FALCOR

One Model Everywhere

Jafar Husain

@jhusain

NETFLIX


Every user wants to believe that the
entire cloud is on their device.

What if you could code that way?

This is the story of how

eliminated 90% of the networking code in our app.

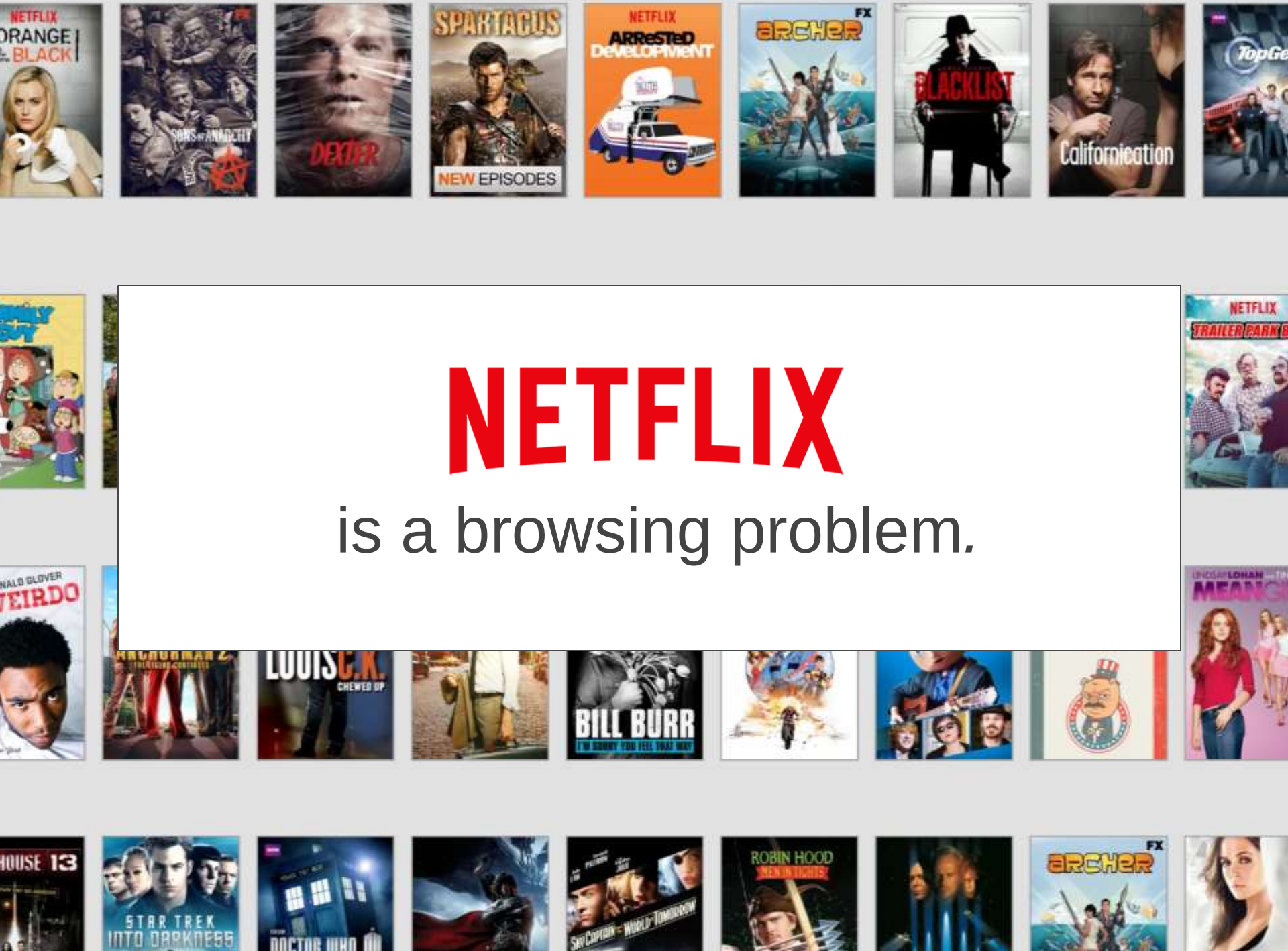
Jafar Husain

- Netflix’s Cross Team UI Tech Lead
- Architect of FALCOR
- Active Participant in TC-39, the JavaScript standards committee

NETFLIX

NETFLIX

Netflix had a RESTful API.


NETFLIX
ORANGE
BLACK
SONS OF ANARCHY
DEXTER
SPARTACUS
NEW EPISODES
NETFLIX
ARRESTED DEVELOPMENT
FX
ARCHER
BLACKLIST
Californication
NETFLIX
TRAILER PARK
NETFLIX
is a browsing problem.
DONALD GLOVER
WEIRDO
LOUIS C.K.
CHEWED UP
BILL BURR
HOUSE 13
STAR TREK
INTO DARKNESS
DOCTOR WHO
ROBIN HOOD
MEN IN TIGHTS
FX
ARCHER

Netflix's RESTful API

http://www.netflix.com/genreList/12429

http://www.netflix.com/title/1601923

RPC API

http://www.netflix.com/genreLists
?rowOffset=0&rowSize=5&colOffset=5
&colSize=15&titleprops=name,boxshot
http://www.netflix.com/setRating
?titleId=5&view=movieDetailPage

NETFLIX
2011

Why move away from REST?

Once the web was a place to get things.

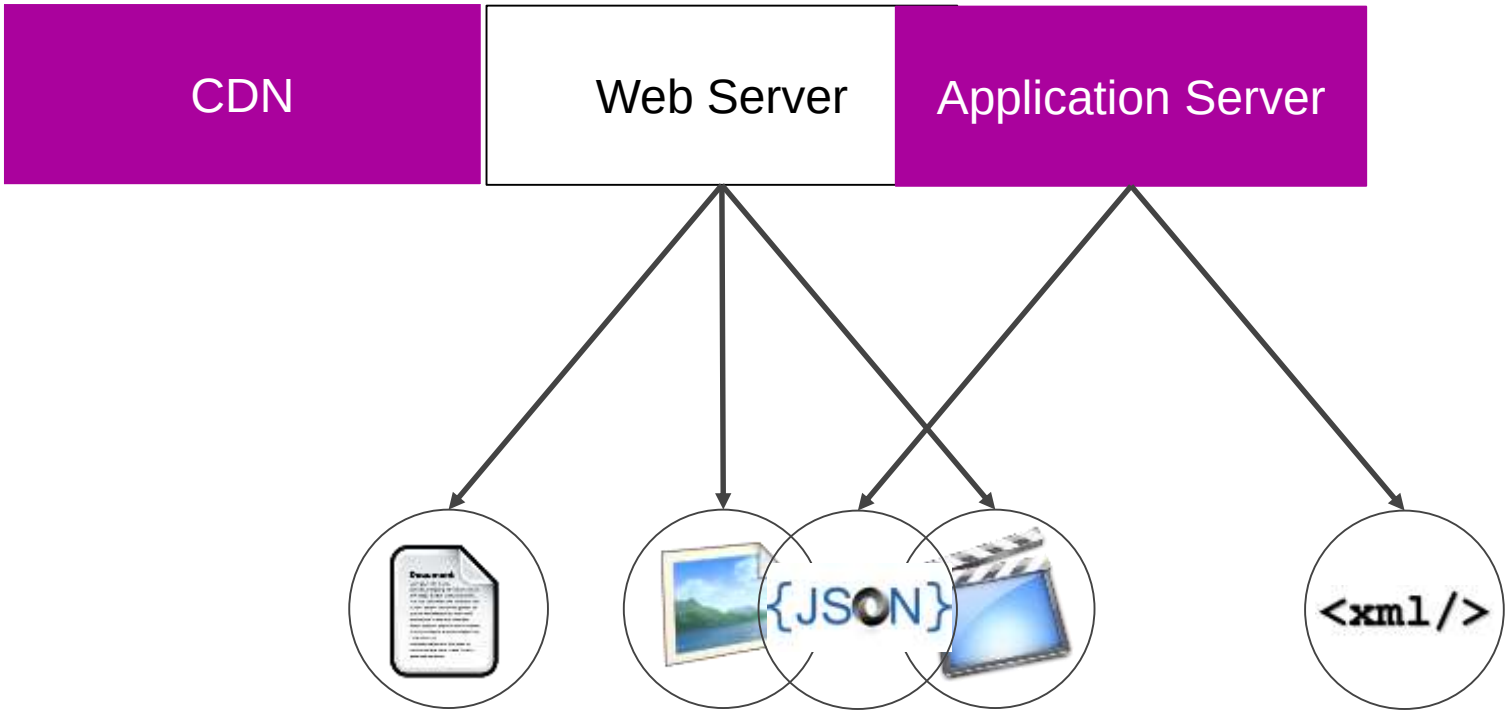
YAHOO!

AMERICA Online.

Today the web is a place to do things.

Web Servers serve large resources.

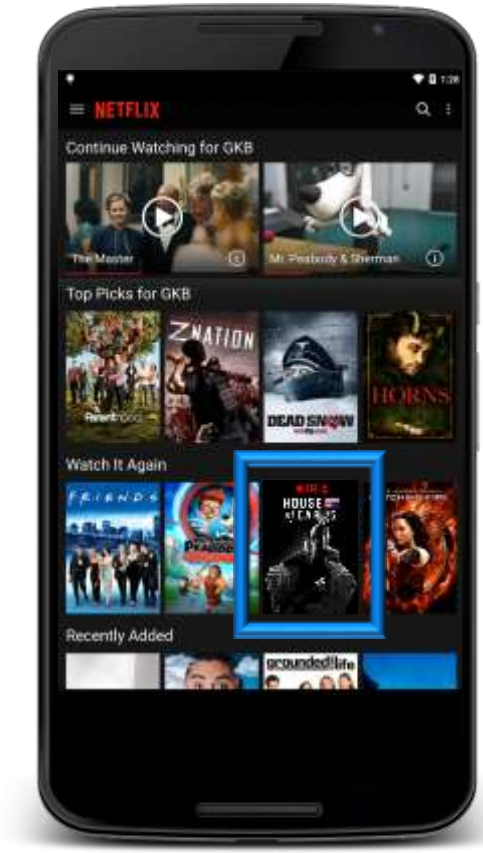
App Servers serve small resources.



Fine-Grained Title Resource

http://netflix.com/titles/95632123

```
{
   id: 2523,
   name: “House of Cards”,
   boxshot: “http://…/018-192-x50.png”,
   rating: 5,
   bookmark: 52119562,
   director: “David Fincher”,
   // a few more fields
}
```



REST: Multiple Roundtrips

01: GET /genreLists

02: GET netflix.com/titles/956325
03: GET cdn01.netflix.com/072-192-x50.png

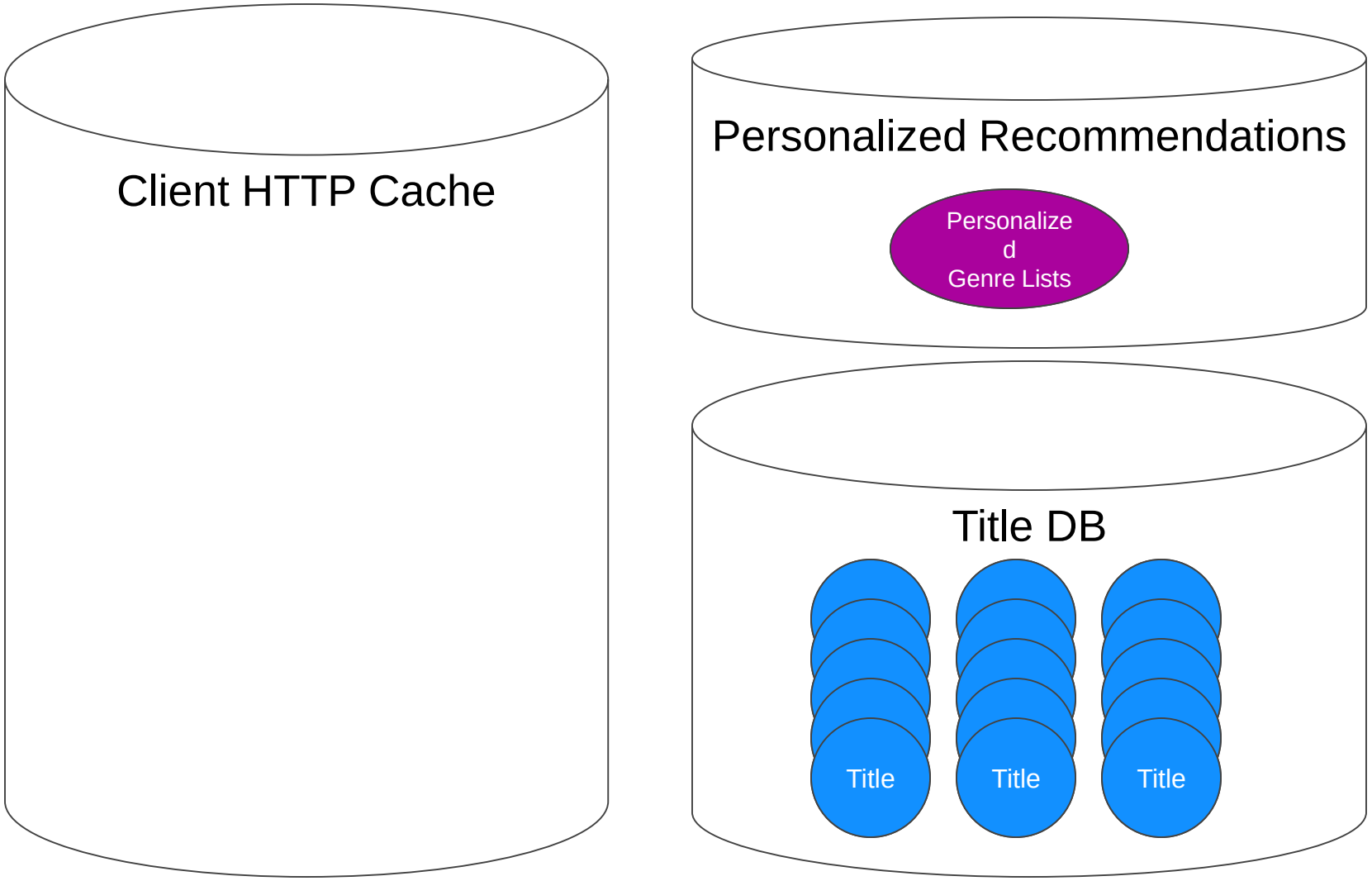
04: GET netflix.com/titles/992338
05: GET cdn03.netflix.com/018-192-x50.png

06: GET netflix.com/titles/912738
07: GET cdn09.netflix.com/651-70x50.png

. . .

27: GET netflix.com/titles/561826
28: GET cdn23.netflix.com/018-70x50.png

REST: Cache Coherence



REST

- Cache Consistency
- Loose Coupling

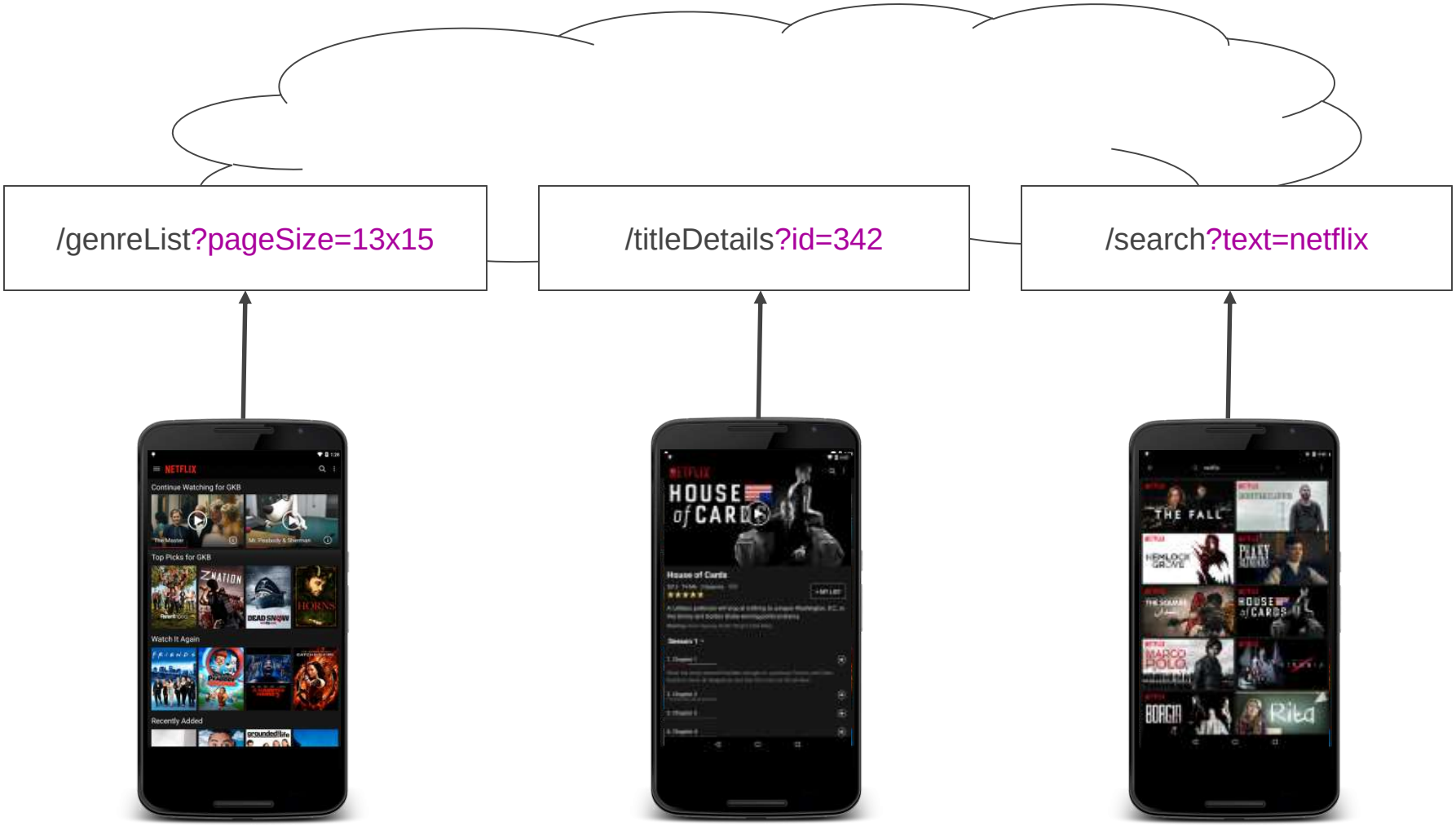
- High Latency
- Large Message Sizes

RPC API

http://www.netflix.com/genreLists
?pageSize=10x15&titleprops=id,boxshot
http://www.netflix.com/setRating
?titleId=5&view=movieDetailPage

NETFLIX
2011

Netflix RPC

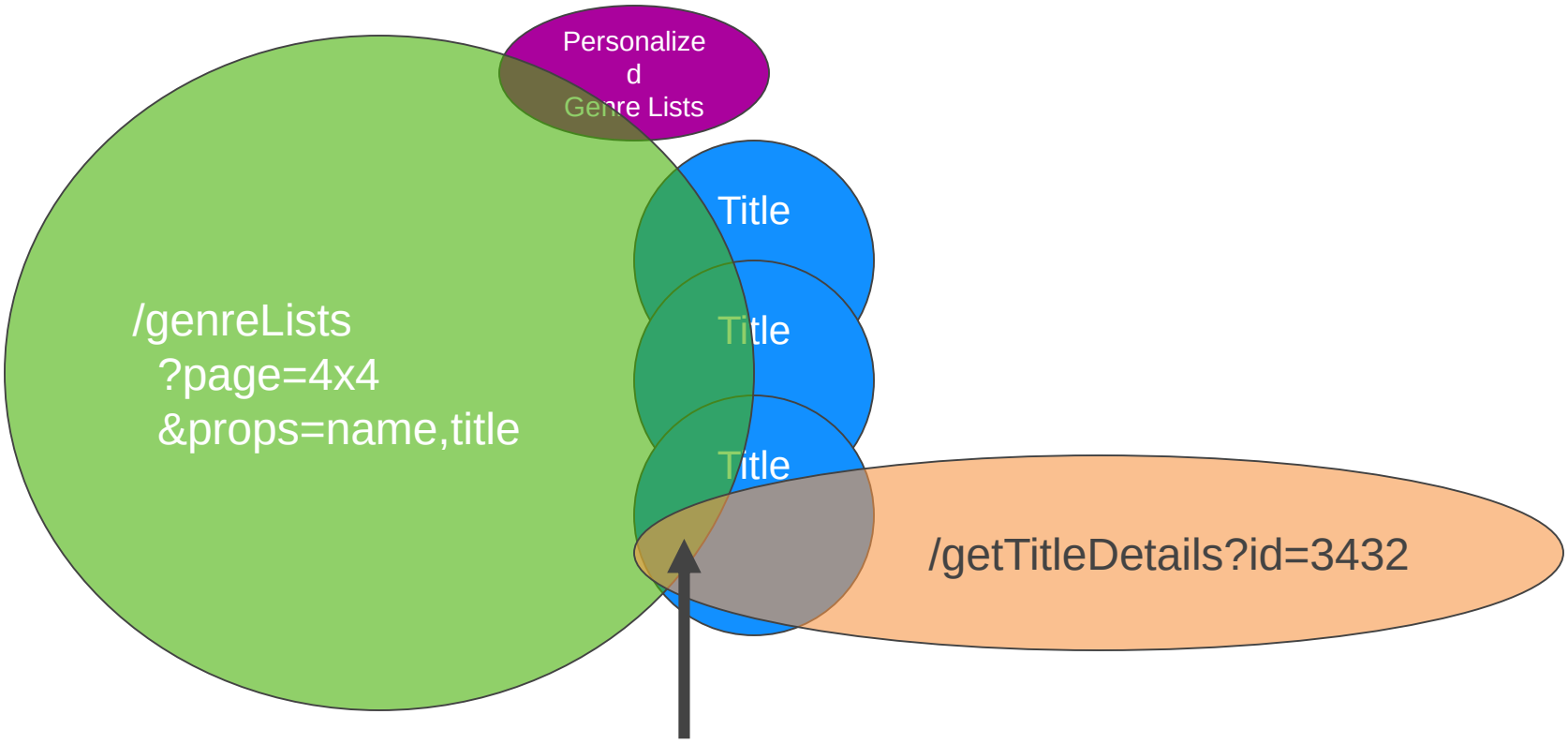

/genreList?pageSize=13x15
/titleDetails?id=342
/search?text=netflix
NETFLIX
Continue Watching for GKB
Top Picks for GKB
Watch It Again
Recently Added
HOUSE of CARDS
House of Cards
THE FALL
HEMLOCK GROVE
MARCO POLO
BORGIA
Rita

RPC: One Request per View

/genres/5632122?page=4x4?props=name,boxshot

```
titles/95622122
[
{{
   id: 2523,
   name: “House of Cards”,
 },boxshot: “http://…/018-192x50.png”,
 { rating: 5,
   bookmark: 52119562,
   boxshot: “http://…/018-192x50.png”,
 },director: “David Fincher”,
 { description: “This Emmy-winning original
   id: 78412,    series    stars Golden Globe winner
   boxshot: “http://…/018-192x50.png”
 },   Francis Underwood, who will stop at nothing to
 {    conquer the halls of power in Washington D.C.
   id: 6342,    secret weapon: his gorgeous, ambitious, and
   boxshot: “http://…/018-192x50.png”
}},

  ...
]
```

RPC: No Cache Consistency



Same info, two different URLS!

RPC

- Low Latency
- Small Messages

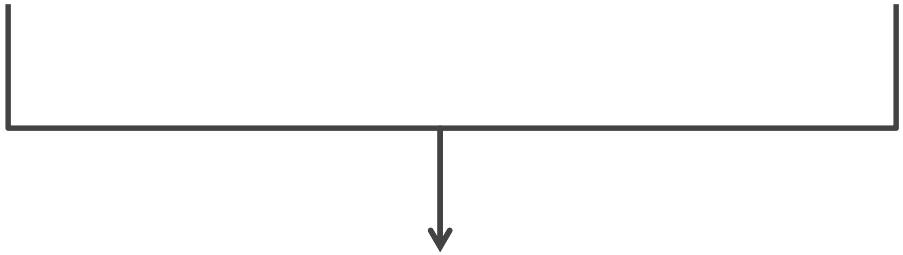
- Tight coupling
- Manual Cache Management

REST

- Cache Consistency
- Loose Coupling

RPC

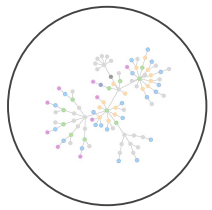
- Small Message Sizes
- Low Latency

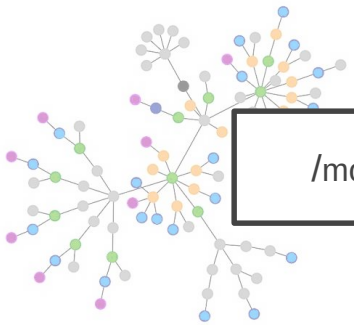


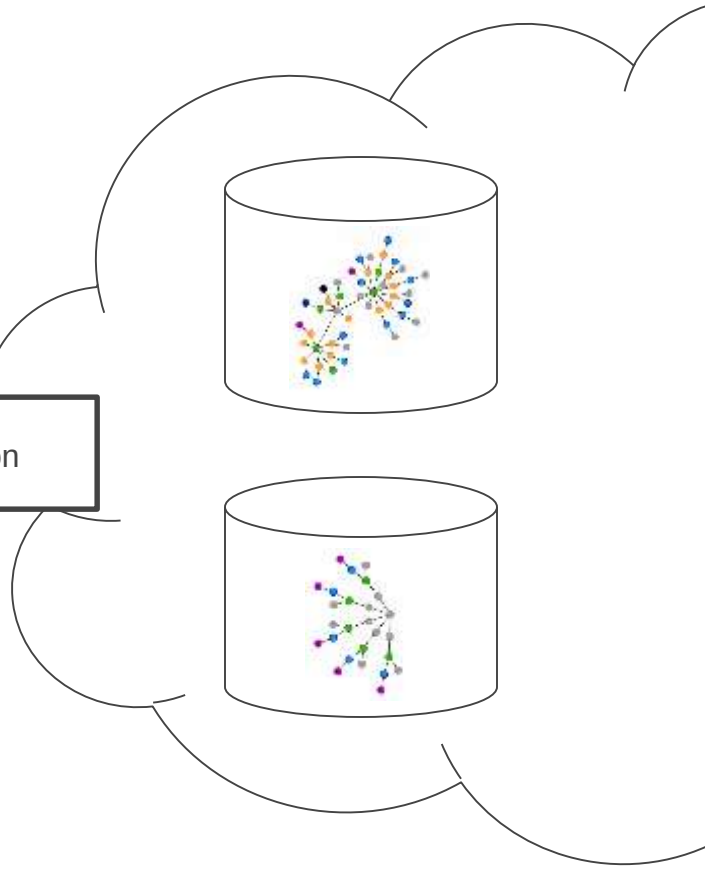
?

FALCOR

One Model Everywhere






/model.json

The Data is the API

JSON

```
var member = {
     name: “Steve McGuire”,
     occupation: “Developer”,
     location: {
          country: “US”,
          city: “Pacifica”,
          address: “344 Seaside”
     }
};

print(member[“location”][“address”]);
```

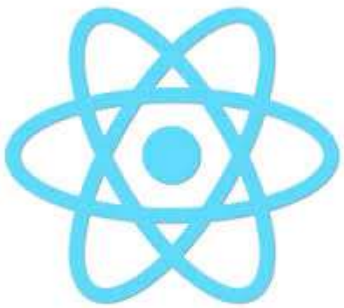
Falcor JSON Model

```
var member = new falcor.Model({
   source: new HttpSource(‘/member.json’)});

member.getValue([“location”, “address”])
   .toPromise()
   .then(print);
```

Bind to the Cloud

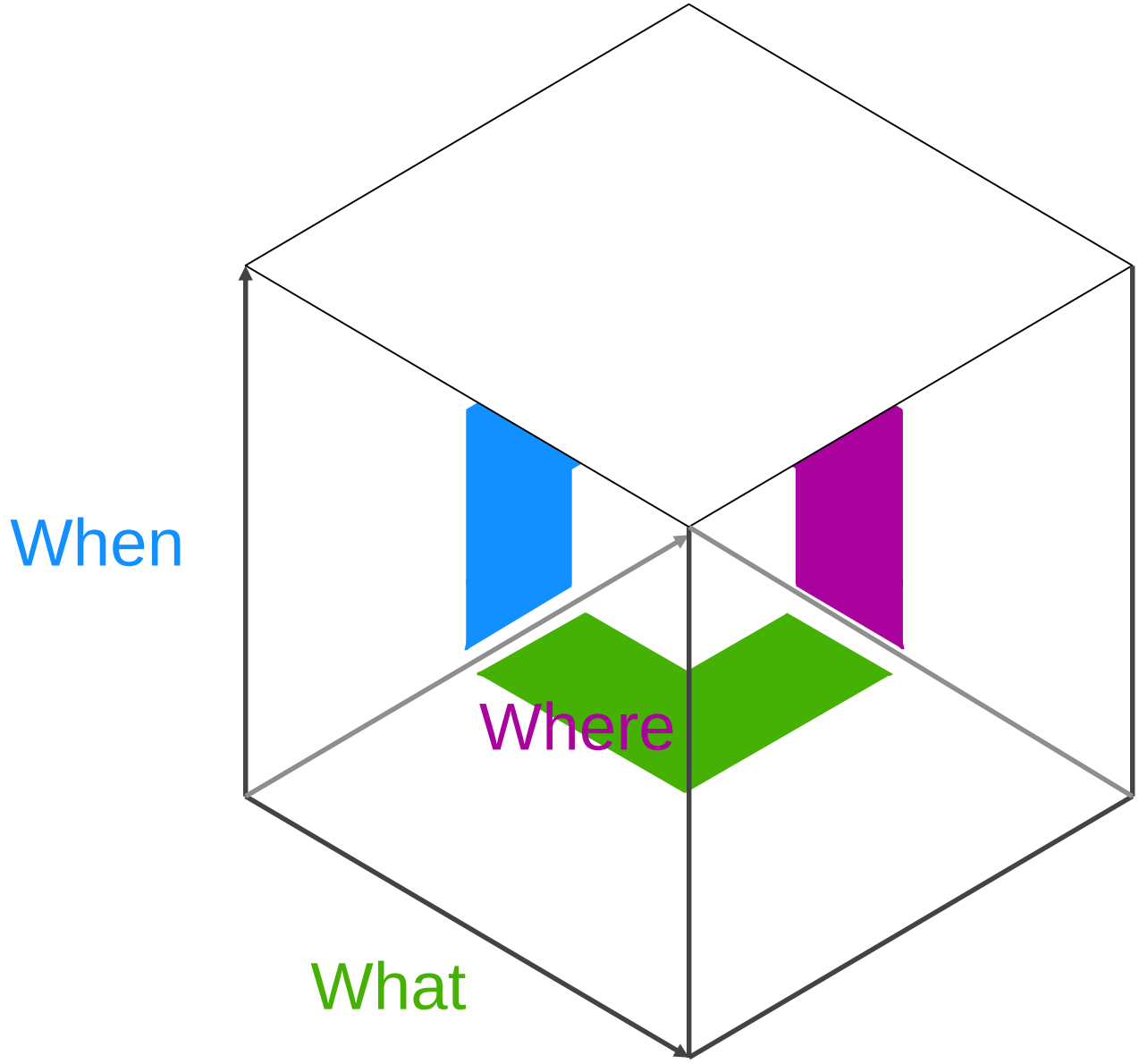
MVC Framework Integration: React

```
var memberModel =
    new falcor.Model(new falcor.HttpSource("/member.json"));

var MemberView = React.createClass({
    mixins: [FalcorComponent],
    renderContent: function() {
        return this.props.model.get(
            ["name"],
            function(name) {
                return <div>Welcome {name}</div>;
            });
    },
    renderLoading: function() {
        return <div><img src="spinner.gif"></div>;
    }
});

React.render(<MemberView model={memberModel}/>);
```



Angular 2.0 (preview)

```
<div>
    {{ model.getValue([“person”, “name”])|rx}} lives at
    {{ model.getValue([“person”, “address”])|rx }}
</div>

Steve McGuire lives at 344 Seaside
```


When
Where
What

What data do you want?

Nothing — Every value

What data do you want?

```
var member = new falcor.Model({
  source: new falcor.HttpSource("/member.json")
});

member.get(["genreLists", 0, "name"])
```

```
member.
   get(["genreLists", {to:2}, {from:0, to:1}, "boxshot"]);
```

When can data arrive?

Now (sync) → Later (async)

When: Reactive Programming

Choose any async model:

- Promises (ES6)
- Observable (Proposed for ES7)
- Node-style Callbacks
- Node Streams

Where is the data?

Local → Remote

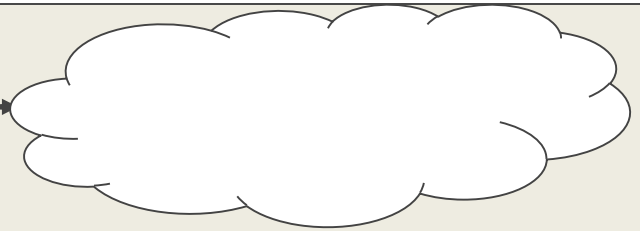
JSON

FalconServer

```
var member = new falcon.Model({
    source: new falcon.HttpSource("/member.json")
});
```

```
var member = new falcon.Model({cache: {
    name: "Steve McGuire",
    occupation: "Developer",
    location: {
        country: "USA",
        city: "Pacifica",
        address: "3344 Seaside"
    }
}});
```

```
member.get( ["location", "address"]).
    toPromise().
    then(json => response.write(json));
```

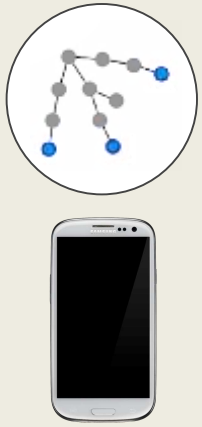


Falcor

```
var member = new falcor.Model({cache:{
  name: "Steve McGuire",
  occupation: "Developer",
  location: {
    country: "US",
    city: "Pacifica",
    address: "344 Seaside"
  }
}});

member.get(["location",["country", "city", "address"]],
  (address) => print(address)).
  toPromise();
```

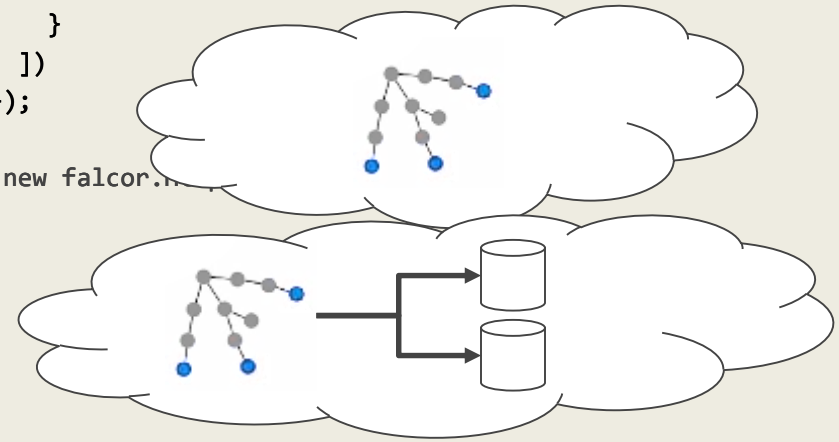
```
new falcor.HttpSource("member.json")
```

```
(new falcor.HttpServer(member)).
  listen(80);
```



Virtual Falcor Server

```
var member = new falcor.Model({cache:{
  name: "Steve McGuire",
  occupation: "Developer",
  location: {
    country: "US",
    city: "Pacifica",
    address: "344 Seaside"
  }
}});

var router = new falcor.Router([
  {
    route: ["member",["name","occupation"]],
    get: (pathSet) =>
      memberDB.
        exec(`SELECT ${pathSet[1].join(',')}
              FROM user
              WHERE id = ${request.cookies.userid}`)
  },
  {
    route: ["member","location",["country", "city", "address"]],
    get: (pathSet) =>
      locationServer.
        getLocation(request.cookies.userid).
        then(location => ({
          member: { location: getProps(location, pathSet[2]) }
        })
  }
]);

(new falcor.HttpServer(member)).
  listen(80);
```



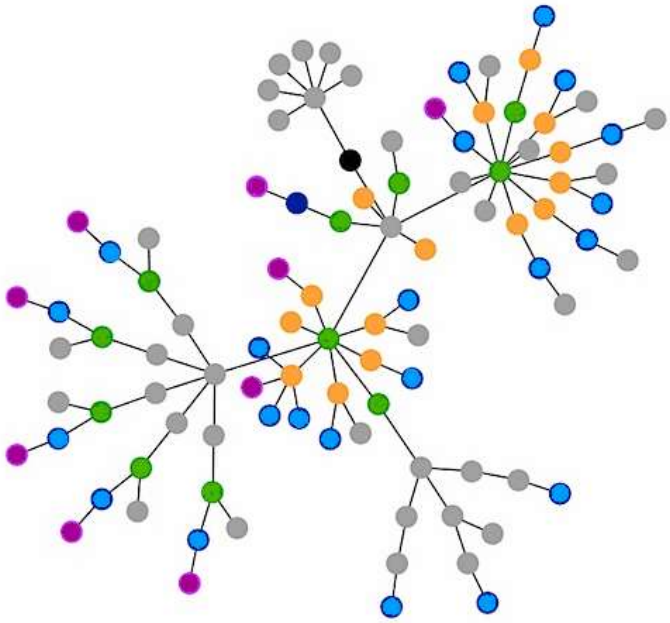
How do we do it efficiently?

Falcor Optimization

- Batching
- Caching
- Path Optimization

Building Netflix with Falcor

Defining a Model



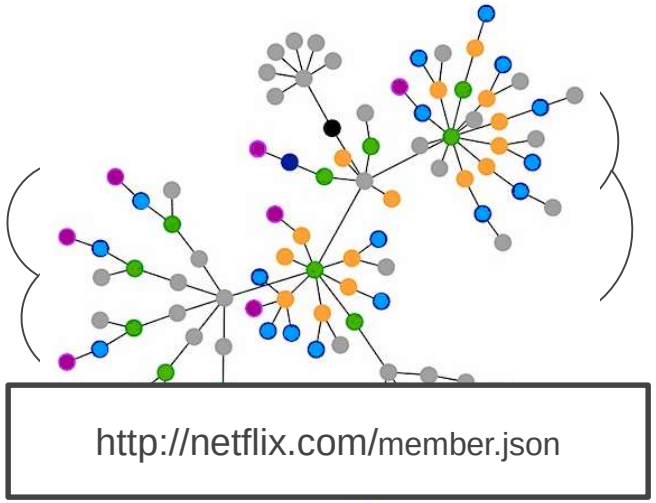
NETFLIX


SAMSUNG
TV Shows
FRIEND
NEW EPISODE
how I met your mother
SHERLOCK
ULTIMATE SPIDER-MAN
DOCTOR WHO
TopGear
Classic TV Shows

Netflix Member JSON Model

```
var member = {
   genreLists: [
      {
         name: "Suggestions For You",
         titleList: [
            {
               id: 523,
               name: "Friends",
               rating: 5,
               // more fields
            },

            // "Brain Games", "Spartacus"...
         ],
      },
      // "New Releases", "British TV Shows", ...

   ]
}
```

We're *almost* ready to move our model into the cloud.

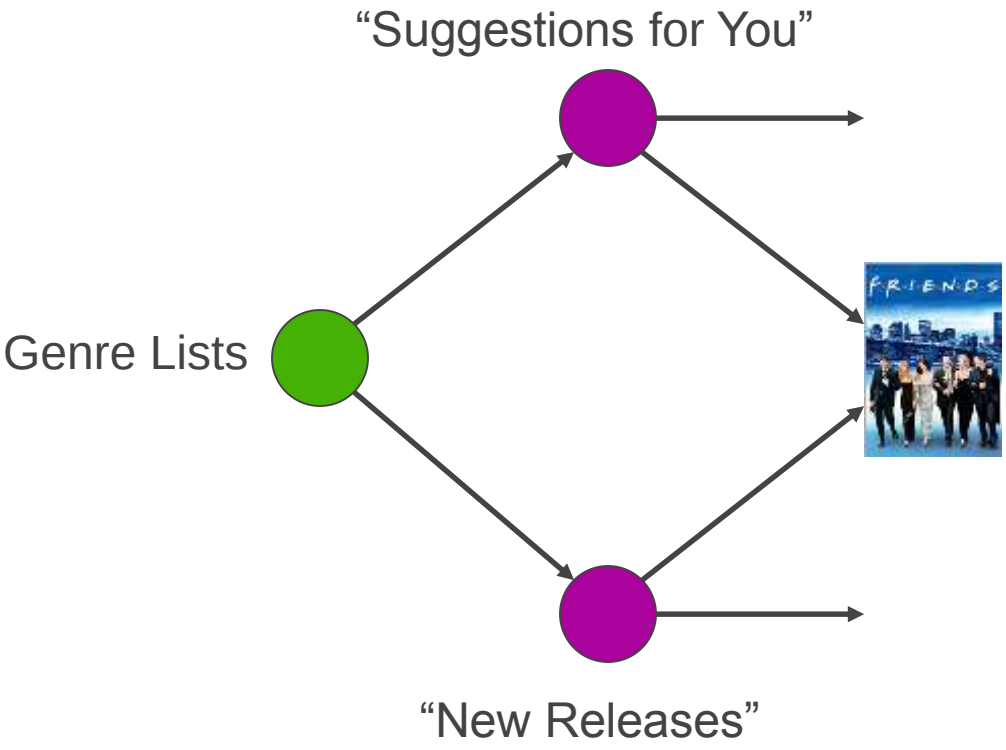


There’s just one problem…

Netflix's Domain Model is a Graph

JSON is for Trees.

When we convert a graph to a JSON we get duplicates.



Introducing JSON Graph

JSON Graph

- Graph format for JSON
- Tree with Symbolic Links
- *Every value type is a resource*

JSON

```
var model = {
  genreLists: [
    {
      name: "Suggestions For You"
      titlesList: [
        {
          id:
          rat
        }
      ]
    },
    {
      name: "New Releases",
      titlesList: [
        {
          id: 956,
          rating: 3
        }
      ]
    }

  ]
}
```

```
model[“genreLists”][0]["titlesList"][0][“rating”] = 5
```

JSON Graph

```
var model = new falcor.Model({ cache: {
  genreLists: {
    “0”: {
      name: "Suggestions For You"
      titlesList: {
        “0”: [“titlesById”, 956],
        length: 75
      }
    },
    “1”: {
      name: "New Releases",
      titlesList: {
        “0”: [“titlesById”, 956],
        length: 75
      }
    },
    length: 50
  },
  titlesById: {
    “956”: {
      rating: 5
    }
  }
});
```

```
id: 956,
name: "Friends",
rating: 3
```

```
model.setValue([“titlesById”, 956, “rating”], 5)
```

JSON Graph: Small Resources

```
{
   genreLists: {
      “0”: {
         name: “Suggestions for You”,
         titlesList: {
            “0”: [“titlesById”, 956],
            “1”: [“titlesById”, 192],
            // more titles
            length: 75
         },
      },
      length: 40
   },
   titlesById: {
      “956”: { name: “Friends”, rating: 5 }
   }
}
```

Fine-Grained Resources

JSON Graph

```
["genreLists", "length"]
["genreLists", *, name]

["titlesById", 23432, "name"]
["titlesById", 23432, "rating"]
["titlesById", 23432, "description"]
["titlesById", 23432, "bookmark"]
["titlesById", 23432, "director"]
// more fields
```

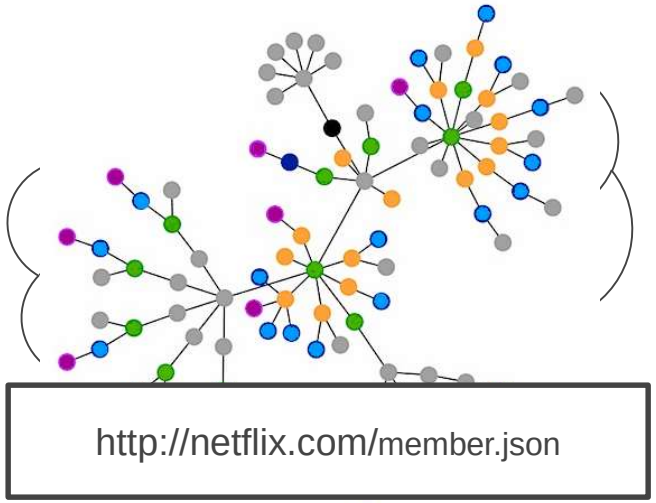
HTTP/REST

```
/genreLists

/titles/23432
```

Idempotent Operations

Falcor Model	HTTP/REST
get	GET
set	PUT

Building a Virtual Model



Virtual Model: Defining Routes

```
var member = new falcor.Model({
  router: new falcor.Router([
    {
      route: ["genreLists", Router.INTEGERS, Router.INTEGERS],
      // example match: ["genreLists", [1,2,3], [1,5,2]]
      pathSet => {
         // retrieve data from member's personalized list DB
         // return as JSON Graph
      },
    },
    {
      route: [“titlesById”, Router.INTEGERS, [“name”, “rating”]],
      // example match: [“titlesById”, [2343,5123], [“name”]]
      pathSet => {
        // retrieve data from title DB
        // return as JSON Graph
      }
    }
  ])
});
```

Title Route

```
var member = new falcor.Model({
  router: new falcor.Router([
    // genre list route snipped
    {
      route: [“titlesById”, Router.INTEGERS, [“name”, “rating”]],
      // example match: [“titlesById”, [2343,5123], [“name”]]
      pathSet => {
        var titleIds = pathSet[1], fields = pathSet[2];
        titleStore.
          getTitlesByIds(titleIds).
          then(titles => {
            var titlesById = {};
            titles.forEach(title => {
              titlesById[title.id] = {};
              fields.forEach(field =>
                titlesByid[title.id][field] = title[field]);
            });

            return { titlesById: titlesById };
          });
    }
  ])
});
```

Virtual Model Path Evaluation

```
var member = new falcor.Model({
  source: new falcor.HttpSource('member.json')});
  cache: {
    genreLists: {
      "0": {
        "0": ["titlesById", 926]
      }
    },
    titlesById: {
      "926": {
        name: "Friends"

      }
    }
  });
```

```
member.
  getValue(["genreLists", 0, 0, "name"]).
  toPromise();
```

```
var member = new falcor.Model({
  router: new falcor.Router([
    {
      route: ["genreLists", Router.INTEGERS, Router.INTEGERS],
      get: pathSet => {/* retrieve data from lists DB */},
    },
    {
      route: ["titlesById", Router.INTEGERS, ["name", "rating"]],
      get: pathSet => {/* retrieve data from titles DB */}
    }
  ]);
});
```

```
{
  genreLists: {
    "0": {
      "0": ["titlesById", 926]
    }
  }
}
```

```
{
  titlesById: {
    "926": {
      name: "Friends"
    }
  }
}
```

```
  cache: {
    genreLists: {
      "0": {
        "0": ["titlesById", 926]
      }
    },
    titlesById: {
      "926": {
        name: "Friends"
      }
    }
  }
});
```

```
member.get(["titlesById", 926, "name"])
```

Virtual Model Path Evaluation

```
var member = new falcor.Model({
  source: new falcor.HttpSource('member.json'),
  cache: {
    genreLists: {
      "0": {
        "0": ["titlesById", 926]
      }
    },
    titlesById: {
      "926": {
        name: "Friends"

      }
    }
  }
});
```

```
var member = new falcor.Model({
  router: new falcor.Router([
    {
      route: ["genreLists", Router.INTEGERS, Router.INTEGERS],
      pathSet => { /* retrieve data from lists DB */},
    },
    {
      route: ["titlesById", Router.INTEGERS, ["name", "rating"]],
      pathSet => {{ /* retrieve data from titles DB */}
    }
  ]);
```

```
member.
  getValue(["genreLists", 0, 0, "rating"]).
  toPromise();
```

```
member.get(["titlesById", 926, "rating"])
```

The World Wide Web is Flat

The Type of the WWW

```
Map<String, Resource>
```

The WWW is Flat

GET http://../genreLists/0/0/name

302 http://../titles/23432/name

GET http://../titles/23432/name

“Friends”

The WWW is Flat (continued)

GET http://../genreLists/0/0/rating

302 http://../titles/23432/rating

GET http://../titles/23432/rating

5.0

No Hierarchy

GET http://../genreLists/0/0/name

302 Redirect
http://../genreLists/0/0/*
http://../titles/23432/*

Sets Are Idempotent

```
model.set(
   {
      path: [“titles”, 234324, “rating”],
      value: 5
   });
```

Non Idempotent Operations

```
model.call(
    [“genreList”,0,”add”],
    [[“titles”, 234234]])
```

Function Calls

- Can invoke function that lives anywhere in Graph
- Functions can only be called, not downloaded
- Functions can invalidate any resource in cache

Falcor

- Cache Consistency
- Loose Coupling

- Low Latency
- Small Message Sizes

Additional Features

- Automatic Cache Management
  - Items purged based on order in LRU
  - Custom size can be applied to items
  - One total size for all heterogenous types
- Fast Dirty Checking
  - All set operations on leaves mark branches as dirty
  - Can be used for fast model diff a la immutable types


NETFLIX
SIRENS
SHERLOCK


NETFLIX
Sci-Fi & Fantasy
Thrillers
HACKERS

Open-Source Soon

- Should be available in the next few months
- Looking for help integrating with existing MVC frameworks
- @falcorjs

Questions?

JSON Graph and REST

[“videosById”, 512]

[“genreLists”, 0, 0]

[“genreLists”,{to:5},{to:5},

/videosById/512

/genreLists?col=0&row=0

/genreLists

JSON Graph

- Serializable Graph
- Fine-Grained

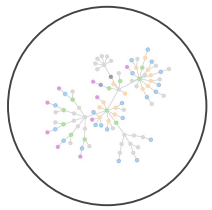
Falcor Client

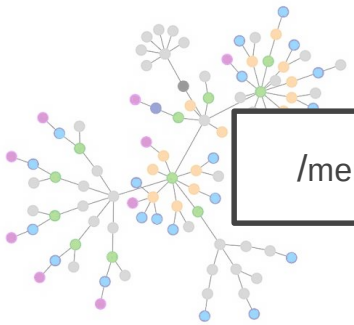
```
var member = new falcor.Model({source:
  new falcor.HttpSource("member.json")});
```

```
member.
  get(["location",["address","city"]],
     (address) => <div>{address}></div>).
<div>{address}></div>)
  subscribe(jsx => updateUI(jsx));
```

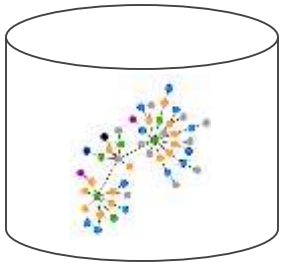
Falcor Server

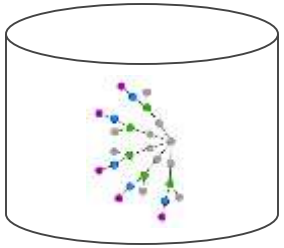
```
var member = new falcor.Model({
     name: "Steve McGuire",
     occupation: "Developer",
     location: {
          country: "US",
          city: "Pacifica",
          address: "344 Seaside"
     }
}});

(new falcor.HttpServer(member)).
     listen(80);
```





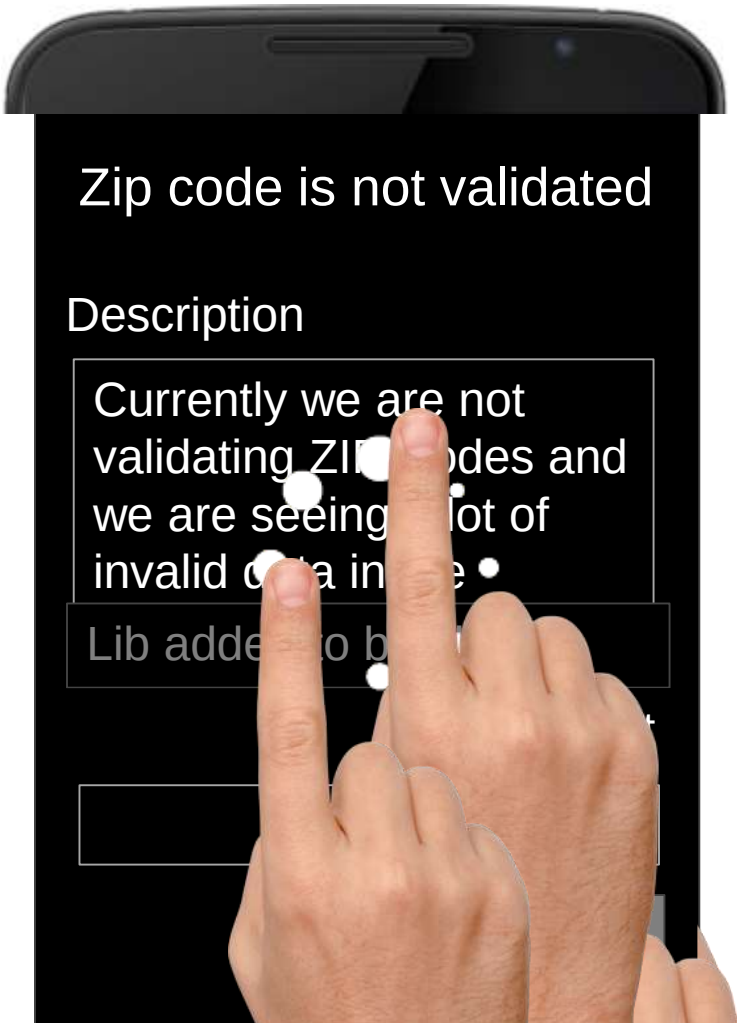
/member.json



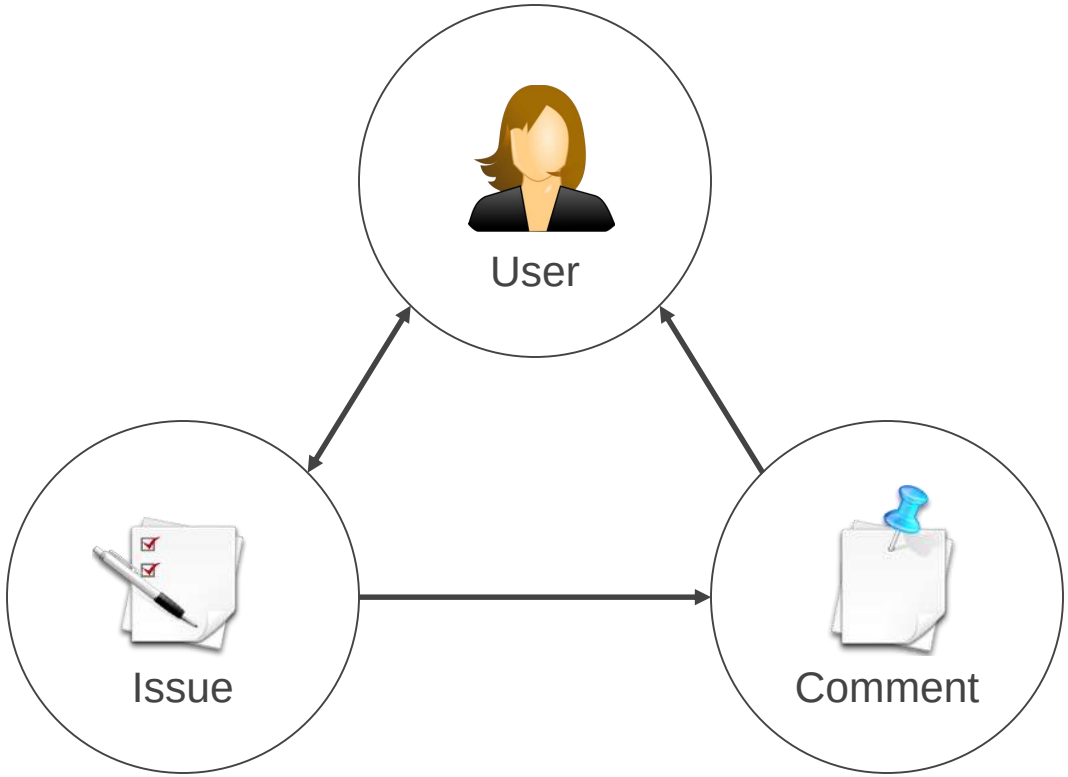


A Simple Issue Tracking App

Issue Tracking



Issue Tracking: Domain Model



Let’s talk about REST.

REST: Issue Tracking End Points

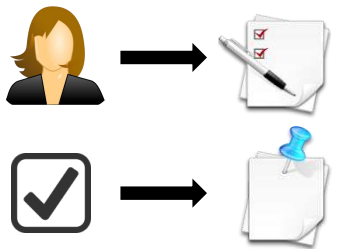
/user/{userId}

/issue/{issueId}

/comment/{commentId}

/user/{userId}/issues

/issue/{issueId}/comments



User Resource: /user/5232

```
{
  name: “Kim Trott”,
  title: “Director”,
  twitter: “@alwayson”,
  email: “alwayson@netflix.com,
  // more fields…
}
```

Issue Resource: /issue/926

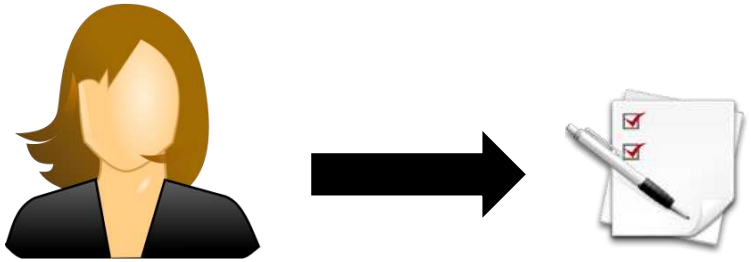
```
{
  name: “Zip code is not validated”,
  description: “We need to apply…”,
  status: “Resolved”,
  dueDate: 956568342,
  // more fields…
}
```

Comment Resource: /comment/612

```
{
  text: “What about the Canadians? Don’t we need to validate postal codes?”,
  user: “/user/5232”
}
```

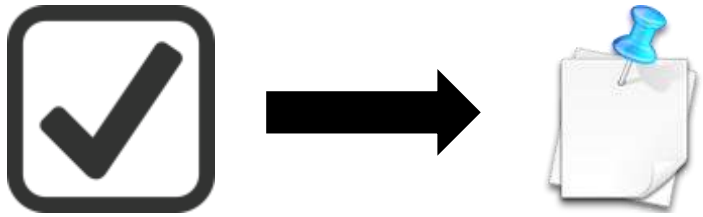
User Issues: /users/5232/issues

```
[
   “/issues/926”,
   “/issues/696”,
   “/issues/651”,
   “/issues/2239”,
]
```

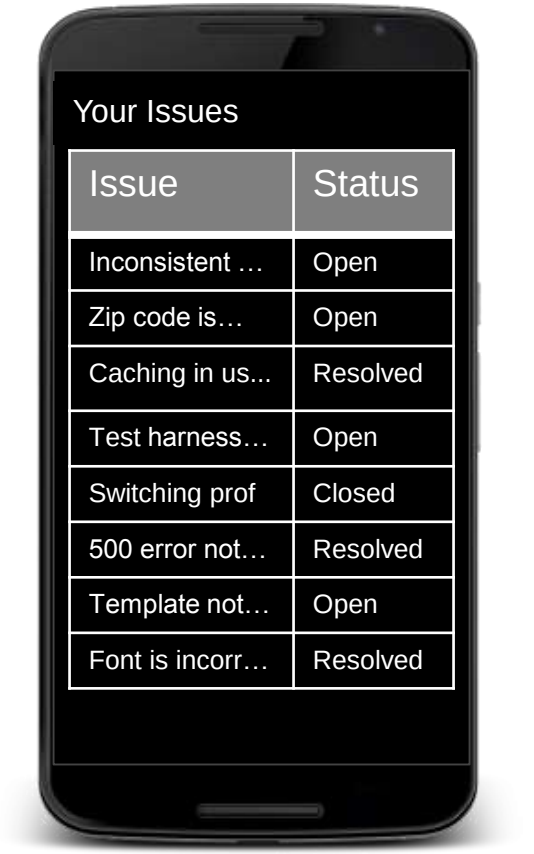


/issues/926/comments

```
[
   “/comments/912”,
   “/comments/555”,
   “/comments/61”,
   “/comments/610
]
```



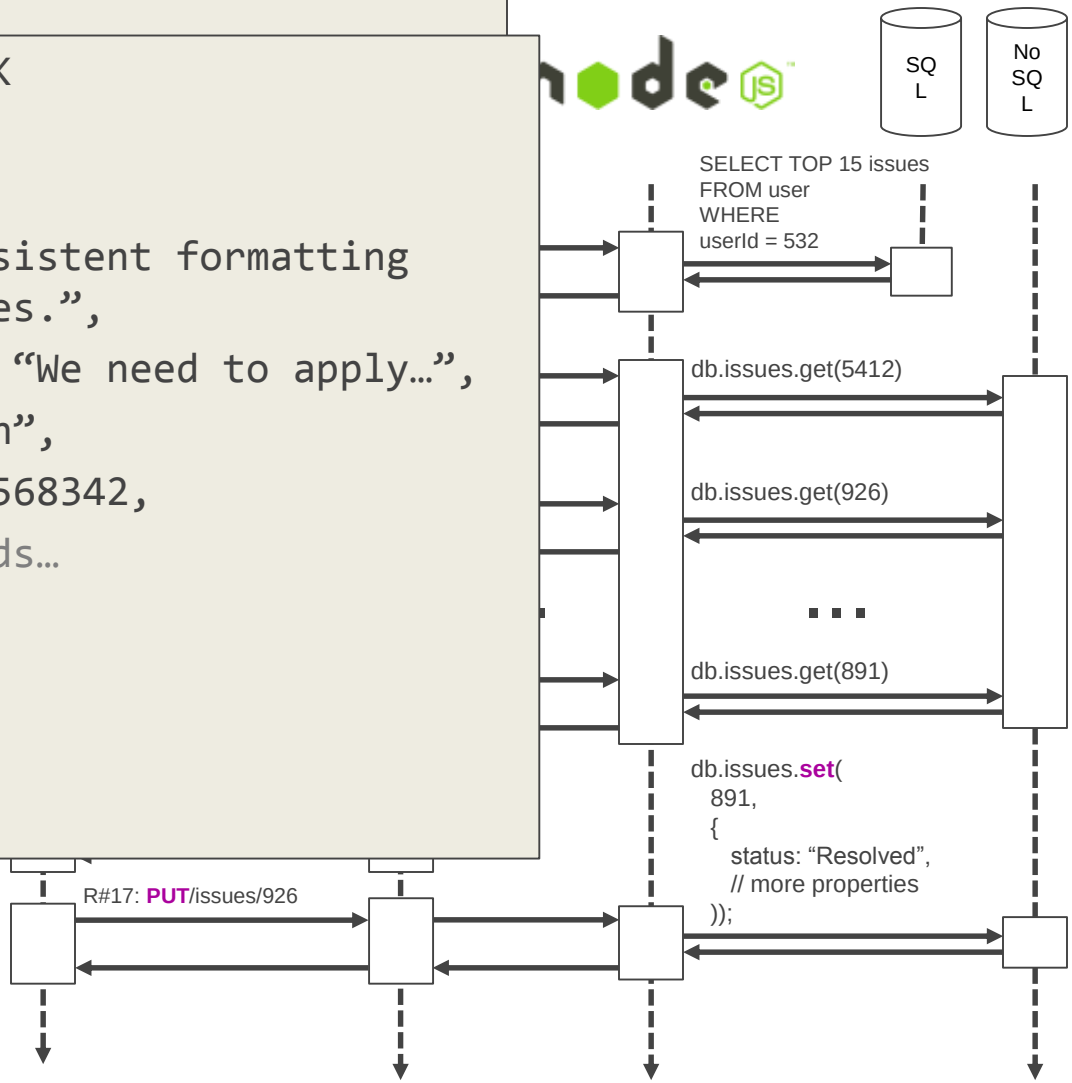
Issue Tracking with REST

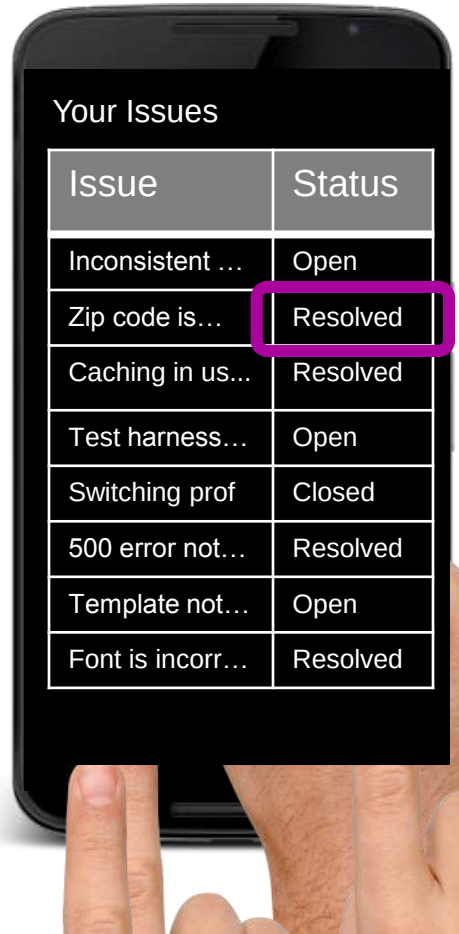


Issue Tracking with REST

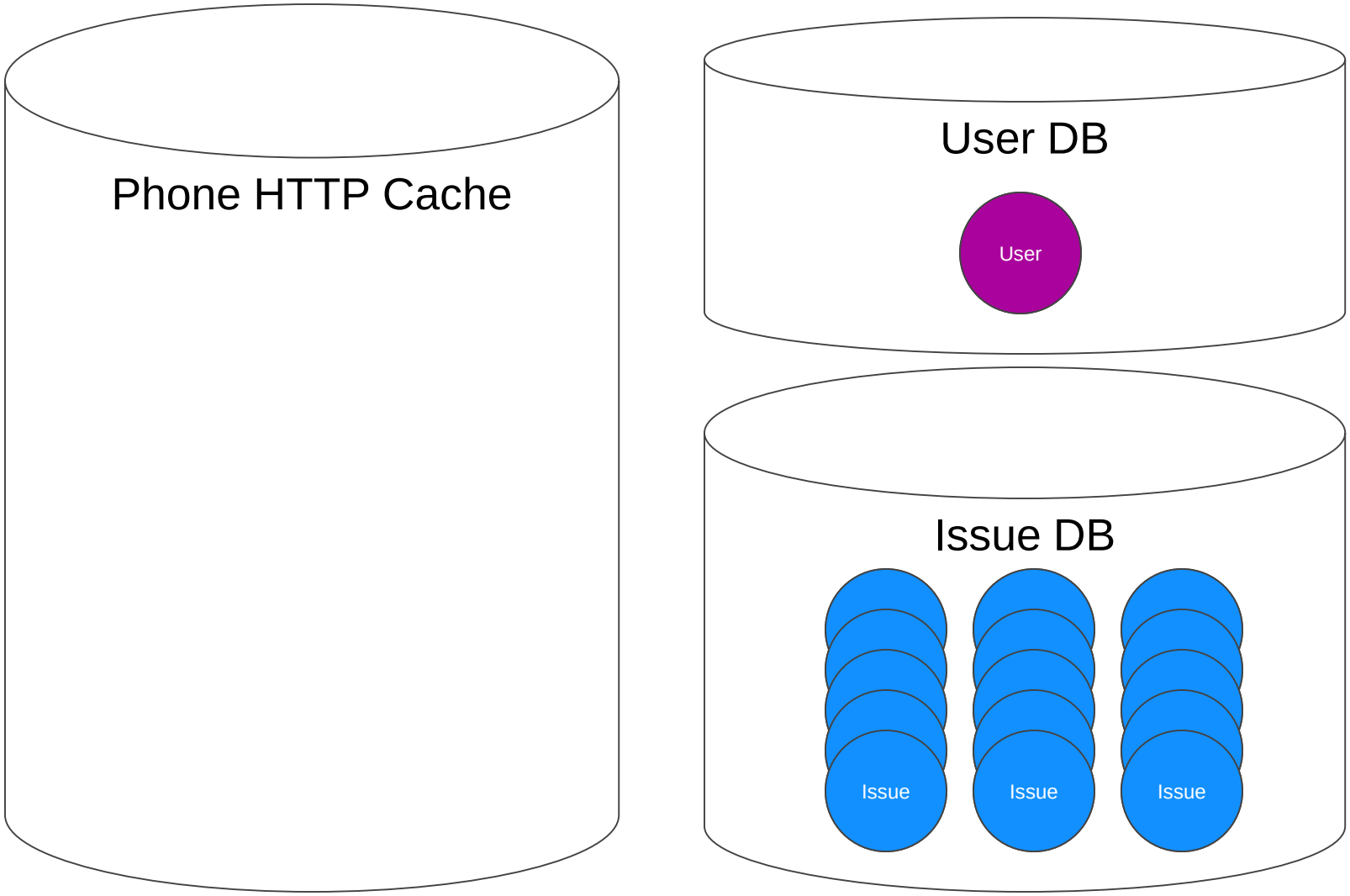
```
0 OK

/5412”,
```

```
200 OK

“Inconsistent formatting
to dates.”,
ption: “We need to apply…”,
: “Open”,
e: 956568342,
e fields…
```





REST: Cache Coherence



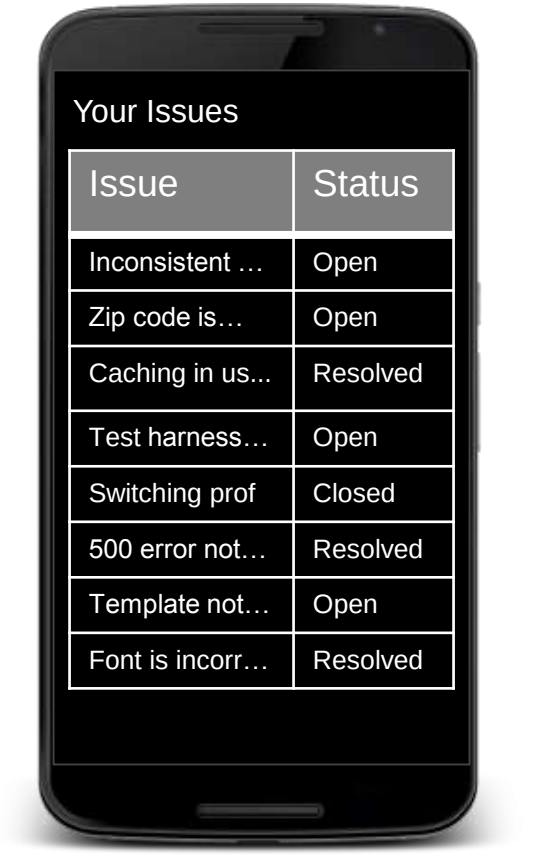
REST = Laaaaaaaattteeennccccyy

Overfetching with REST

/issues/26343

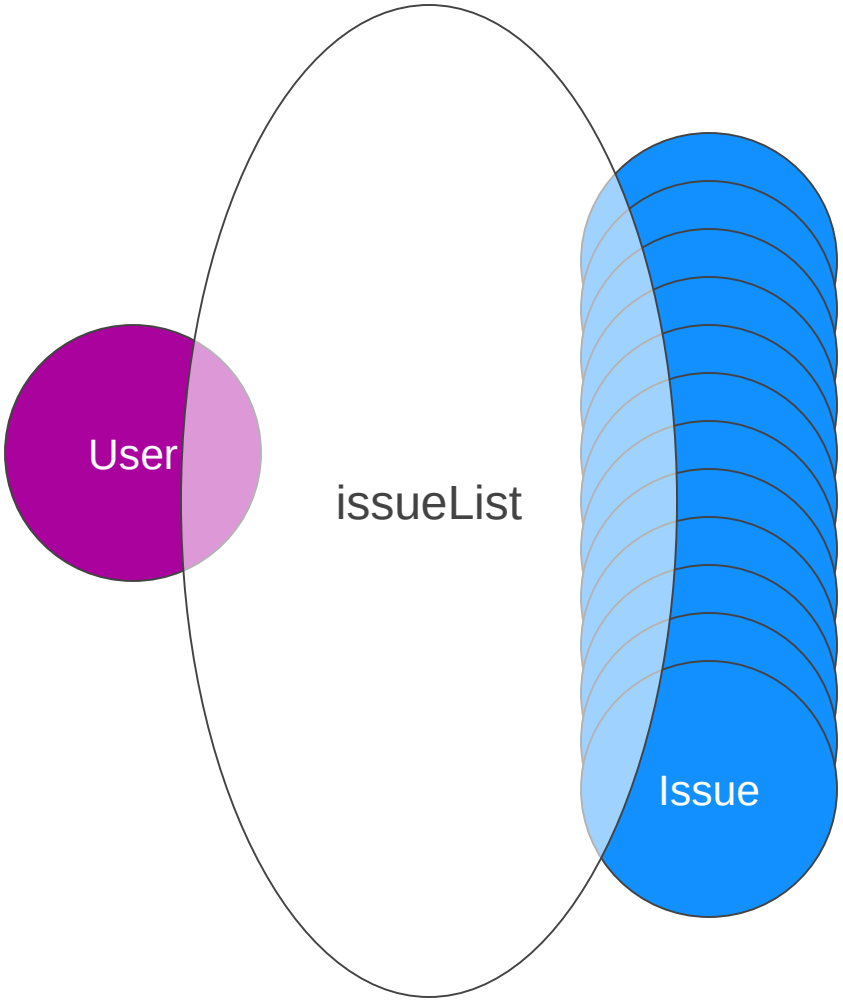
```
{
  name: “Zip code is not validated”,
  description: “We need to apply…”,
  status: “Resolved”,
  dueDate: 956568342,
  // more fields…
}
```

Issue Tracking with RPC



RPC: Stale Caches

RPC: Gerrymandered “Resource”



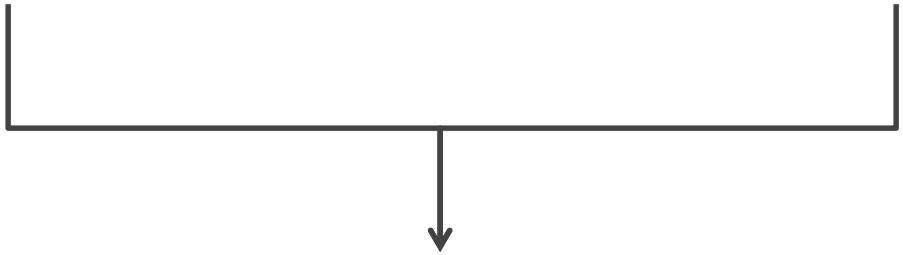
RPC: Tight Coupling

REST

- Cache Consistency
- Loose Coupling

RPC

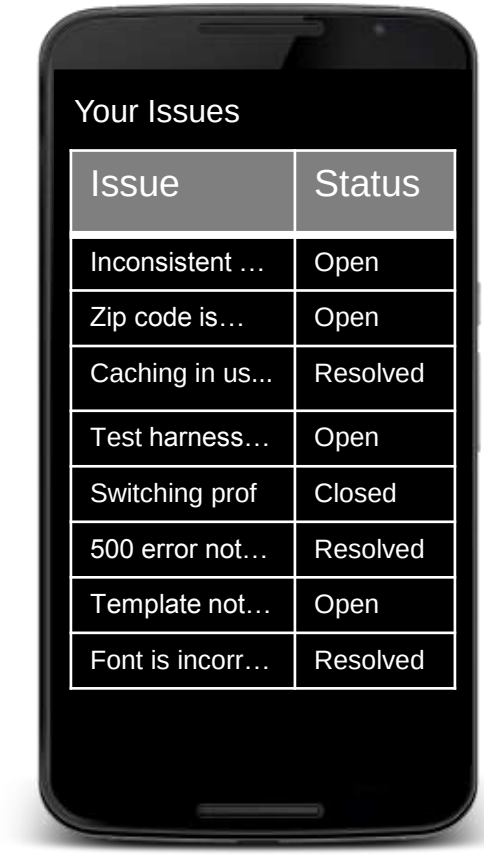
- Small Message Size
- Low Latency



- Distributed Architecture for Web Applications
- Support Idempotent Operations *and* RPC calls
- Maximize Cache Utility, Minimize Message Size

How it works

Issue Tracking with Falcor



- Distributed Architecture for Web Applications
- Maximize Cache Utility, Minimize Message Sizes
- Support for Idempotent Operations and RPC calls

Most domain models are **Graphs**.

JSON Graph

```
var model = {
  user: {
    id: 5232,
    name: “Kim Trott”,
    // more props…
    issues: [
      {
        id: 926
        name: “Zip code is not validated”,
        // more props…
        assignedTo: { … }
        comments: [
          {
            id: 612
            text: “Is there a library?”,
            user: { … }
          },
          // more comments…

        ]
      },
      // more issues

    ]
  }
}
```

```
var model = new falcor.Model({cache: {
  user: {
    id: 5232,
    name: “Kim Trott”,
    // more properties…
    issues: [
      {
        id: 926
        name: “Zip code is not validated”,
        // more properties
        assignedTo: { … }
        comments: [
          {
            id: 612
            text: “Is there a library?”,
            user: { … }
          },
          // more comments…

        ]
      },
      // more issues

    ]
  },
}});
```

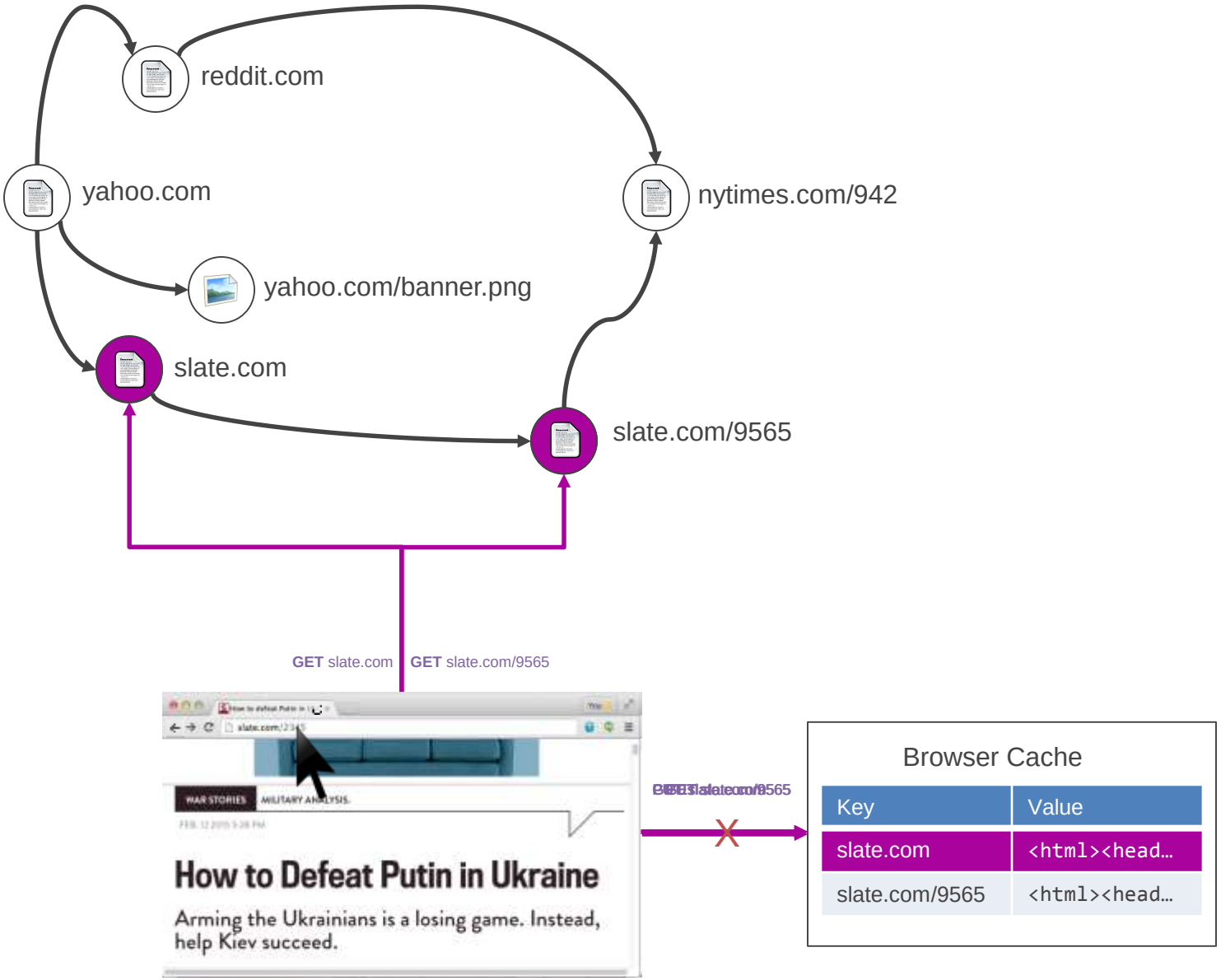
```
model[“user”][“issues”][0][“comments”][“0”][“text”]
```

REST End Points

REST

/issues/{id}

/users/{id}


reddit.com
yahoo.com
nytimes.com/942
yahoo.com/banner.png
slate.com
slate.com/9565
GET slate.com
GET slate.com/9565
WAR STORIES
MILITARY ANALYSIS
How to Defeat Putin in Ukraine
Arming the Ukrainians is a losing game. Instead, help Kiev succeed.
Browser Cache
Key
Value
slate.com
<html><head…
slate.com/9565
<html><head…

The Power of Idempotence

GET /genreLists/0/name

PUT /genreLists/0/0/rating
5

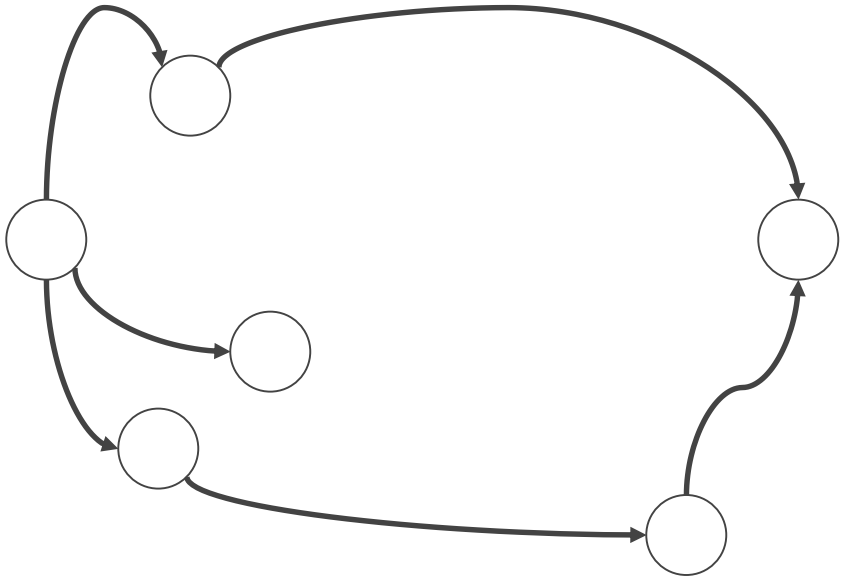
PathSets

[“genreLists”, {from: 0, to: 5}, “titles”, {to:3}, [“name”, “rating”]]

How do we do it efficiently?

1. Batching
2. Caching
3. Path Optimization

Most domain models are **Graphs**.



How do we store a graph as JSON *without* duplicates?

Introducing JSON Graph

JSON

```
var model = {
  genreLists: [
    {
      name: "Suggestions For You"
      titlesList: [
        {
          id:
          rat
        }
      ]
    },
    {
      name: "New Releases",
      titlesList: [
        {
          id: 956,
          rating: 3
        }
      ]
    }

  ]
}
```

```
model[“genreLists”][0]["titlesList"][0][“rating”] = 5
```

JSON Graph

```
var model = new falcor.Model({ cache: {
  genreLists: {
    “0”: {
      name: "Suggestions For You"
      titlesList: {
        “0”: [“titlesById”, 956],
        length: 75
      }
    },
    “1”: {
      name: "New Releases",
      titlesList: {
        “0”: [“titlesById”, 956],
        length: 75
      }
    },
    length: 50
  },
  titlesById: {
    “956”: {
      rating: 5
    }
  }
});
```

```
id: 956,
name: "Friends",
rating: 3
```

```
model.setValue([“titlesById”, 956, “rating”], 5)
```

Expire at Absolute Time

```
var member = new falcor.Model({
     location: {
          country: {
               $type: “sentinel”,
               $expires: 1200063422,
               value: “US”
          }
     }
});

(new falcor.HttpServer(member)).
     listen(80);
```

Get as JSON

```
var member = new falcor.Model({
     name: “Steve McGuire”,
     occupation: “Developer”,
     location: {
          country: “US”,
          city: “Pacifica”,
          address: “344 Seaside”
     }
});
```

```
member.get(
     [“name”],
     [“location”, “address”]).
     subscribe(json => console.log(json));
```

```
{
  “json”:{
     name: “Steve McGuire”,
     location: {
       address: “344 Seaside”
     }
  }
}
```

Get as Path Values

```
var member = new falcor.Model({
     name: “Steve McGuire”,
     occupation: “Developer”,
     location: {
          country: “US”,
          city: “Pacifica”,
          address: “344 Seaside”
     }
});
```

```
member.get(
     [“name”],
     [“location”, “country”]).
     toPathValues()
     subscribe(pathValue =>
          console.log(jpathValue));
```

```
{path: [“name”], value: “Steve McGuire”}
{path: [“location”, “country”], value: “US”}
```

Get with Selector Function

```
var member = new falcor.Model({
     name: “Steve McGuire”,
     occupation: “Developer”,
     location: {
          country: “US”,
          city: “Pacifica”,
          address: “344 Seaside”
     }
});

member.
     get(
          [“name”],
          [“location”, “country”],
          (name, country) =>
               <div>{name} lives in {country}</div>).
     subscribe(jsx =>
          console.log(jsx));
```

```
<div>Steve McGuire lives in US</div>
```

Get with *Fast* Selector Function

```
var member = new falcor.Model({
     name: “Steve McGuire”,
     occupation: “Developer”,
     location: {
          country: “US”,
          city: “Pacifica”,
          address: “344 Seaside”
     }
});

member.
     get(
          [“name”],
          [“location”, “country”],
          () =>
               <div>{this.getValueSync([“name”)} lives
               in {this.getValueSync([“location”,
“country”])}</div>).
     subscribe(jsx =>
          console.log(jsx));
```

```
<div>Steve McGuire lives in US</div>
```

Introducing Dense JSON

Selector Fn Arguments get Dense JSON

```
member.
  get(
    [“location”, [“country”, “address”]).
  subscribe(json => console.log(json));
```

```
{
    location: {
        country: “US”,
        address: “344 Seaside”
    }
}
```

```
member.
  get(
    [“location”, [“country”, “address”]],
    (location) => locationx).
  subscribe(json => console.log(json));
```

```
{
    country: “US”,
    address: “344 Seaside”
}
```

Falcor Client

```
var member = new falcor.Model(
  new falcor.HttpSource("member.json"),
  {
      location: {
          city: "Pacifica",
      }
  });
member.
  get(["location",["country","city"]],
     (address) =>
         <div>{city},{country}></div>).
  subscribe(jsx => updateUI(jsx));
```

Falcor Server

```
var member = new falcor.Model({
     name: "Steve McGuire",
     occupation: "Developer",
     location: {
          country: "US",
          $type: "sentinel",
          $expires: 0,
          value: "US"
     },
     city: "Pacifica",
     address: "344 Seaside"
     }
});

(new falcor.HttpServer(member)).
     listen(80);
```

What about REST/HTTP?

Heavyweight HTTP

Easy React Integration

1. Async Rendering
2. Fast Model Diffing

Async Component Render

```
render: function() {
  var jsx,
      jsxReceived = false;
  var model = new falcor.Model(
    new falcor.DataSource("member.json"));
  model.get(
    ["name"],
    ["location", "city"],
    ["location", "address"],
    (name, city, address) => {
      jsx = <p>{name} lives at
        <span>{address}, {city}</span></p>;
      if (!jsxReceived) this.forceUpdate();
    }).
    toPromise();
  if (jsx) {
    jsxReceived = true;
    return jsx;
  }
  else {
    return <img src="progress.gif" />;
  }
}
```

Steve McGuire lives at 344 Seaside, Pacifica

Fast Model Diffing

```
shouldComponentUpdate: function(nextProps, nextState, nextContext) {
  return (this.state.generation !== nextState.generation) ||
    (this.state.key !== nextState.key);
}
```

New, more convenient API.

One Model *Everywhere*

JSON

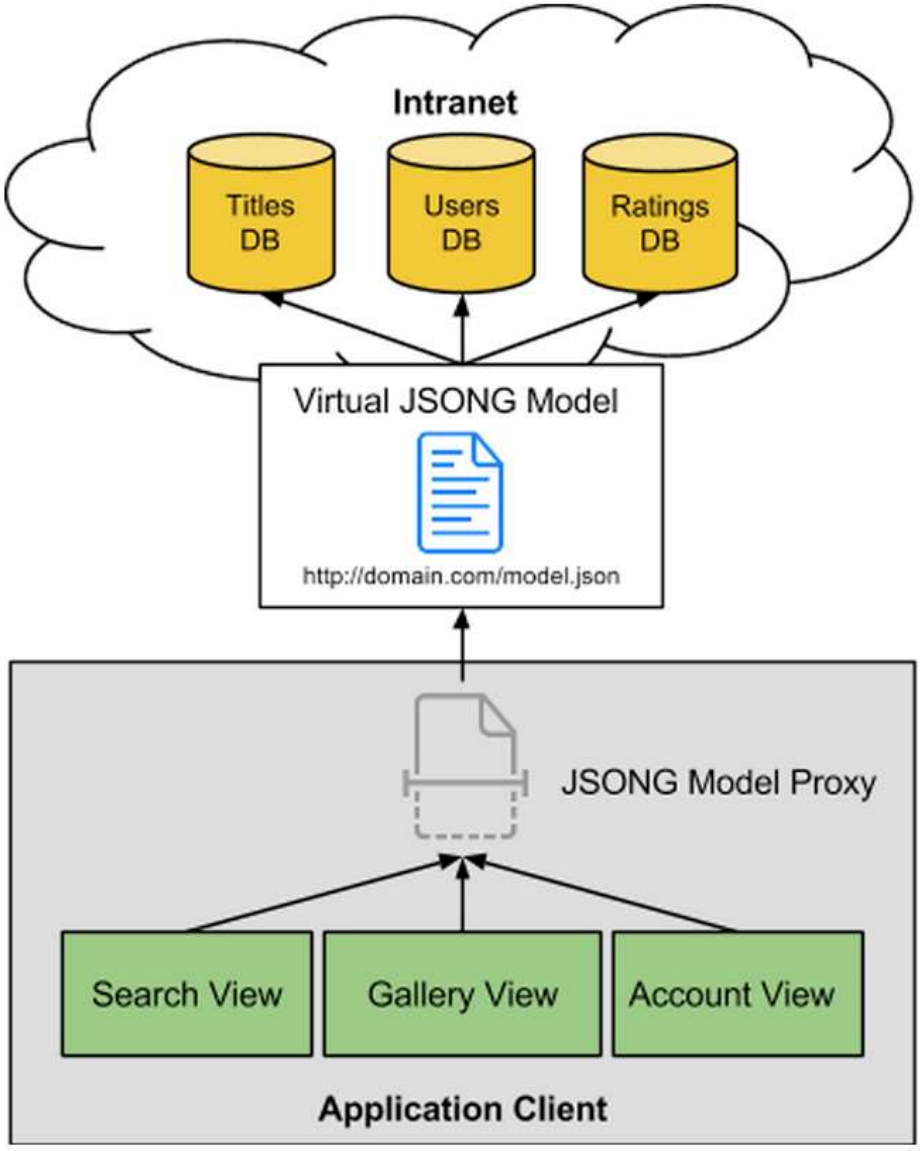
```
var model = {
  genreLists: [
    {
      name: "Suggestions For You"
      titlesList: [
        {
          id: 956,
          "Friends",
          rating: 5
        }
      ]
    },
    {
      name: "New Releases",
      titlesList: [
        {
          id: 956,
          "Friends",
          rating: 3
        }
      ]
    }

  ]
}
```

```
model["genreLists"][0]["titlesList"][0]["name"] = 5
```

JSON Graph

```
var model = {
  genreLists: {
    {
      name: "Suggestions For You"
      titlesList: {
        "0": ["titlesById", 956],
        length: 75




      }
    },
    {
      name: "New Releases",
      titlesList: {
        "0": ["titlesById", 956],
        length: 75




      }
    },
    length: 50
  },
  titlesById: {
    "956": {
      name: "Friends",
      rating: 3
    }
  }
}
```

```
{
  id: 956,
  name: "Friends",
  rating: 3
}
```


Intranet
Titles DB
Users DB
Ratings DB
Virtual JSONG Model
http://domain.com/model.json
JSONG Model Proxy
Search View
Gallery View
Account View
Application Client

In the last 10 years, the web has transformed.

In the last 10 years, the web has transformed.

Issue Tracking: REST End Points

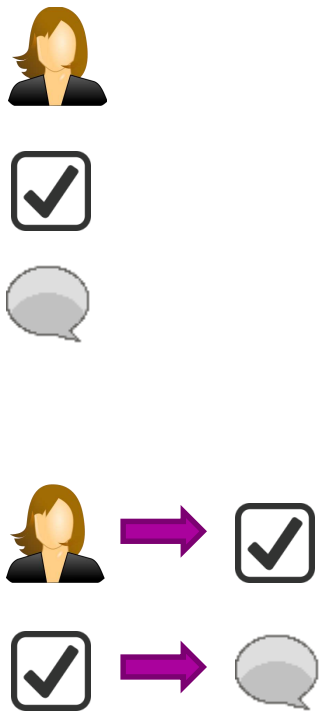
/user/{userid}

/issue/{issueid}

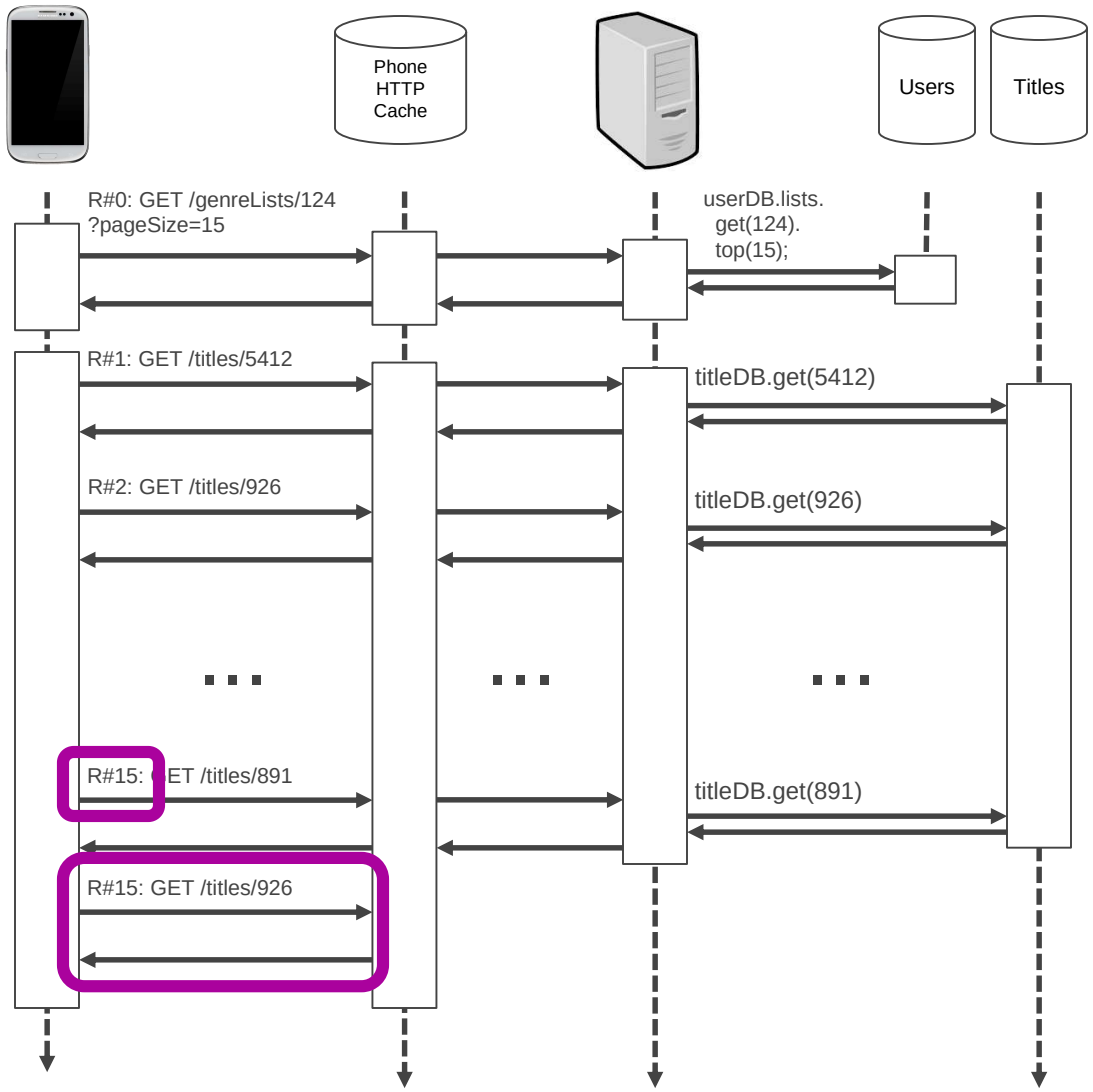
/comment/{commentid}

/user/{userid}/issues

/issue/{issueid}/comments

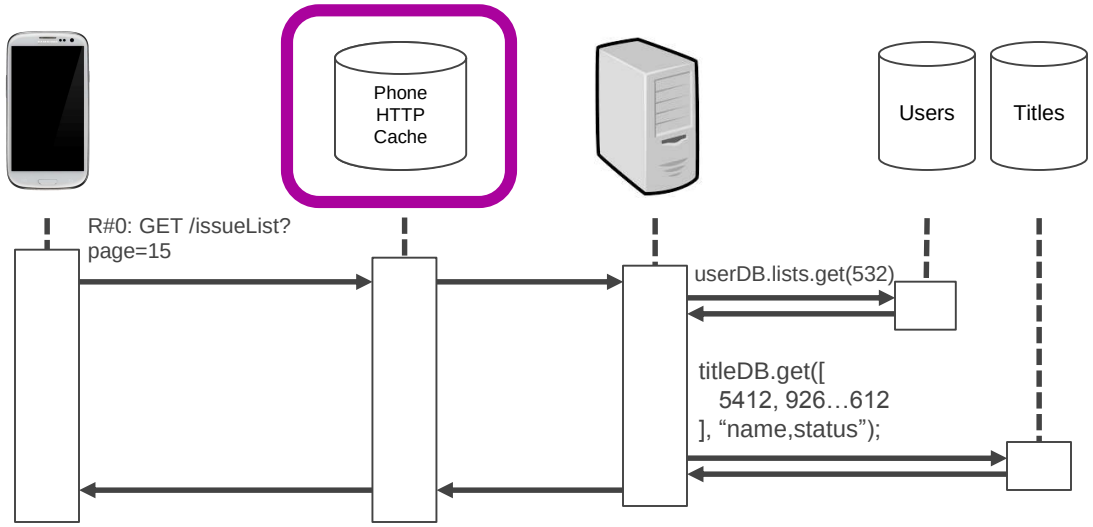


Why not REST?


Phone
HTTP
Cache
Users
Titles
R#0: GET /genreLists/124
?pageSize=15
userDB.lists.
get(124).
top(15);
R#1: GET /titles/5412
titleDB.get(5412)
R#2: GET /titles/926
titleDB.get(926)
R#15: GET /titles/891
titleDB.get(891)
R#15: GET /titles/926


The Hunger Games: Catching Fire
+ MY LIST
More Like: The Hunger Games: Catching Fire

Netflix with RPC



HTTP/REST is for Large Resources

The REST interface is designed to be efficient for large-grain hypermedia data transfer…resulting in an interface that is not optimal for other forms of architectural interaction.

-Dr Roy T Fielding

JSON

Falcor Server

```
var member = new falcor.Model({
  source: new falcor.HttpDataSource("/member.json")
});
```

```
var member = new falcor.Model({cache: {
    name: "Steve McGuire",
    occupation: "Developer",
    location: {
        country: "USA",
        city: "Pacifica",
        address: "3344 Seaside"
    }
}});
```

```
member.get(["location"]["address"]).
    then((address) => print(address)).
    toPromise();
```

```
member.get( ["location", "address"]).
    toPromise().
    then(json => response.write(json));
```

Falcor

```
var member = new falcor.Model({cache: {
  name: “Steve McGuire”,
  occupation: “Developer”,
  location: {
    country: “US”,
    city: “Pacifica”,
    address: “344 Seaside”
  }
}});

member.get([“name”], [“location”, “address”])
  .then(address => print(address))
  .toPromise();
```

```
new falcor.HttpSource(“/member.json”)
```

```
(new falcor.HttpServer(member)).
  listen(80);
```

Virtual Falcor Server

```
var member = new falcor.Model({cache:{
  name: “Steve McGuire”,
  occupation: “Developer”,
  location: {
    country: “US”,
    city: “Pacifica”,
    address: “344 Seaside”
  }
}});

(new falcor.HttpServer(member)).listen(80);
```

```
var member = new falcor.Model({
  router: new falcor.Router([
    {
      route: [“member”,[“name”,”occupation”]],
      get: (pathSet) =>
        memberDB.
          exec(`SELECT ${pathSet[1].join(‘,’)}
                FROM user
                WHERE id = ${request.cookies.userid}`)
    },
    {
      route: [“member”,“location”,[“country”, ”city”, “address”]],
      get: (pathSet) =>
        locationServer.
          getLocation(request.cookies.userid).
          then(location => ({
            member: { location: getProps(location, pathSet[2]) }
          })
    }
  ])
});

(new falcor.HttpServer(member)).listen(80);
```

Falcor Client

```
var member = new falcor.Model({source:
  new falcor.HttpSource("member.json")});
```

```
member.
  get(["location",["address","city"]]).
  toPromise();
```

Falcor Server

```
var member = new falcor.Model({
    name: "Steve McGuire",
    occupation: "Developer",
    location: {
        country: "US",
        city: "Pacifica",
        address: "344 Seaside"
    }
});

(new falcor.HttpServer(member)).
    listen(80);
```

Falcor Client

```
var member = new falcor.Model({source:
  new falcor.HttpSource("member.json")});
```

```
member.
  get(["location",["address","city"]],
     (address) => <div>{address}></div>).
<div>{address}></div>)
  subscribe(jsx => updateUI(jsx));
```

Falcor Server

```
var member = new falcor.Model({
     name: "Steve McGuire",
     occupation: "Developer",
     location: {
          country: "US",
          city: "Pacifica",
          address: "344 Seaside"
     }
}});

(new falcor.HttpServer(member)).
     listen(80);
```

Virtual Model Path Evaluation

```
var member = new falcor.Model({
  source: new falcor.HttpSource('member.json')});
  cache: {
    genreLists: {
      "0": {
        "0": ["titlesById", 926]
      }
    },
    titlesById: {
      "926": {
        name: "Friends"

      }
    }
  });
```

```
var member = new falcor.Model({
  router: new falcor.Router([
    {
      route: ["genreLists", {from:0, to:1}, {from:0, to:1}],
      get: () => { /* retrieve data from lists DB */},
    },
    {
      route: ["titlesById", Router.INTEGERS, ["name", "rating"]],
      get: (pathSet) => { /* retrieve data from titles DB */}
    }
  ]);
})cache: {
    genreLists: {
      "0": {
        "0": ["titlesById", 926]
      }
    },
  } titlesById: {
});   "926": {
        name: "Friends"
      }
    }
  }
});
```

```
member.
  getValue([                , "name"]).
  toPromise();
```

```
member.get(["titlesById", 926, "name"])
```