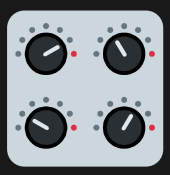


1.2 - Controllers and Layouts

During skills lab:

Do the following assignments during the skills lab and bring the results to the Checkpoint. You can do these assignment either in your project or build upon the provided snapshot, which already includes all the features of yesterdays assignments:

laravel-taskiteasy-snapshot-1-2-0.zip 755.2KB

1.2.A - Design the Routing Code Structure

👥 Work in groups of 3-5 students.

During the lecture you saw that it is not convenient to have all routes handled in one big `web.php` file. We introduced Controllers as the main concept for this (and also mentioned Single Action Controllers and View Routes) and showed you that you need to decide on how to group these routes into one or more Controllers. The basics are discussed during class, but you can find the related theory in this knowledge bank article:

🗺️ Routing

🛠️ [±5 min.] - Do this individually, without talking to your group members - Create a **route list table** design for all **3 routes** of the *TaskITEasy* app that shows which handlers you have chosen. Try to find logical names for you

The design rules for this are:

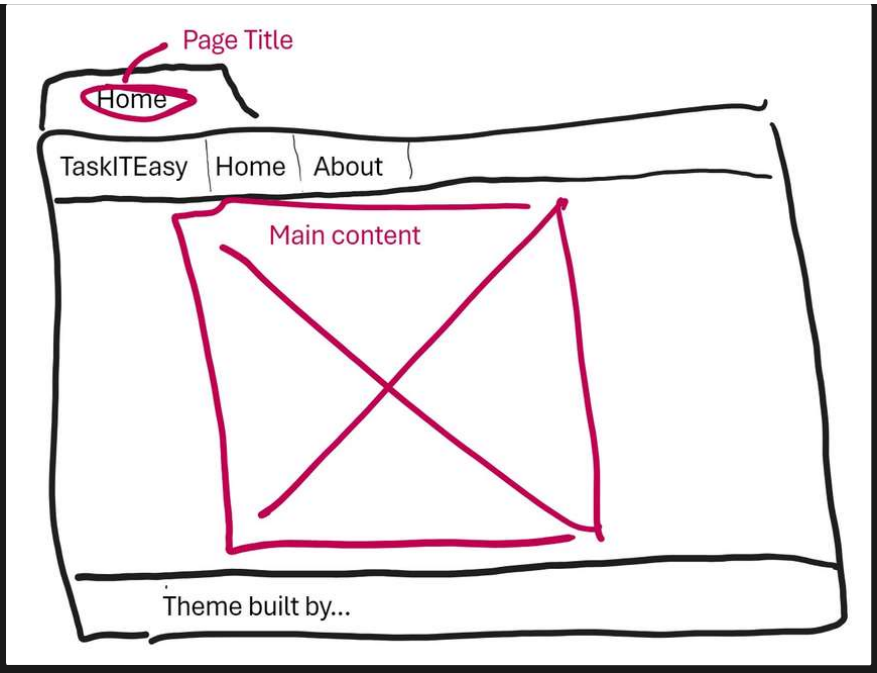
- No closures for any routes. So, no: `Route::get('...', function() { ... });` in your `web.php`
- High cohesion: a Controller should handle routes about one 'thing'
- Meaningful naming: Controllers and controller methods must have meaningful naming
- Framework Naming Conventions: which naming conventions apply to this?
- Modifiability: adding new routes for, i.e. adding and editing tasks or other pages like a contact page

🛠️ [±15 minutes] - Within your group, compare the different results with respect to the design rules. Try to come to a 'best result' route list table. But also write down discussion points when they occur. Discuss and ask questions to each other and/or the teachers. Bring the result to the Checkpoint

1.2.B - Crafting the Layout Foundation

👥 Work in groups of 3-5 students.

During the lecture you learned that it is important for your applications internal consistency to build your app around one or more page layouts. You are going to implement this in your *TaskITEasy* implementation. The following simple sketch (called a *lo-fi wireframe*) shows the *generic page structure* and highlights parts where *page specific content* needs to be injected.



Lo-fi wireframe of the TaskITEasy layout design

You also learned that Laravel Blade has 2 different technologies that you can use to implement this: Blade Components and Template inheritance. If needed, you can read about his in the following Knowledge Band Article:

Building Layouts

In this part, you get the opportunity to discover both technologies and select the one you want to use in your applications. While working collaboratively in your group, you have the ability to adopt a dual approach. Some members can focus on Blade Components while the others work with Template Inheritance. Later, during group discussions, you can compare and contrast the outcomes of both approaches. This dual strategy encourages a dynamic exchange of ideas, promotes a comprehensive understanding of the material, and provides valuable insights into different coding methodologies within the group.

🛠️ [±10 min.] - Familiarize yourself with both technologies by reading the *Building Layouts* knowledge bank article in the link below. Note that using template inheritance is considered a digging deeper subject, but you can at least familiarize yourself with the concept. Decide on if you already prefer one.

Building Layouts

🛠️ [±5 min.] - Discuss both technologies in your group. Explain uncertainties and questions to each other and/or ask one of the teachers for help. Then divide your group into two teams: one that will experiment with Layout Components and one that will experiment with Template Inheritance.

Now you can conduct experiments with both technologies to compare them and see what the pros and cons are of each technology. Each team will convene to collectively decide which version will be used as the foundation for an upcoming experiment. Consider the strengths embedded in each member's version. You may also choose to use the version we provided in prior to today's class. With this project you will conduct the following experiments as a team. Which means that the entire team is actively involved with the programming, a bit like *pair-programming*.

First, take the following design decisions:

🛠️ [±5 min.] - Decide on the names of the slots/sections required for the page specific content in the layout.

Then rework the home page first so it uses the layout:

🛠️ [±15 min.] - Create the layout file by copy-and-pasting the code of the home view. Then replace all the page specific content and replace it with the appropriate structures for injecting. Then rework the home page so it utilizes the layout and put the page specific content into the appropriate slots/sections. Test and debug until you see the correct content. One of the team members takes notes on possible difficulties

Then rework the other pages

🛠️ [±10 min.] - Rework the `about` and `tasks.show` pages so they utilize the same layout

You should encounter at least one difficulty here: the navigation menu. The `is-active` class needs to be assigned to different menu items pending on the specific page.

🛠️ [±10 min.] - Discuss this problem and possible solutions. Experiment with this in your prototype. **Tip**: you should need to access the current route, see:

URL generation

🛠️ [±10 min.] - Start this not later than 10 minutes before Checkpoint starts. Compare both outcomes and discuss pros and cons of both technologies. Create a list of pros and cons, discussion points and further questions and bring this to the checkpoint

Checkpoint

Visit the checkpoint. Bring along the following:

- The 'best result' route list table and your own individually created one. Accompanied by discussion points and/or questions when they occurred.
- The prototypes your team created for both Layout Components and Template Inheritance technologies. Each group member should at least have access to both projects
- The list of pros and cons of Layout Components and Template Inheritance, discussion points and further questions

After the Checkpoint

You can experiment further with your own project.

1.2.C - Catching up: Implementing previously missed features

👤 This is an individual assignment

In this part, you have the flexibility to either implement any missing features from yesterdays skill assignments in your project or build upon the provided snapshot at the top of this document, which already includes all the features of yesterdays assignments.

🛠️ Make your project ready for the following assignments. Check if all the features are present and bug free. If not, choose to either finish your project or use the provided snapshot by copying the required view files and the content of `web.php` to your project

1.2.D - Implement the Controller structure

👤 This is an individual assignment

In this part you will rework the `web.php` so that it only contains route declarations and all closures and other functions are removed from that file.

> 🛠️ Revise, if needed, your route list table. Remember the discussions during skills lab and/or checkpoint. Maybe (re)consider Single Action Controllers and/or View Routes

All closures and other functions need to move to one of your designed Controllers. So, it's best to create them first so that you can copy and paste code. Remember that you can use the `php artisan make:controller` command for this

> 🛠️ Create the required controllers. Take care of the naming and case style (PascalCase) for both file name and class name

Now you can rework all the routes to your design. When using normal controllers, you should:

1. Create the designed method in the fitting Controller. Leave it empty for now
2. Go to your `web.php` and copy the content of the closure that you want to replace into the method you created before
3. Replace the closure in `web.php` with the appropriate definition so it will call the method (i.e. `[FooController::class, 'show']`. Test and debug until you are sure it works

> 🛠️ Rework all 3 routes one at a time using the procedure above. Make sure to test each route before you move on to the next. Ask a teacher or a student assistant if something goes wrong and/or it takes longer than 15 min. to get it working.

1.2.E - Implement the layout design

👤 This is an individual assignment

Finally, you can experiment on your own with your views and layouts. Her you can use the results of the experiments you did with your group during the skills lab.

> 🛠️ Create the layout structure using the technology of your choice. Rework until all 3 views use the same layout file. Don't forget the navigation menu including the `is-active` class

1.2.F - Spicing up

Following are some suggestions how to spice up your app. Consider these as extra challenges. Choose for yourself how many and which of these you MAY want to implement.

Dynamic navbar

Spend some time to get to know Laravel Components and the Bulma Navbar. It would be nice if your layout contains a 'dynamic' navigation menu like:

```
<div class="navbar-start"> @foreach($navItems as $navItem) <x-navbar-item
:item="$navItem"></x-navbar-item> @endforeach </div>
```

Responsive navbar

Bulma Navbar can also be made more responsive to make it optimal for both desktop as mobile browsers. Experiment with Laravel Components and Bulma to make an advanced Navbar that might include the dynamic menu and/or responsiveness.

More Bulma

Spend some more time to get to know the Bulma and Font Awesome CSS libraries. Experiment with Bulma (and may play with asset compilation if you like) and Font Awesome. Use the `tasks.show` and/or `welcome` views for this by creating fancy ways of showing each of the tasks attributes. Some inspiration (but feel free to look around the Bulma site and use your own):

- Tags with icons for task state and priority
- a Panel, Card or Message as a container for the task data. With a color that represents the state
- Progress bars for estimated and spent times
- Add icons to your navigation menu
- Tabs to group some attributes
- Media Object or Tiles to show each task in the list