Input Validation

Luminus uses Struct as the default validation library. Struct is a Clojure/Script library and allows us to share validation logic between the client and the server.

Struct provides `struct.core/validate` and `struct.core/valid?` functions for handling validation.

Before we see how validation works, let's include `struct.core` in our namespace.

```
(ns myapp.home
  (:require
    ...
    [struct.core :as st]))
```

Next, we'll define a validation schema for our data using the helpers from the `struct.core` namespace:

```
(def album-schema
  [[:band st/required st/string]
   [:album st/required st/string]
   [:year st/required st/number]])
```

We can now validate the data using the schema as follows:

```
(st/validate {:band "MONO" :album "Hymn to the Immortal Wind" :year 20
;; => [nil {:band "MONO :album "Hymn to the Immortal Wind" :year 2009}

(st/validate {:band "MONO" :album "Hymn to the Immortal Wind" :year "2
;; => [{:year "must be a number"} {:band "MONO" :album "Hymn to the Im
```

As you can see above, the `validate` function will return a vector with two elements. When the data passes validation the first element will be `nil`, and the second will be the original data. When the validation fails, the first element will be a map of errors associated with the keys that failed validation.

The `valid?` function will return a boolean value indicating whether the data is valid or not:

```
(st/valid? {:band "MONO" :album "Hymn to the Immortal Wind" :year 2009
;; => true
```

Build Tool: lein ▼

Topics

Books

Validation for nested data is specified using a vector path to the elements as follows:

```
(def schema
  {[:a :b] st/integer
   [:c :d] st/string})

(st/valid? {:a {:b "foo"} {:c {:d "bar"}}} schema)
;; => false
```

For further examples, please refer to the official project page.