

Internationalization

Contents

1. [Internationalization](#)

Internationalization

The [Tempura](#) library provides the functionality for internationalization and translation.

We'd need to add it to the `project.clj` dependencies:

```
[com.taoensso/tempura "1.0.0"]
```

We can now require Tempura in the namespace we wish to use for internationalization:

```
(ns <app>.i18n
  (:require [taoensso.tempura :as tempura :refer [tr]]))
```

We'll create a dictionary map such as the one below.

```
(def tconfig
  {:dict
   {:en-US
    {:missing ":en-GB missing text"
     :page {:title "Here is a title"
            :content "Time to start building your site."}}
    :fr-FR
    {:page {:title "Voici un titre"
            :content "Il est temps de commencer votre site."}}}}})
```

We can use the translation function directly as follows:

```
(tr tconfig [:en-US :fr-FR] [:page/title])
```

We can also add the Tempura middleware wrapper to our handler with the above config in our `middleware` namespace. We'll have to require

Build Tool:

Topics

- Your First Application
- REPL Driven Developme
- Application Profiles
- HTML Templating
- Static Assets
- ClojureScript
- Routing
- RESTful Services
- Request types
- Response types
- Websockets
- Middleware
- Sessions and Cookies
- Input Validation
- Security
- Component Lifecycle
- Database Access
- Database Migrations
- Logging
- **Internationalization**
 - Testing
 - Server Tuning
 - Environment Variables
 - Deployment
 - Useful Libraries
 - Sample Applications
 - Upgrading
 - Clojure Resources

Books

taoensso.tempura and add the new middleware wrapper to wrap-base as follows: the wrap-base function.

```
(ns <app>.middleware
  (:require [taoensso.tempura :as tempura :refer [tr]]
    ...))

(defn wrap-i18n [handler]
  (tempura/wrap-ring-request handler {:tr-opts tconfig}))

(defn wrap-base [handler]
  (-> handler
    wrap-i18n
    ...))
```

The middleware uses the Accept-Language HTTP header to infer the preferred locale for the client. The middleware will append two keys to the request. The first key is :tempura/accept-langs that contains a vector of keys parsed from the request, e.g: ["en-ES" "en-US"]. The second key is :tempura/tr and it contains the translations function initialized with the tconfig and the locales from the request. The locale middleware will use the first available locale contained in the :tempura/accept-langs key.

With the middleware setup, we can now use translations in our pages as seen below.

```
(ns mysite.routes.home
  (:use compojure.core)
  (:require [i18ntest.layout :as layout]
    [i18ntest.util :as util]))

(defn home-page [{tr :tempura/tr}]
  (layout/render
    "home.html" {:title (tr [:page/title])
      :content (tr [:page/content])}))

(defn about-page []
  (layout/render "about.html"))

(defroutes home-routes
  (GET "/" req (home-page req))
  (GET "/about" [] (about-page)))
```

See the official [Github](#) project page for more details on Tempura usage.

