Lisp in Summer Projects Submission

Submission Date	2013-10-01 13:52:26
Full Name	Ryan McGowan
Country	United States
Project Name	manners
Type of software	library
General category	library
LISP dialect	Clojure
GitHub URL	https://github.com/RyanMcG/manners
Did you start this project?	Yes, all the code is written by me
Project Description	I want to describe my project in this form.
Purpose	I felt there was a need for a general purpose validation library. manners is the result. Other libraries are restricted to validating maps only while manners is built around using simple predicates and can be applied to any data type.
Function	It is a validation library. See purpose.
Motivation	I worked on many projects over the summer but this is the one I am most pleased with. I think there is a need for it and I think it is a simplified take on validation that I have not seen in other projects.
Audience	The audience is Clojure developers who want to validate things besides maps.
Methodology	Manners is a validation library written in Clojure. It uses higher-order functions to do most of the work and relies on memoization so that you are not hurt by repeating yourself. To see full usage instructions see the README and a detailed description of manners API (and a bit of its internals). https://github.com/RyanMcG/manners

	The less detailed version: etiquette - A sequence of manners manner - A predicate and a message as the first and second arguments of a sequential collection [empty? "Must be empty."] coach - A higher order function which takes an etiquette and returns a function that applies the predicates from the etiquette to its argument returning a sequence of messages for the predicates that fail. The returned function is called an etiquette coach (see what I did there?). coaches can then be invoked with an argument to validate. Manners' most defining aspect is the simplicity of defining a manner. It is a tupple of a predicate and a message. The message is used when the the predicate is invoked on a value being validated to provide feedback to the developer or user. Using a simple predicate is advantageous. Clojure is a functional language after all so composing functions to define predicates is often a very easy thing to do.
Conclusion	This was a spin off of a project I would have liked to have submitted to lisp-in-summer but is unfinished. I think there is room for improvement of this library but it is not something I think needs to grow. This software works well, it is fully tested and it is damn useful.
Build Instructions	Use leiningen and add [manners "0.1.0"] to your dependencies.
Test Instructions	lein spec
Execution Instructions	It is a library so N/A. Detailed usage instructions and documentation can be found in the project README. https://github.com/RyanMcG/manners There is also API documentation. http://www.ryanmcg.com/manners/
Official	I have read rules and have abided by them. I am 18 years of age or older. I am not living in Brazil, Quebec, Saudi Arabia, Cuba, Iran, Myanmar (Burma), North Korea, Sudan, or Syria.