# FubuMVC Innovation Time Project

## Notes

* We can add our own conventions that coerce people into doing it the right way
  + E.g the one model in one model out for people who don’t know the project very wel
* We can easily add our own conventions for generating output (as I have done) and exceptions (as I did with the enum exception)
  + But we can copy the fubu guys and use behaviour chains to create our convention systems – with or without open rasta
* <http://blog.mikeobrien.net/2012/02/fubumvc-automatic-binding-of-request.html#comment-form>
* Low level access to the current http context just by using IoC
* You can have vertical strips of functionality by putting each verb of each action in its own handler
  + You can have input and view models specific just to that view which means changes are only specific to that action
* Source code is a lot easier to understand than Open Rasta
* Had to create my own model binder for taking incoming xml. Not sure if this would be a problem or using some OSS framework that does the whole lot including json parsing etc.
* Put request binder is a bit hacky, but was easy to implement conventional binding and access to the http request was easy (although you had to know what abstraction to get – but this allows to work on different environments – and the documentation and googling was superb)
* Everything is more granular than OR – using the compositional approach. There is always a place to hook into with a small piece of functionality, while the rest of the process reverts to normal behaviour.

## Requirements

1. RESTful
   1. Http status codes
   2. Codecs / content negotiation – conventional?
   3. Compare with OR for linking between resources
   4. Meets Raoul’s strict criteria - http://code.google.com/p/implementing-rest/wiki/LitmusTestForFrameworks
2. Handlers / Controllers
   1. What is one?
   2. Can we have action-per-controller
   3. How are URLs defined?
3. Model binding & Validation
   1. Compare with OR for granularity and ease of implementation
   2. Compare validation with OR and ASP.NET MVC
4. Diagnostics
   1. Which routes end up at which handler
   2. What about when something goes wrong?
      1. Compare with Open Rasta’s generic error page
5. Testing
   1. Is there a lot of Mocking of Http context going on?
   2. Is there an alternative to OpenRastas in-memory host for integration testing?
6. Compatibility
   1. Does it work on mono?
   2. Does it support http://owin.org/ or is it tied to IIS?
7. Community
   1. How many commits to the main repository
   2. How active is the google discussion group
   3. Associated projects – fubuvalidation, bottles etc

## Plan

I’ll mimic the catalogue api for now. This example will be for an api and not html. Show html blog post at the end to show how it is useful across the board

1. Xml happy path – Show a service homepage – with end to end test
   1. Define controllers
   2. Action per controller
   3. Url pattern

\*\* Mention the action per-controller allows things to be grouped vertically – things that change together live together\*\*

1. Content negotiation – with end-to-end test
   1. Request header determines otput type
      1. Json, xml,

\*\* Fubu are working on a richer content negotiation model \*\*

1. Add an entity: release
   1. Should 404 with error message when artist not found – E2E test
   2. Should bad request when incorrect type parameter supplied – E2E test
2. Allow releases to be searched for using type end-to-end test
   1. Add property-level model binder
   2. Compare with OR model binding
3. Boost up the hypermedia
   1. Add an artist resource
   2. Have links in artist to their releases
   3. Have links in releases back to the artist
   4. Compare with OR for linking between resources
4. Allow entities to be created using PUT
5. Allow entities to be deleted using DELETE
6. Allow releases to be added to an artist using POST
7. Add validation to the PUT and POST of releases
   1. Compare with OR
8. Show testing
   1. Unit testing handlers is easy like OR
   2. In-memory host for integration testing?
9. Try and host it away from IIS / ASP.NET
10. Try and get it working on mono
11. Show diagnostics and community stuff
    1. Mention how the behaviour chain configuration model allows such awesome diagnostics