CVBuilder: A Swift Package for Professional CV Management

CVBuilder is a powerful Swift package that I developed to solve a common problem: managing and rendering professional CVs in multiple formats. Whether you need to generate a Markdown CV for GitHub, an HTML version for your website, or a plain text version for job applications, CVBuilder has you covered.

Why CVBuilder?

In today's digital world, professionals often need to maintain their CVs in multiple formats and keep them in sync. Traditional word processors make this process tedious and error-prone. CVBuilder solves this by:

- Providing a strongly-typed data model for CVs
- Supporting multiple output formats (Markdown, HTML, plain text)
- Making it easy to maintain a single source of truth
- Enabling programmatic CV generation and updates

Core Features

1. Strongly Typed Data Model

CVBuilder provides a comprehensive data model that covers all aspects of a professional CV:

```
CV
??? ContactInfo
??? Education
??? WorkExperience
?   ??? Company
?   ??? Role
?   ??? ProjectExperience
?       ??? Project
?       ??? Tech
??? Period
```

Each component is strongly typed, ensuring data integrity and making it impossible to create invalid CVs.

2. Multiple Renderers

CVBuilder supports three different renderers out of the box:

1. **MarkdownCVRenderer**: Generates clean, well-formatted Markdown
2. **StringCVRenderer**: Produces plain text output
3. **IgniteRenderer**: Creates beautiful HTML using the Ignite static site generator

3. Modular Architecture

The package is split into two main components:

`CVBuilder`: Core types and basic rendering
`CVBuilderIgnite`: Optional HTML rendering support

This separation ensures that projects that don't need HTML rendering don't have to pull in unnecessary dependencies.

Getting Started

Installation

Add CVBuilder to your project using Swift Package Manager:

```
.package(url: "https://codeberg.org/Mihaela/cvbuilder.git", branch: "main")
```

Then add the desired product:

```
.product(name: "CVBuilder", package: "cvbuilder"),
.product(name: "CVBuilderIgnite", package: "cvbuilder") // if using Ignite
```

Creating a CV

Here's a simple example of creating a CV:

```
import CVBuilder

// Create contact info
let contactInfo = ContactInfo(
    email: "jane.doe@example.com",
    phone: "+1 (555) 123-4567",
    linkedIn: URL(string: "https://linkedin.com/in/janedoe"),
    github: URL(string: "https://github.com/janedoe"),
    website: URL(string: "https://janedoe.dev"),
    location: "San Francisco, CA"
)

// Create education
let education = Education(
    institution: "Stanford University",
    degree: "B.S.",
    field: "Computer Science",
    period: Period(
        start: .init(month: 9, year: 2010),
        end: .init(month: 6, year: 2014)
    )
)

// Create the CV
let cv = CV.create(
    name: "Jane Doe",
    title: "Senior Mobile Developer",
    summary: "Passionate mobile developer with 5+ years experience...",
    contactInfo: contactInfo,
    education: [education],
    projects: [project1, project2]
)
```

Generating Output

Once you have your CV data model, generating different formats is straightforward:

```
// Generate Markdown
let markdown = MarkdownCVRenderer(cv: cv).render()

// Generate plain text
let plainText = StringCVRenderer(cv: cv).render()

// Generate HTML (requires CVBuilderIgnite)
let html = IgniteRenderer(cv: cv).render()
```

Advanced Features

Project Builder Pattern

CVBuilder includes a builder pattern for creating project experiences:

```
let project = try! Project.Builder()
    .withName("iOS App Redesign")
    .withCompany(appleCompany)
    .withRole(seniorIOS)
    .withPeriod(start: (month: 3, year: 2020), end: (month: 9, year: 2021))
    .addDescription("Led a team of 5 developers...")
    .withTechs([swift, swiftUI])
    .build()
```

Tech Stack Management

The package includes a `Tech` type for managing technical skills:

```
let swift = Tech(name: "Swift", category: .language)
let swiftUI = Tech(name: "SwiftUI", category: .framework)
let restAPI = Tech(name: "REST API", category: .concept)
```

How I Use It on This Webpage

My blog is built using Ignite, a static site generator for Swift. I've integrated CVBuilder into my website to maintain and display my CV in a clean, professional format.

Package Dependencies

I include CVBuilder in my website's package file:

```
let package = Package(
    name: "IgniteStarter",
    platforms: [.macOS(.v13)],
    dependencies: [
        .package(url: "https://github.com/twostraws/Ignite.git", branch:
"main"),
        .package(url: "https://codeberg.org/Mihaela/cvbuilder.git", branch:
"main")
    ],
    targets: [
        .executableTarget(
            name: "IgniteStarter",
            dependencies: [
                .product(name: "Ignite", package: "Ignite"),
                .product(name: "CVBuilder", package: "cvbuilder"),
                .product(name: "CVBuilderIgnite", package: "cvbuilder")
```

```
            ]
        ),
    ]
)
```

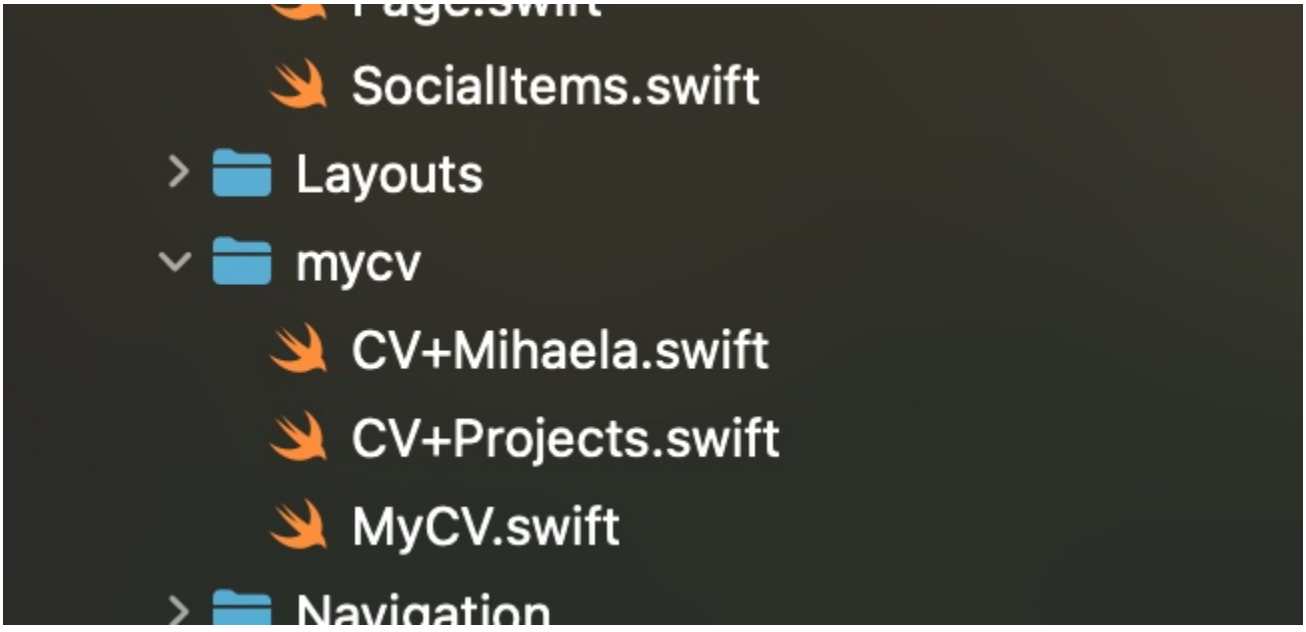
Personal Information

I maintain my personal information in a dedicated Swift file (`CV+Mihaela.swift`):

```
public extension CV {
    static func createForMihaela() -> CV {
        let contactInfo = ContactInfo(
            email: "me@me.com",
            phone: "+12233445566",
            linkedIn: URL(string: "https://www.linkedin.com/in/mylinkedin"),
            github: URL(string: "https://github.com/mygithub"),
            website: URL(string: "https://mywebsite.com"),
            location: "City, Country"
        )

        let education = Education(...)

        // ... rest of the implementation
    }
}
```



Project History

My professional experience is organized in a separate file (`CV+Projects.swift`), which contains detailed information about all my projects:

```
public extension CV {
```

```
    static func createMihaelasProjects() -> [Project] {
        var result = [Project]()

        // Create companies
        let undabot = Company(name: "Undabot")
        let token = Company(name: "Token")
        // ... more companies

        // Create tech skills
        let swift = Tech(name: "Swift", category: .language)
        let uiKit = Tech(name: "UIKit", category: .framework)
        // ... more tech skills

        // Create projects using the builder pattern
        let project1 = try! Project.Builder()
            .withName("SomeProject")
            .withCompany(Company name)
            .withRole(juniorIOS)
            .withPeriod(start: (month: 9, year: 20XX), end: (month: 12, year:
20XX))
            .addDescription("iOS (iPad) book application about...)
            .withTechs([swift, uiKit])
            .build()

        // ... more projects

        return result
    }
}
```

This modular approach allows me to:

1. Keep my CV data in a strongly-typed format
2. Easily update my experience and skills
3. Generate both HTML and Markdown versions of my CV
4. Maintain consistency across different platforms

The CV is automatically generated when the website is built, ensuring that my professional information is always up-to-date and consistently formatted.

Markdown Generation

I also generate a Markdown version of my CV that I can use to create a fresh PDF whenever needed. This is done using a simple function:

```
func generateMihaelasCVMarkdownInContentFolder() {
    let cv = CV.createForMihaela()
    let markdown = MarkdownCVRenderer().render(cv: cv)

    let fileURL = URL(fileURLWithPath: "Assets/mihaela-cv.md")

    do {
        try markdown.write(to: fileURL, atomically: true, encoding: .utf8)
```

```
        print("? Written to \(fileURL.path)")
    } catch {
        print("? Failed to write Mihaela's CV: \(error.localizedDescription)")
    }
}
```

This function:

1. Creates my CV using the `createForMihaela()` function
2. Renders it to Markdown using `MarkdownCVRenderer`
3. Saves the output to `Assets/mihaela-cv.md`
4. Provides feedback about the success or failure of the operation

I can then use this Markdown file to generate a fresh PDF whenever I need to update my CV for job applications or other purposes.

Future Plans

CVBuilder is actively maintained and has several planned improvements:

1. Command-line interface for CV generation
2. Additional renderers (PDF, LaTeX)
3. Import/export functionality for common formats
4. Template system for customizing output

Conclusion

CVBuilder provides a robust solution for managing professional CVs in Swift. Its type-safe design, multiple renderers, and modular architecture make it an excellent choice for developers who want to maintain their CVs programmatically.

Whether you're building a personal website, managing multiple CV versions, or creating a CV management system, CVBuilder can help streamline your workflow and ensure consistency across all your professional documents.

Resources

- Codeberg Repository
- Sample CV
- Documentation