PRADITA University

IF140303-Web Application Development

Session-14: Phoenix v1.3 Changes

PRU/SPMI/FR-BM-18/0222

Alfa Yohannis

Phoenix v1.3 Changes: Overview

PRADITA University

- Phoenix v1.3 introduces significant changes compared to v1.2, making it more powerful but also more complex.
- v1.2 is generally easier to learn due to a simpler structure and fewer abstractions.
- Major changes include an updated directory structure, new CLI commands, and the introduction of contexts.

Generating a New Project in Phoenix v1.3

PRADITA University

- To generate a new project in Phoenix v1.3, use the command:
- `mix phx.new project_name`
- This creates a new folder with the updated directory structure.
- The structure is more modular, promoting a clearer separation of concerns.
- Example: Controllers, views, and templates are now in separate folders, enhancing maintainability.

CLI Command Changes in Phoenix v1.3

- Phoenix v1.3 introduces new CLI commands and changes existing ones.
- Many commands that started with `phoenix` in v1.2 now start with `phx`.
- Example: `mix phoenix.server` is now `mix phx.server`.
- This change aims to standardize the naming conventions and align with Elixir’s overall ecosystem.

Client Assets Directory Separation

- In Phoenix v1.2, client assets were stored within the `web` folder.
- Phoenix v1.3 introduces a separate `assets` folder outside of the `web` directory.
- The `assets` folder contains everything related to front-end assets, including a new `static` subfolder.
- This separation improves the clarity of the project structure and better supports modern front-end development workflows.

Web Folder Split in Phoenix v1.3

- The `web` folder from Phoenix v1.2 is split into two parts in v1.3:
- `update` and `update_web`.
- In v1.2, the `web` folder contained controllers, views, templates, and models.
- In v1.3, `update` contains core business logic, while `update_web` contains web-specific code like controllers and views.
- This separation enhances code organization and makes large projects more manageable.

Understanding Contexts in Phoenix v1.3

- Contexts are a new concept in Phoenix v1.3 that group related functionality.
- Example contexts in our application could be `Accounts` for user management and `Posts` for topic and comment management.
- Contexts provide a layer of abstraction, but they can also introduce complexity, especially when linking different contexts together.
- In contrast, Phoenix v1.2 used a simpler, more direct approach to organizing code.

Generating HTML with Phoenix v1.3 Commands

- Phoenix v1.3 introduces new commands for generating HTML resources:
- `mix phx.gen.html Accounts User users email:string`
- Generates a schema, migration, controller, view, and templates for managing users.
- The files are placed in the appropriate folders based on the new structure.

Generating HTML for Discussions and Comments

- To generate resources for discussions:
- `mix phx.gen.html Discussions Topic topics title:string`
- For comments within discussions:
- `mix phx.gen.html Discussions Comment comments content:string`
- These commands create all necessary files, organized within the context.

Models are Now Called Schemas in Phoenix v1.3

- In Phoenix v1.3, what was referred to as a ”model” in v1.2 is now called a ”schema”.
- A schema defines the structure of your data and corresponds to a database table.
- This renaming better reflects the purpose and functionality of these modules within the Elixir ecosystem.

Summary

- Phoenix v1.3 introduces several significant changes, including new project structures, updated CLI commands, and the introduction of contexts.
- Client assets are now in a separate `assets` folder, improving project organization.
- The web folder is split into `update` and `update_web`, enhancing code modularity.
- The new concept of contexts in v1.3 groups related functionality, but with added complexity.