ZULF'S MINIMAL MODEL OF METEOR RECEIVER JSON DOCUMENT APPS

ZULFIKAR MOINUDDIN AHMED

Service receivers are unique individuals who want a service from an app. We consider Meteor for the service and examine Blaze HTML templating. The problem is very simple. We want an arbitrary receiver to log in and be able to fill a questionnaire. We want a large centralised MongoDB database to sync the answers keyed on the receiver. That's it. We do not want to examine anything more complicated.

Let's take a step back from this problem and note some features of this problem. First, we want to fully leverage Meteor modules and not do any coding ourselves. Meteor produces a module for user/password management. We can leverage this and avoid having to track users.

Second, Meteor will manage front-end to backend database syncing of web forms. We do not want to do any work here either.

Our goal is not to show off any deep knowledge of internet and database coding. Instead our goal is to assume that meteor modules are highly debugged and avoid doing custom coding and hacking and exploit Meteor's capabilities well.

Our secondary goal will be to do the use cases here with clean minimal code so that the code is crisp and clear and not impossibly complex and robust.

Our tertiary goal is to demonstrate a low level aspect of all psychologically oriented per-receiver handling that serves as the abstraction of our choice, that positive psychology services will involve managing surveys and questionnaire data per receiver. Let us consider this a bit more clearly. Per receiver h (for human being), we consider the full state of h to be represented by m JSON documents $D_1(h), \ldots, D_m(h)$. Our goals above are to show that this abstraction can be faithfully implemented with a minimal canonical code in Meteor. Our example has a simple document that represents answers to two simple questions. We can genericize this for arbitrary questionnaire answers $D_1(h), \ldots, D_m(h)$ in a clear manner later.

Note what we are not doing. We are not interested in complex MongoDB code at all. We want the front-back database syncing design of Meteor do that for us. Second we might have several billion receivers using our system in production. We want to produce a simple example which will not choke if we ran the app on 1000 high end servers. We are using MongoDB thoroughput for dealing with that rather than babysitting database issues.

Now imagine the system in operation. All psychology applications can be thought of as JSON document management and processing keyed to individual receiver. It is the job of the psychologists to decide what questionnaires are appropriate and what responses are appropriate. We want here to show a basic prototype say of $D_o(h)$ a simple null document with an eye to production scalability to eight billion people. We want to produce discipline so that once a generic code is produced, we

Date: July 13, 2021.

do not have to continuously debug database and web delivery problems. That's the key to design here. We want to solve technology technicalities once and for all and allow the cluster management provide robustness and faultless simultaneous service to eight billion.

1. A Note on Positive Psychology Model of Human Beings

Positive Psychology is a natural scientific effort. Unlike physics the most difficult problem of psychology is careful choice of concepts and measurements. In physics some of the basic concepts and measurements are highly established. We have charged leptons, measurement of electric charge in Coulombs, for example. Psychology deals with macroscopic but extremely complex systems. Therefore the combinatorial explosion that would result with the wrong concepts and measurements is the most difficult problem. Services for positive psychology involve carefully crafted questionnaires and surveys. Scientific aspects of psychology involve statistical models of the relationships between variables. Therefore the positive psychology model of the human being is not made of large numbers of biological macromolecules; that would be suicidal as nothing would ever be discovered of any substance or depth as the variable numbers would be too high and measurements too difficult.

The essence of positive psychology applications is management of questionnaire result data per person and application of statistical relationship between variables.

Abstractly, we consider per person a set of JSON documents $D_1(h), \ldots, D_m(h)$, which contain implicitly measurements of variables $v_1(h), \ldots, v_{k(m)}(h)$, and then we use psychological knowledge which are some set of statistical models $S_1, \ldots, S_p$ involving $v_1, \ldots, v_k$. Then based on the predictions of the statistical models we want to have responses $R_1, \ldots, R_p$ to the receiver and these responses are services when the receiver benefits from psychological knowledge that will have impact on their lives, such as improvement of life satisfaction.

To the extent that actual people's life satisfaction improves based on the responses, the service has value to the person, and that is the basis for why people might want to *pay* for the services.

2. Psychology is Very Hard Technology is Trivial

From our point of view, MongoDB, Meteor, Mesos running on clusters are *solved problems*. Syncing of front-back between database and web front end using Meteor's Reactive Variables is a solved problem. Once we have Mesos running on thousand node clusters with Nginx servers and Meteor apps connected to MongoDB cloud server, efficient delivery to billions will be a *solved problem*. That the solution of these problems took many decades of effort is immaterial. For us they are all like telephones. We do not intend to compete with these solved problems because we have nothing much to add. They are solved, so we are interested only in using the technology to solve problems that are unsolved.

Psychology of Life Satisfaction is not a solved problem on billions of users scale. I could go into the issues, but the essence is that without billions of people measurement we do not know what variables are important in a global scale to significantly improve life satisfaction with any certainty yet. This is a very hard problem and this is the problem that interests us.

We are not here to show off our knowledge of wiring of databases and internet communication. Those things will not give us any advancement in our project. They will also not solve our major problems which is life satisfaction improvement for the human race. They will merely waste a lot of our time.

3. Example Template For Meteor

Here is working code. It's a simple form with some sample questions. It is working code, so we want to highlight the simplicity of Blaze Templating to produce something reasonable on the web page.

```
<head>
  <title>test-opt</title>
</head>

<body>
  <h1>Testing a questionnaire</h1>

  {{> optq}}
</body>


<template name="optq">
<form>
{{> q t="You and your spouse make up after a fight" a1="I forgive him/her" a2="I am usually f
{{> q t="You forget your spouse's birthday" a1="I am not good at remembering birthdays" a2="I

</form>
</template>

<template name="q">

<p>{{t}}</p>

<div>
  <input type="radio" name={{t}} id="a" value={{a1}} checked>
    <label for="a">{{a1}}</label>
</div>

<div>
  <input type="radio" name={{t}} id="b" value={{a2}}>
    <label for="b">{{a2}}</label>
</div>

</template>
```

4. Comments About The Code

This is simple questionnaire with Blaze Templating which is readable, uses the parametrized template facility, and is fast to load. We want to use the user/pass

model with login and a form connected to the MongoDB backend using Meteor facilities.

Scalability is automatic the moment we deploy the app on a thousand node cluster all running the app, all the way to several billion.

Syncing to database is managed the moment we hook up the form using Reactive Variables features of Meteor.

This means that this minimal code can fill up the database without choking and can serve billions simultaneously without much complex Meteor code.

This then allows us to package the whole 1000-node cluster, large Meteor cloud DB, etc. with simple elementary parts and keep focus on the actual Psychology difficulties. The technology stack is robust for use immediately.

5. Psychology Is A New Frontier

No one has successfully served billions with psychology apps, so this is a new frontier. We will have new psychology discoveries along the way. What we will not face are constant technology disruptions and problems because our design is world class.

6. I Intend to Hire Ph.D.s from Berkeley Who Worked On Mesos Code

Mesos is a very sophisticated system that is not tested on thousands of server nodes. I will not take a chance and hire Director level folks who know Mesos and can ensure Research to ensure Mesos is tested on thousands of harware nodes. That could be a problem but Microsoft and Bill Gates cannot compete with large clusters running Mesos and Nginx and Ubuntu at all. Even optimisations are superior in these systems than what Microsoft has done. Microsoft tends to produce mixture of arbitrary low level modifications and this makes their system totally impossible to maintain without large staff of coders just tracking the development. I will avoid this because Mesos is a beautiful system and will work with some high quality Directors of Research from Berkeley. A good team getting paid top salaries can ensure years of robust service that is world class. Microsoft and Bill Gates cannot compete with this arrangement in a thousand years.