        li
          a(href="#find_employee") Find an employee
        li
          a(href="add_employee") Add new employee
  ul.link-right
    li
      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")
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\sheetbydate.dt

```

extends layout
block maincontent
include csstable.dt
h2 Timesheet as of #{today}
div.form-grid-wrapper
form.form-grid(method="get", action="all_timesheets")
    input.form-grid-input(type="submit", value="Back")
div.table-wrapper
table
tr
th Date
th Week
th Emp #
th Time in
th Time out
th Hours
-foreach(s; sheets)
tr
td #{s.thedate}
td #{s.period}
td #{s.empid}
td #{s.timein}
td #{s.timeout}
td.right-align #{s.hours}

```

Here is views\sheebyempid.dt

```
extends layout
block maincontent
  include csstable.dt
  h2 Timesheet for #{fullname} - #{empid}
  div.form-grid-wrapper
    form.form-grid(method="get", action="all_timesheets")
      input.form-grid-input(type="submit", value="Back")
  div.table-wrapper
    table
      tr
        th Week
        th Date
        th Time in
        th Time out
        th Hours
      -double totalhours = 0.0;
      -foreach(s; sheets)
        -totalhours += s.hours;
      tr
        td #{s.period}
        td #{s.thedate}
        td #{s.timein}
        td #{s.timeout}
        td.right-align #{s.hours}
      tr
        td.right-align(colspan="5") -----
      tr
        td.right-align(colspan="4") Total hours:
        td.right-align #{totalhours}
```

Here is views\sheetscreate.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")
      label.form-grid-label From:
      select#from.form-grid-input(name="from")
        -foreach(d; dates)
```

```

        option(value="#{d}") #{d}
    div
    div
    label.form-grid-label Up to:
    select#from.form-grid-input(name="upto")
        -foreach(d; dates)
            option(value="#{d}") #{d}
    div
    div
        input.form-grid-button(type="submit", value="Generate timesheet")

```

Here is views\sheetlistall.dt

```

extends layout
block maincontent
    include csstable.dt
    h2 Timesheets
    div.form-grid-wrapper
        form.form-grid(method="post", action="by_empid")
            input.form-grid-input(type="submit", value="By employee")
            select.form-grid-input(name="empid")
                -foreach(e; emps)
                    option(value="#{e.empid}") #{e.fname} #{e.lname} - #{e.empid}
            br
            br
        form.form-grid(method="post", action="by_date")
            input.form-grid-input(type="submit", value="Arrange by date")
            br
            br
            input(type="hidden", name="empid", value="0")
    div.table-wrapper
        table
            tr
                th Week
                th Emp #
                th Date
                th Time in
                th Time out
                th Hours
            -foreach(s; sheets)
                tr
                    td #{s.period}
                    td #{s.empid}
                    td #{s.thedate}
                    td #{s.timein}

```

```
    td #{s.timeout}
    td.right-align #{s.hours}
```

Here is views\timein.dt

```
extends layout
block maincontent
  include cssformgrid.dt
  div.form-grid-wrapper
    br
    h1#time.center-align
    br
    form.form-grid(method="post", action="time_in")
      label.form-grid-label Employee:
      select#empid.form-grid-input(name="empid")
        -foreach(e; emps)
          option(value="#{e.empid}") #{e.empid} - #{e.fname} #{e.lname}
      div
      div
        input.form-grid-button(type="submit", value="Punch in!")
:javascript
  const displayTime = document.querySelector("#time");
  function showTime() {
    let time = new Date();
    displayTime.innerText = time.toLocaleString("en-CA", { hour12: true });
    setTimeout(showTime, 1000);
  }
  showTime();
```

Here is views\timeout.dt

```
extends layout
block maincontent
  include cssformgrid.dt
  div.form-grid-wrapper
    br
    h1#time.center-align
    br
    form.form-grid(method="post", action="time_out")
      label.form-grid-label Employee:
      select#empid.form-grid-input(name="empid")
        -foreach(e; emps)
          option(value="#{e.empid}") #{e.empid} - #{e.fname} #{e.lname}
      div
      div
```

```
    input.form-grid-button(type="submit", value="Punch out!")
:javascript
    const displayTime = document.querySelector("#time");
    function showTime() {
        let time = new Date();
        displayTime.innerText = time.toLocaleString("en-CA", { hour12: true });
        setTimeout(showTime, 1000);
    }
    showTime();
```

And here is, at last, 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": "bundy"
}
```

That's it! Yaaaaay!