PRADITA University

IF140303-Web Application Development

Session-08: Phoenix's MVC Model

PRU/SPMI/FR-BM-18/0222

Alfa Yohannis

Phoenix MVC Model: Model, View, Controller

PRADITA University

- Model: Represents the data and business logic.
- View: Handles the presentation layer.
- Controller: Manages the flow of the application.

Model, View, Controller Workflow

PRADITA University

- Request received by the router.
- Router directs the request to the appropriate controller.
- Controller interacts with the model and selects a view.
- View renders the final output.

Understanding Phoenix Project Structure

PRADITA University

- `controllers/`: Manages request handling.
- `models/`: Represents database tables and business logic.
- `views/`: Manages how data is presented.

Request Workflow: Router, Controller, View

PRADITA University

- Request hits the router.
- Router directs to the controller.
- Controller processes the request, uses model, and chooses view.
- View generates HTML or JSON response.

Templates vs. Views

PRADITA University

- Templates: HTML or eEx files used for rendering.
- Views: Elixir modules that process data and select templates.
- Naming conventions are crucial for Phoenix to locate files.

Starting Phoenix with IEx

PRADITA University

- In Elixir: `iex -S mix`.
- In Phoenix: `iex -S mix phx.server`.

Phoenix Models and Database

PRADITA University

- Models represent database tables.
- Phoenix needs to be told what data is in the database.
- Models define the structure of database tables.

Creating a Topics Model

PRADITA University

```
defmodule Discuss.Topic do
use Discuss.Web, :model

schema "topics" do
field :title, :string
belongs_to :user, Discuss.User
has_many :comments, Discuss.Comment
end

def changeset(struct, params \\ %{}) do
struct
|> cast(params, [:title])
|> validate_required([:title])
end
end
```

Understanding Migrations

PRADITA University

- Migrations manage changes to your database schema.
- Command: `mix ecto.gen.migration add_topics`.
- Files stored in `priv/repo/migrations`.
- Migration filenames include a timestamp.

Creating a Topics Migration

PRADITA University

```
defmodule Discuss.Repo.Migrations.AddTopics do
use Ecto.Migration

def change do
create table(:topics) do
add :title, :string
end
end
end
```

Using GUI Databases for PostgreSQL

PRADITA University

- GUI tools like Postico or pgAdmin can simplify database management.
- These tools provide a user-friendly interface for interacting with PostgreSQL.

Project Problems

PRADITA University

Problem

Need a new URL for the user to visit

New routes must map up to a method in a controller

Need to show a form to the user

Need to translate data in the form to something that can be saved

The controller and view are related to 'Page', but we are making stuff related to 'Topic'

Project Solutions

PRADITA University

Solution

Add a new route in router file

Add a new method in a controller to handle this request

Make a new template that contains the form

Create a 'Topic' model that can translate raw data into something that can be saved

Make a new controller and view to handle everything related to topics

Handling Requests in Router

PRADITA University

- Ensure request handling in the router.
- Example code:
-

```
 scope "/", Discuss do
pipe_through :browser
get "/", PageController, :index
get "/topics/new", TopicController, :new
end
```

Controller Function Name Conventions

PRADITA University

- `new`: Render form for new resource.
- `create`: Handle form submission and create resource.
- `index`: List all resources.
- `delete`: Remove a resource.
- `edit`: Render form to edit resource.
- `update`: Handle form submission and update resource.

Creating a Simple TopicController

PRADITA University

```
defmodule Discuss.TopicController do
use Discuss.Web, :controller
def new(conn, params) do
end
end
```

Fixing the Init Error

PRADITA University

- Error: function Discuss.TopicController.init/1 is undefined or private.
- Solution: Add `use Discuss.Web, :controller` to include required functions.
- Example:

```
 defmodule Discuss.TopicController do
use Discuss.Web, :controller
def new(conn, params) do
end
end
```

Elixir Keywords: `import`, `alias`, `use`

PRADITA University

- `import`: Bring functions or macros into scope.
- `alias`: Create shorter names for modules.
- `use`: Inject functionality into a module.

Role of `web.ex` in Phoenix

PRADITA University

- `web.ex`: Contains functionality for models, controllers, and views.
- `use Discuss.Web, :controller`: Imports default controller functions and setup.

Content of controller in `web.ex`

PRADITA University

```
def controller do
quote do
use Phoenix.Controller
alias Discuss.Repo
import Ecto
import Ecto.Query
import Discuss.Router.Helpers
import Discuss.Gettext
end
end
```

Eliminating Errors with `use Discuss.Web, :controller`

PRADITA University

- Adding `use Discuss.Web, :controller` resolves init function errors. Because the init function are included in it
- Example:

```
 defmodule Discuss.TopicController do
use Discuss.Web, :controller
def new(conn, params) do
end
end
```

Function Signature Mismatch Errors

PRADITA University

- Error format: `<function_name>/<number>`
- Example: `new/0` indicates the function `new` takes 0 arguments.

Debugging with `IO.inspect`, `conn`, and `params`

PRADITA University

- `IO.inspect`: Prints values to the console for debugging.
- `conn`: Contains information about the request and response.
- `params`: Contains query parameters and form data.

Modifying TopicController to Use `conn` and `params`

PRADITA University

```
[language=Elixir]
   defmodule Discuss.TopicController do
   use Discuss.Web, :controller
   def new(conn, params) do
   IO.inspect(conn)
   IO.inspect(params)
   end
   end
```

Form Validation: Title Required

PRADITA University

- Ensure that form input has a title.
- Show error message: ”You must enter a title” if missing.