# Rest Resources in some more depth

Databases Three

Otago Polytechnic
Dunedin, New Zealand

## We know so far

- We can create a $resource object with a RESTful URL scheme.
- This lets us access a RESTful resource that presents a typical REST API.
- We can add custom methods to access other parts of our API.
- We need be be attentive to *Cross-origin resource sharing* (CORS) issues.

## So, we can do this:

```
app.factory('User', function($resource) {
   return $resource('http://someurl/users/:id.json')});

... later ...

 user = User.get({id: 1});
```

## Creating a new user

```
  user = new User({name:'John Doe'});
  user.$save();
```

OR

```
 User.save({},user);
```

## Problem 1: CORS issues

- Once you start doing something like a POST, you may run into CORS problems.
- If you see your server getting an OPTIONS request, that's a CORS issue.
- One method: `http://blog.rudylee.com/2013/10/29/rails-4-cors/`

## Another CORS approach: in JS

```
app.config(['$httpProvider', function($httpProvider) {
  $httpProvider.defaults.headers.common = {};
  $httpProvider.defaults.headers.post = {};
  $httpProvider.defaults.headers.put = {};
}]);
```

## Problem 2: POST/PUT payloads

- Our JSON data should be going into the payloads of our POST or PUT requests.
- However, I've run into problems where this data isn't showing up.

## Custom methods

We also want to be able to add custom methods for our resources.

```
app.factory('User', function($resource) {
   return $resource('http://someurl/users/:id.json', {},
     {update: {method:'PUT', url:'http://someurl/users/:id.json'}}
   )});

 Then we can call

 User.update({id: 1}, {user: data});
```