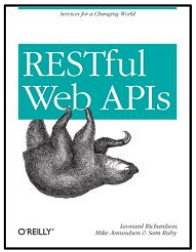
Building

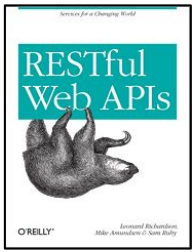
Mike Amundsen
@mamund

Designing and Building Great APIs



Building Great APIs

- The Build Process
- Data, Object, Resource, Representation
- API Server Example
- API Client Example



build

/bild/

verb

1. construct (something, typically something large) by putting parts or material together over a period of time.
   "the factory was built in 1936"
   synonyms: construct, erect, put up, assemble; More

Building APIs

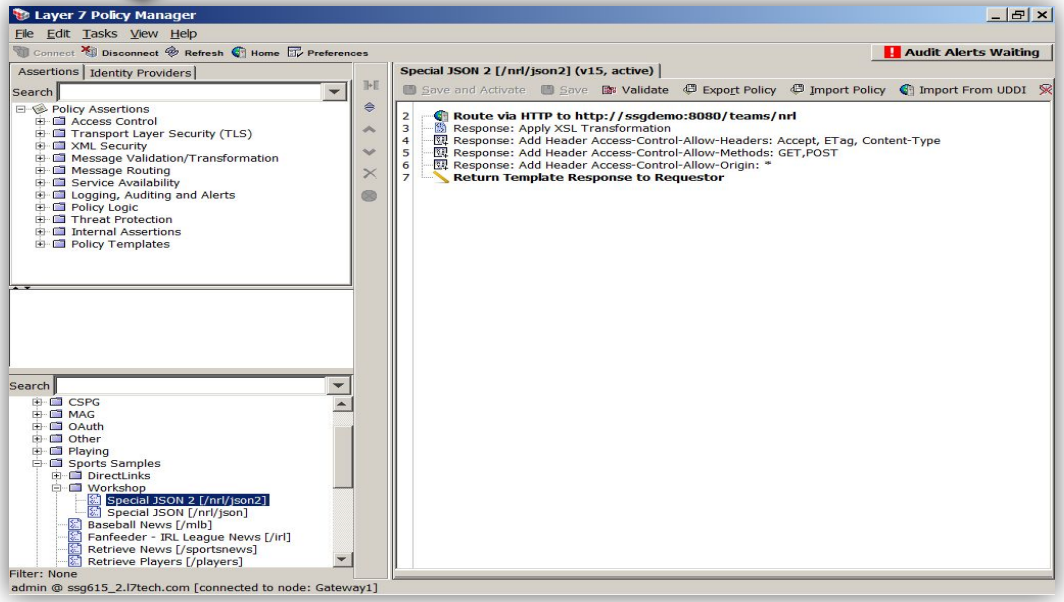
- API builds are the real thing
- Production-ready, access-controlled, resilient, scalable.
- Building the production implementation means
  - Working out all the kinks
  - Supporting all the use-cases identified during the sketch and prototype phases.

Building APIs

```
var express = require('express'),
app = express();
var port = 8080;
app.listen(port);

app.get("/tasks", function(req, res) {
    res.status(200).send(‘<response>
        <tasks>
            <task>
            <name>Pick up Kai</name>
            <priority>1</priority>
        </tasks>
    </response>’);
}
```

Building APIs

```
GET
/tasks
```



Building APIs

- Each implementation has their own challenges to overcome.
- Each deserves their own guidance and style-guides.
  - Gateway Policies
  - ESB Rules
  - Scripting (NodeJS)
  - Code (Java/C#)
- All require exhaustive testing at the unit, acceptance, and integration levels.
- All require detailed access control.

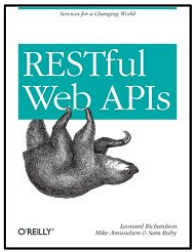
Production APIs are made last.

D.O.R.R

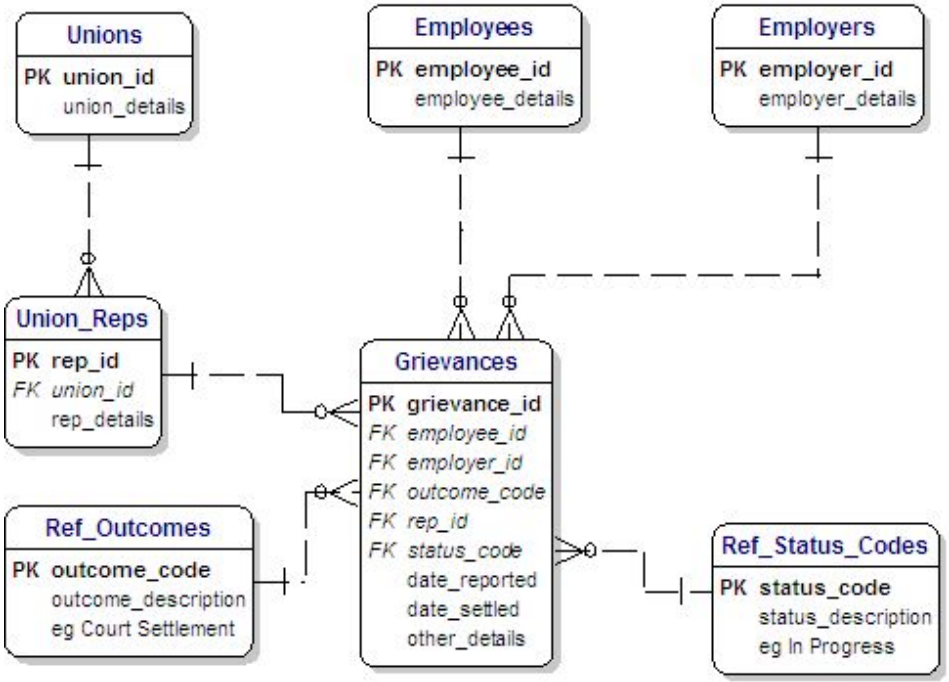
#mcaTravels

@mamund

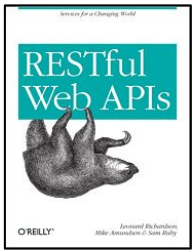
#perth2018



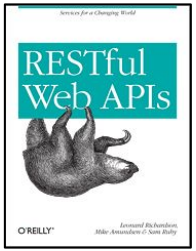
Data Model

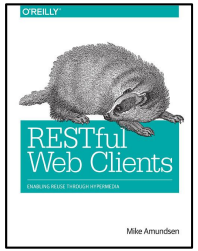


#mcaTravels **@mamund** #perth2018

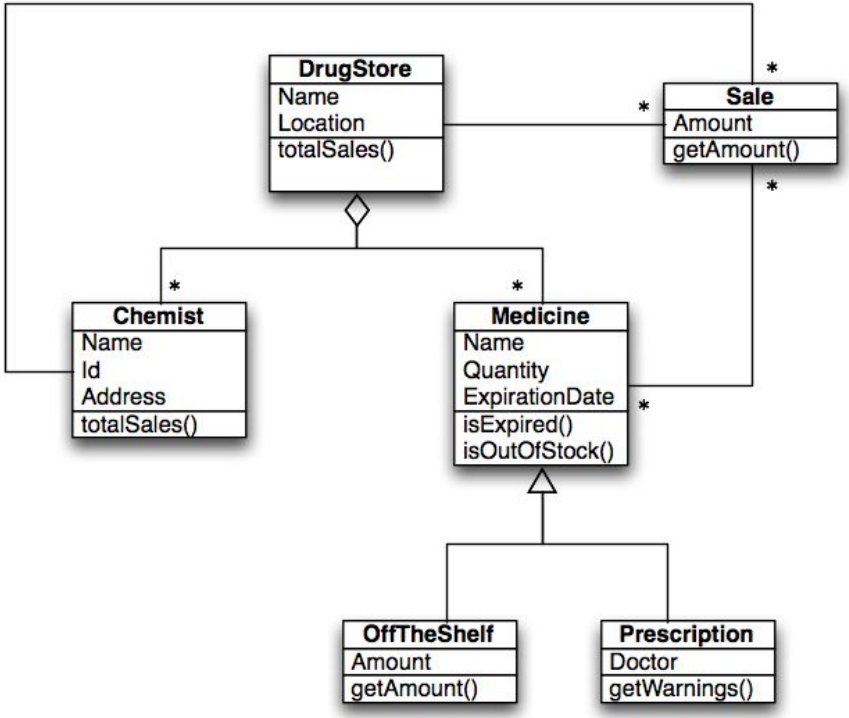


```
function main(object, action, arg1, arg2, arg3) {
  var rtn;

  switch(action) {
    case 'list':
      rtn = getList(object);
      break;
    case 'filter':
      rtn = getList(object, arg1);
      break;
    case 'item':
      rtn = getItem(object, arg1);
      break;
    case 'add':
      rtn = addItem(object, arg1, arg2);
      break;
    case 'update':
      rtn = updateItem(object, arg1, arg2, arg3);
      break;
    case 'remove':
      rtn = removeItem(object, arg1);
      break;
    default:
      rtn = null;
      break;
  }
  return rtn;
```

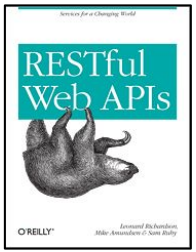




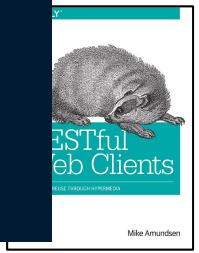
Object Model



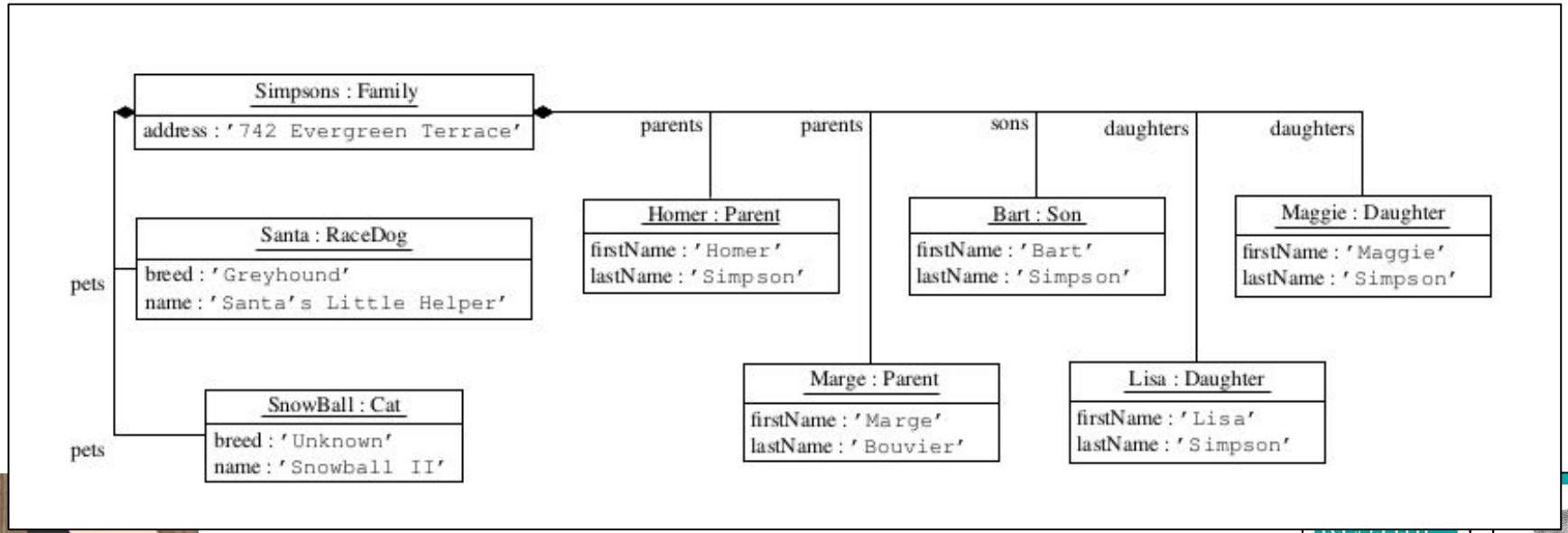
#mcaTravels **@mamund** #perth2018

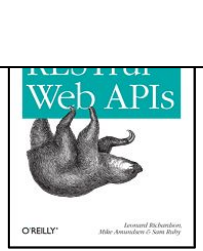


```
70 // update an existing task object
71 function updateTask(elm, id, task, props) {
72   var rtn, check, item;
73
74   check = data(elm, 'item', id);
75   if(check===null) {
76     rtn = utils.exception("File Not Found", "No record on file", 404);
77   }
78   else {
79     item = check;
80     item.id = id;
81     item.title = (task.title===undefined?check.title:task.title);
82     item.completed = (task.completed===undefined?check.completed:task.completed);
83
84     if(item.completed!=="false" && item.completed!=="true") {
85       item.completed="false";
86     }
87     if (item.title === "") {
88       rtn = utils.exception("Missing Title");
89     }
90     else {
91       data(elm, 'update', id, setProps(item, props));
92     }
93   }
```

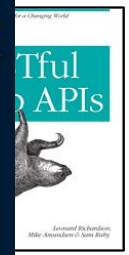


Resource Model

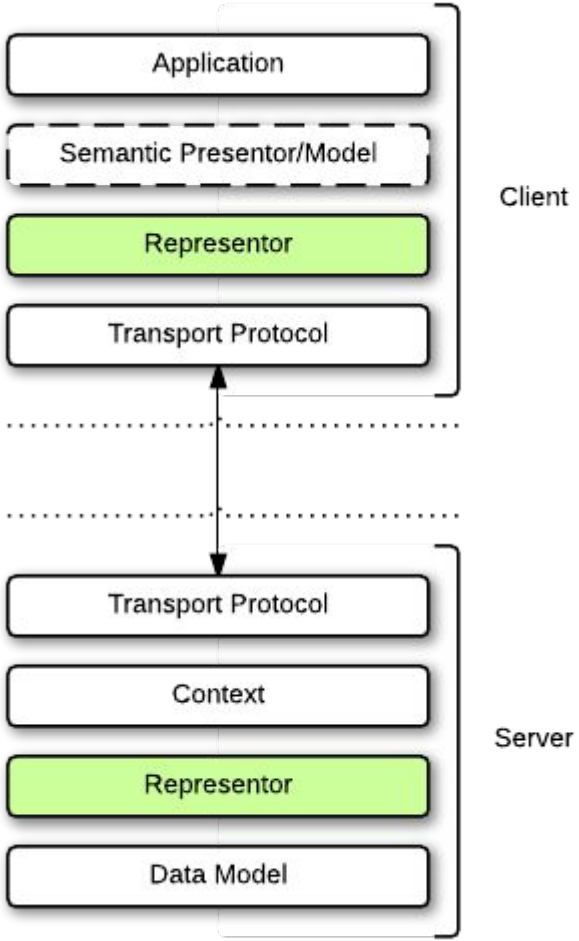


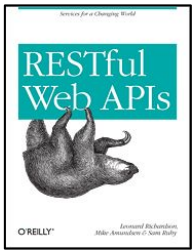


```
switch (req.method) {
case 'GET':
  switch(sw[0]) {
    case '?':
      sendList(req, res, respond, utils.getQArgs(req));
      break;
    case "*":
      sendList(req, res, respond);
      break;
    default:
      sendItem(req, res, sw, respond);
      break;
  }
  break;
case 'POST':
  if(sw[0]==="*") {
    addItem(req, res, respond);
  }
  else {
    respond(req, res, utils.errorResponse(req, res, 'Method
  }
  break;
case 'PUT':
  if(sw[0]!=="*") {
    updateItem(req, res, respond, parts[0]);
```

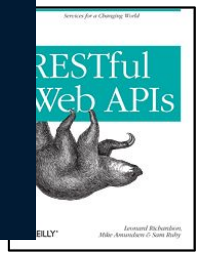


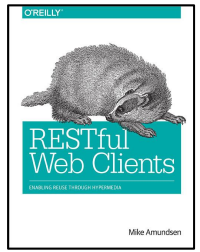
Representor Model





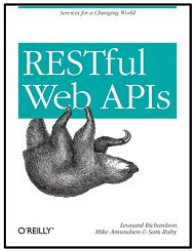
```
27   // dispatch to requested representor
28   switch(mimeType.toLowerCase()) {
29     case "application/json":
30       doc = json(object, root);
31       break;
32     case "application/vnd.collection+json":
33       doc = cj(object, root);
34       break;
35     case "application/vnd.amundsen.uber+json":
36       doc = uberjson(object, root);
37       break;
38     case "text/html":
39     case "application/html":
40     default:
41       doc = html(object, root);
42       break;
43   }
44
45   return doc;
```





Putting it all Together

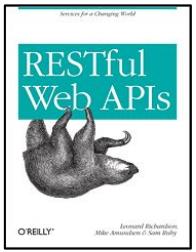
```
exports.task = function(action, args1, args2, args3) {
  var object, rtn;

  object = 'task';
  rtn = null;

  switch(action) {
    case 'list':
      rtn = loadList(storage(object, 'list'), object);
      break;
    case 'read':
      rtn = loadList(storage(object, 'item', args1), object);
      break;
    case 'filter':
      rtn = loadList(storage(object, 'filter', args1), object);
      break;
    case 'add':
      rtn = loadList(storage(object, 'add', args1), object);
      break;
    case 'update':
      rtn = loadList(storage(object, 'update', args1, args2, args3), object);
    default:
      rtn = null;
  }

  return rtn;
}
```



Putting it all Together

```
exports.task = function(action, args1, args2, args3) {
  var object, rtn;

  object = 'task';
  rtn = null;

  switch(action) {
    case 'list':
      rtn = loadList(storage(object
      break;
    case 'read':
      rtn = loadList(storage(object
      break;
    case 'filter':
      rtn = loadList(storage(object
      break;
    case 'add':
      rtn = loadList(storage(object
      break;
    case 'update':
      rtn = loadList(storage(object
    default:
      rtn = null;
  }

  return rtn;
}
```

```
switch(req.method) {
  case 'GET':
    if(parts[1] && parts[1].indexOf('?')===-1) {
      switch (parts[1]) {
        case "complete":
          sendCompleteForm(req, res, parts[2], respond);
          break;
        case "assign":
          sendAssignUserForm(req, res, parts[2], respond);
          break;
        case "add":
          sendAddTaskForm(req, res, respond);
          break;
        case "all":
        case "bycategory":
        case "bytitle":
        case "bycomplete":
          sendList(req, res, respond, parts[1]);
          break;
        default:
          sendItem(req, res, parts[1], respond);
          break;
      }
    }
    else {
      sendList(req, res, respond);
    }
    break;
```

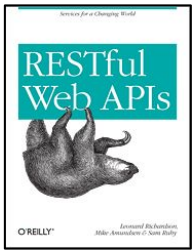


Putting it all Together

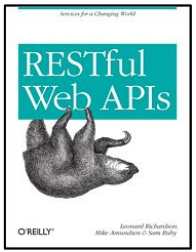
```
exports.task = function(action, args1, args2, args3) {
  var object, rtn;

  object = 'task';
  rtn = null;

  switch(action) {
    case 'list':
      rtn = loadList(storage(object
      break;
    case 'read':
      rtn = loadList(storage(object
      break;
    case 'filter':
      rtn = loadList(storage(object
      break;
    case 'add':
      rtn = loadList(storage(object
      break;
    case 'update':
      rtn = loadList(storage(object
    default:
      rtn = null;
  }

  return rtn;
}
```

```
switch(req.method) {
  case 'GET':
    if(parts[1] && pa
      switch (parts[1
        case "complet
          sendComplet
          break;
        case "assign"
          sendAssignU
          break;
        case "add":
          sendAddTaskForm(req, res, respond);
          break;
        case "all":
        case "bycategory":
        case "bytitle":
        case "bycomplete":
          sendList(req, res, respond, parts[1]);
          break;
        default:
          sendItem(req, res, parts[1], respond);
          break;
      }
    }
    else {
      sendList(req, res, respond);
    }
    break;
```

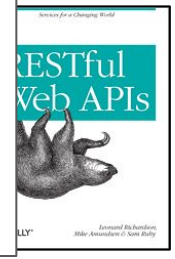
```
// dispatch to requested representor
switch(mimeType.toLowerCase()) {
  case "application/json":
    doc = json(object, root);
    break;
  case "application/vnd.collection+json":
    doc = cj(object, root);
    break;
  case "application/vnd.amundsen.uber+json":
    doc = uberjson(object, root);
    break;
  case "text/html":
  case "application/html":
  default:
    doc = html(object, root);
    break;
}

return doc;
```

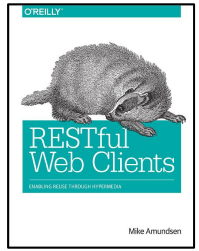


Server Example

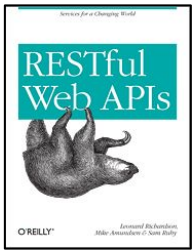


```
{
  - collection: {
        version: "1.0",
        href: "//rwcbook12.herokuapp.com/task/",
        title: "TPS - Tasks",
        content: "<div class="ui segment"><h3>Manage your TPS Tasks here.</h3><p>You can do t
        "complete", assign tasks to a user</li><li>Filter the list by Title, Assigned User, a
      - links: [
          - {
                href: "http://rwcbook12.herokuapp.com/home/",
                rel: "home collection",
                prompt: "Home"
            },
          - {
                href: "http://rwcbook12.herokuapp.com/task/",
                rel: "self task collection",
                prompt: "Tasks"
            },
          - {
                href: "http://rwcbook12.herokuapp.com/user/",
                rel: "user collection",
                prompt: "Users"
            }
        ],
      - items: [
          - {
                rel: "item",
                href: "//rwcbook12.herokuapp.com/task/1l9fz7bhaho",
              - data: [
                  - {
                        name: "id",
                        value: "1l9fz7bhaho",
```

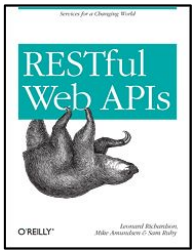





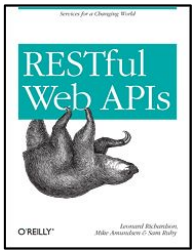
```
- queries: [
  - {
        rel: "usersByNick search",
        href: "http://rwcbook12.herokuapp.com/user/",
        prompt: "Search By Nick",
      - data: [
          - {
              name: "nick",
              value: "",
              prompt: "Nickname",
              required: false,
              readOnly: false,
              patttern: ""
            }
        ]
    },
  - {
        rel: "usersByName search",
        href: "http://rwcbook12.herokuapp.com/user/",
        prompt: "Search By Name",
      - data: [
          - {
              name: "name",
```



```
- template: {
      prompt: "Add User",
      rel: "create-form userAdd create-form",
    - data: [
        - {
            name: "nick",
            value: "",
            prompt: "Nickname",
            required: true,
            readOnly: false,
            patttern: "[a-zA-Z0-9]+"
          },
        - {
            name: "email",
            value: "",
            prompt: "Email",
            required: false,
            readOnly: false,
            patttern: ""
          },
        - {
            name: "name",
            value: "",
            prompt: "Full Name",
            required: true,
            readOnly: false,
            patttern: ""
```



API Client Example



Home | Tasks | Users

TPS - Tasks

Manage your TPS Tasks here.

You can do the following:

- Add, Edit and Delete tasks
- Mark tasks "complete", assign tasks to a user
- Filter the list by Title, Assigned User, and Completed Status

item | Edit | Delete — Assign User | Mark Completed | Mark Active

ID	1l9fz7bhaho
Title	extensions
Tags	forms testing
Complete Flag	true
Asigned User	carol

item | Edit | Delete — Assign User | Mark Completed | Mark Active

ID	1m80s2qgsv5
Title	Run client-side tests
Tags	test
Complete Flag	true
Asigned User	alice

Add Task

Title

Tags

Complete false

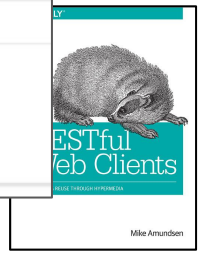
Submit

Queries

Completed Tasks

Complete true

Submit



Home | Tasks | Users

TPS - Users

Manage your TPS Users here.

You can do the following:

- Add and Edit users
- Change the password, view the tasks assigned to a user
- Filter the list by Nickname or FullName

item — Edit User | Change Password | Assigned Tasks

Nickname	alice
Email	alice-ted@example.org
Password	a1!c#
Full Name	Alice Teddington, Jr.

item — Edit User | Change Password | Assigned Tasks

Nickname	bob
Password	b0b
Full Name	Bob Carrolton

item — Edit User | Change Password | Assigned Tasks

Nickname	carol
Password	c@r0l

Add User

Nickname

Email

Full Name

Password

Submit

Queries

Search By Nick

Nickname

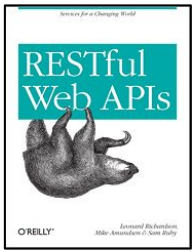
Submit

So...

#mcaTravels

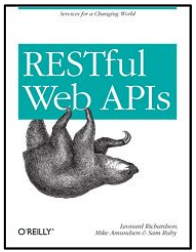
@mamund

#perth2018

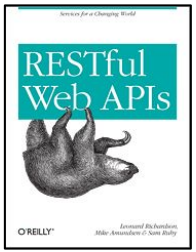


Building Great APIs

- The Build Process (built to last)
- Data, Object, Resource, Representation



Exercise


RESTful
Web APIs
O'REILLY


O'REILLY
RESTful
Web Clients
Mike Amundsen

Building

Mike Amundsen
@mamund

Designing and Building Great APIs