We are very excited today to formally announce the availability of ColdBox Elixir, coldbox/elixir. ColdBox Elixir provides a clean, fluent API for defining basic Gulp tasks for your ColdBox applications.

This project was forked from the Laravel Elixir project, so many many thanks for all their hard work and ideas.

Overview

We wanted to provide ColdBox developers and easy way to approach an asset pipeline and task executor. The elixir project accomplished this and we have extended it to match the conventions used in ColdBox but also took it a step further to make it compatible with TestBox and CommandBox. This way you can have a full task runner available to you than can integrate with CommandBox recipes, package management,
command execution and also TestBox executions.

**How it works**

Elixir supports several common CSS, JavaScript pre-processors, and TestBox runner integrations. By leveraging Gulp and a `Gulpfile.js` configuration file in the root of your application, you can use method chaining and Elixir will allow you to fluently define your asset pipeline using conventions.

All resources in your ColdBox application will be stored under the `resources/assets` folder. Under this folder you can find several sub-sections:

- **css** - Where you can store your css
- **js** - Where you can store your js, vue.js and more
- **less** - Where you can store your less files
- **sass** - Where you can store your sass files

Depending on the formulas you mix up in the `Gulpfile.js` using Elixir, you will end up with all your assets linted, minified, etc in their appropriate destination in the `includes` folder:

- **css** - Destination for css
- **js** - Destination for js
- **build** - Destination for versioned assets

For example:

```javascript
var elixir = require( 'coldbox-elixir' );
elixir( function( mix ) {
    // Look in the 'resources/sass' folder
    mix.sass( 'app.scss' )
    // Look in the 'resourcess/css' folder
    .styles( 'modules.css' );

    // Elixir will then output all assets to ColdBox 'includes' folder by convention.
```
Fully Documented

We are serious about our software, but we are also serious about our documentation. As with any product we do at Ortus, it is fully documented and a gitbook has been created: ColdBox Elixir Documentation

![GitBook](https://github.com/coldbox-templates/coldbox-elixir)

Welcome to ColdBox Elixir

ColdBox Elixir

ColdBox Elixir provides a clean, fluent API for defining conventions based asset and task pipeline for your ColdBox MVC applications based on Gulp. — Ortus Solutions

ColdBox Application Templates

We have created a new Github organization: coldbox-templates

[https://github.com/coldbox-templates](https://github.com/coldbox-templates) to host all the new ColdBox application templates. You can also collaborate and create new ones if desired. All application templates will now include the ability to use elixir. Here are the links to those:

- elixir
- elixir-bower
- elixir-vuejs

Manual Installation

ColdBox Elixir is a node package and can be installed by a simple command on any project: `npm install coldbox-elixir`. You can then create your `Gulpfile.js` and
start mixing your assets.

**Extensibility**

ColdBox Elixir is fully customizable and extensible. Please check out the documentation for further information on extensibility: [http://coldbox-elixir.ortusbooks.com/writing_elixir_extensions.html](http://coldbox-elixir.ortusbooks.com/writing_elixir_extensions.html)