MBER REST Service

Model Based Extensibility
at Runtime

Reggie Wilbanks
reggie@wilbanks.info

Thought Experiment

1. I want to create a thing.
2. I want to give a thing a relationship (which is a thing) to another thing.

Just with those two concepts I should be able to do some really cool stuff.

So, what's an easy way to create a thing, like a "project", RESTfully?

```
POST /mber/projects HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 15
Content-Type: application/x-www-form-urlencoded

project=modelmaker
```

Let's dissect the important parts of that post call above:

```
POST /mber/projects HTTP/1.1
```

The first line is a POST, meaning in RESTful terms that we're creating something. In my service, the "/mber/" part tells me that this is an MBER instruction (to be handled by the MBER service). Notice that "projects" is plural. This means we're working within a group (or namespace) of items. In my service, if a namespace doesn't already exist, it would be created on the fly the first time something is POSTed to it.

```
project=modelmaker
```

This key-value pair takes the form of the singular form of the namespace "project" and name of the new resource to be created "modelmaker".

But, let's say I wanted to capitalize the name modelmaker instead. I'd need some way to modify the name RESTfully:

```
PUT /mber/projects/modelmaker HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 15
Content-Type: application/x-www-form-urlencoded

project=ModelMaker
```

We do this by PUTting to the target resource (modelmaker), passing the new project name:

project=ModelMaker

So, that lets me create (and modify) a thing, but not a very useful thing. Although, given the constructs I have so far I can start to do some interesting stuff. Like add developers to my project:

```
POST /mber/projects/ModelMaker/developers HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 15
Content-Type: application/x-www-form-urlencoded

developer=Bob
```

and, add some properties to my developer:

```
POST /mber/projects/ModelMaker/developers/Bob HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 15
Content-Type: application/x-www-form-urlencoded

speed=fast
```

Hey, wouldn't it be cool if anytime a namespace was created (like developers, under ModelMaker) that it would become accessible from the top level, like:

```
GET /mber/developers HTTP/1.1
Accept: application/x-www-form-urlencoded
```

which so for, would return "Bob". Whoa! But wait a minute, now I've already built a relationship and could imagine this GET:

```
GET /mber/developers/Bob/projects HTTP/1.1
Accept: application/x-www-form-urlencoded
```

which would return "ModelMaker".

Now, let's say I want to add some more properties to Bob:

```
POST /mber/developers/Bob HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 39
Content-Type: application/x-www-form-urlencoded

documentationStyle=verbose&team=CoreModelMaker
```

Meaning, under default circumstances (because we're posting to Bob as a developer in the root namespace), Bobs documentation style is verbose. But let's say in the ModelMaker project, he's brief with his documentation:

```
PUT /mber/projects/ModelMaker/developers/Bob HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 24
Content-Type: application/x-www-form-urlencoded

documentationStyle=brief
```

There's that PUT again, to modify Bobs documentation style in the context of ModelMaker. Now, let's add some more developers:

```
POST /mber/developers HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 29
Content-Type: application/x-www-form-urlencoded

developer=Amy&developer=Brian
```

And give them some teams:

```
POST /mber/developers/Amy HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 39
Content-Type: application/x-www-form-urlencoded

team=Services
```

and maybe even a project at the same time:

```
POST /mber/developers/Brian HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 39
Content-Type: application/x-www-form-urlencoded

team=CoreModelMaker&project=ModelMaker
```

Hey, look! I should now be able to do something like:

```
GET /mber/teams HTTP/1.1
Accept: application/x-www-form-urlencoded
```

which would return "CoreModelMaker" and "Services". And now, maybe I can start to get a bit tricky:

```
GET /mber/projects/ModelMaker/developers HTTP/1.1
Accept: application/x-www-form-urlencoded
```

which would return "Bob" and "Brian", even though Brian was added to the ModelMaker project by adding a "project" property.

And how about this:

```
GET /mber/projects/ModelMaker/teams HTTP/1.1
Accept: application/x-www-form-urlencoded
```

which would return "CoreModelMaker", but not "Services" yet, until something like:

```
POST /mber/developers/Amy HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 39
Content-Type: application/x-www-form-urlencoded

project=ModelMaker
```

Or,

```
POST /mber/projects/ModelMaker HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 39
Content-Type: application/x-www-form-urlencoded

team=Services
```

Or,

```
POST /mber/teams/Services HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 39
Content-Type: application/x-www-form-urlencoded

project=ModelMaker
```

Well, that was fun. But what does it all mean? For my sample use case, I'd like to set up all the projects I'm involved with. Add the teams involved, and the developers in those teams. Each of those developers have skills, which may change depending on which project they're working on. Once I've added all this information, I may want to look at from different views. Maybe I want to see all the projects my team is working on, or maybe all the teams working on a project.

A person.

```
POST /mber/person HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 0
Content-Type: application/x-www-form-urlencoded
```

That creates a collection of "person" things called "persons",

```
GET /mber/persons HTTP/1.1
Accept: application/x-www-form-urlencoded
```

But the word "persons" is goofy, so how about:

```
POST /mber/persons HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 0
Content-Type: application/x-www-form-urlencoded

aka=people
```

There! That's better.

```
GET /mber/people HTTP/1.1
Accept: application/x-www-form-urlencoded
```

Bob is a person.

```
POST /mber/people HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 0
Content-Type: application/x-www-form-urlencoded

person=Bob
```

Bob is also a developer. And moreover, all developers are people.

```
POST /mber/people/developers HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 0
Content-Type: application/x-www-form-urlencoded
```

But, that doesn't mean all people are developers. Okay, so doing this:

```
GET /mber/people HTTP/1.1
Accept: application/x-www-form-urlencoded
```

Would now give me "Bob", "Amy", and "Brian".

Lets say Bob doesn't work on Wednesdays. How could one represent that? Well, first Wednesday is something new, it's a day:

```
POST /mber/days/Wednesday HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 0
Content-Type: application/x-www-form-urlencoded
```

And we might as well throw in the other days as well:

```
POST /mber/days HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 0
Content-Type: application/x-www-form-urlencoded

Monday,Tuesday,Thursday,Friday,Saturday,Sunday
```

See, what I did there? Why type more than I have to? Let's create a weekend:

```
POST /mber/weekend HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 0
Content-Type: application/x-www-form-urlencoded

Saturday,Sunday
```

So,

```
GET /mber/days HTTP/1.1
Accept: application/x-www-form-urlencoded
```

Would return Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. So, back to the problem at hand; Bob doesn't work on Wednesdays.

```
POST /mber/people/Bob HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 0
Content-Type: application/x-www-form-urlencoded

nonWorkDay=Wednesday
```

Which creates a nonWorkDays collection, with Wednesday in it. Let's also create a work day collection:

```
POST /mber/WorkDays HTTP/1.1
Host: reggie.wilbanks.info
User-Agent: Mozilla/4.0
Content-Length: 0
Content-Type: application/x-www-form-urlencoded

includes=days&excludes=Weekend
```