DAVID TANG

EMBER DATA AND CUSTOM APIS

verizon

Digital Media Services

USC

Viterbi

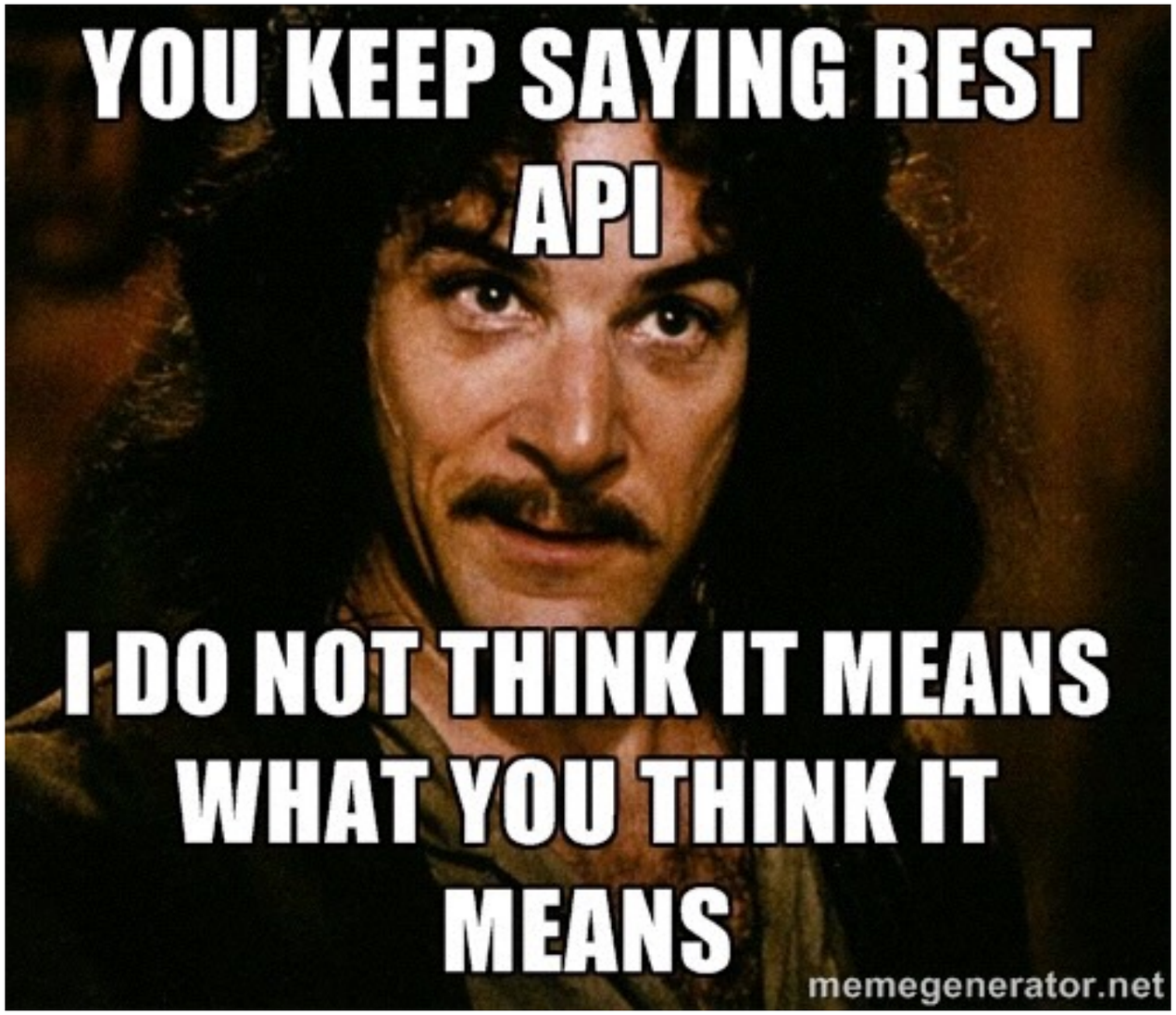
School of Engineering

Information

Technology Program

OVERVIEW

- Adapters vs Serializers
- Built-in Adapters
- Common Adapter Customizations
- Built-in Serializers
- Common Serializer Customizations
- Testing it all


YOU KEEP SAYING REST API
I DO NOT THINK IT MEANS WHAT YOU THINK IT MEANS
memegenerator.net

STORE

SERIALIZERS

ADAPTERS

TRANSFORMS

¯_(ツ)_/¯

MODELS

IDENTITY MAPPING

REST SERIALIZER

JSON API SERIALIZER

JSON SERIALIZER

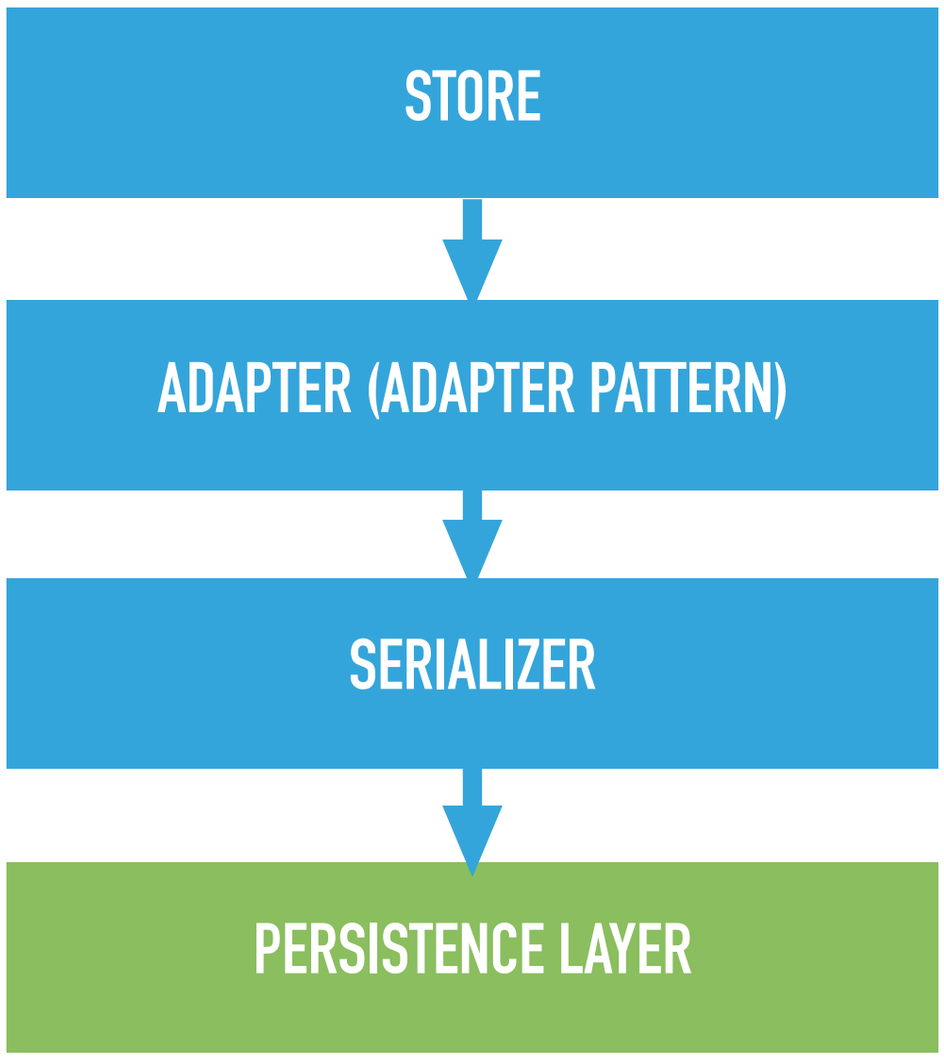
SNAPSHOTS

ACTIVE MODEL ADAPTER

REST ADAPTER


Awesome

EMBER DATA OVERVIEW



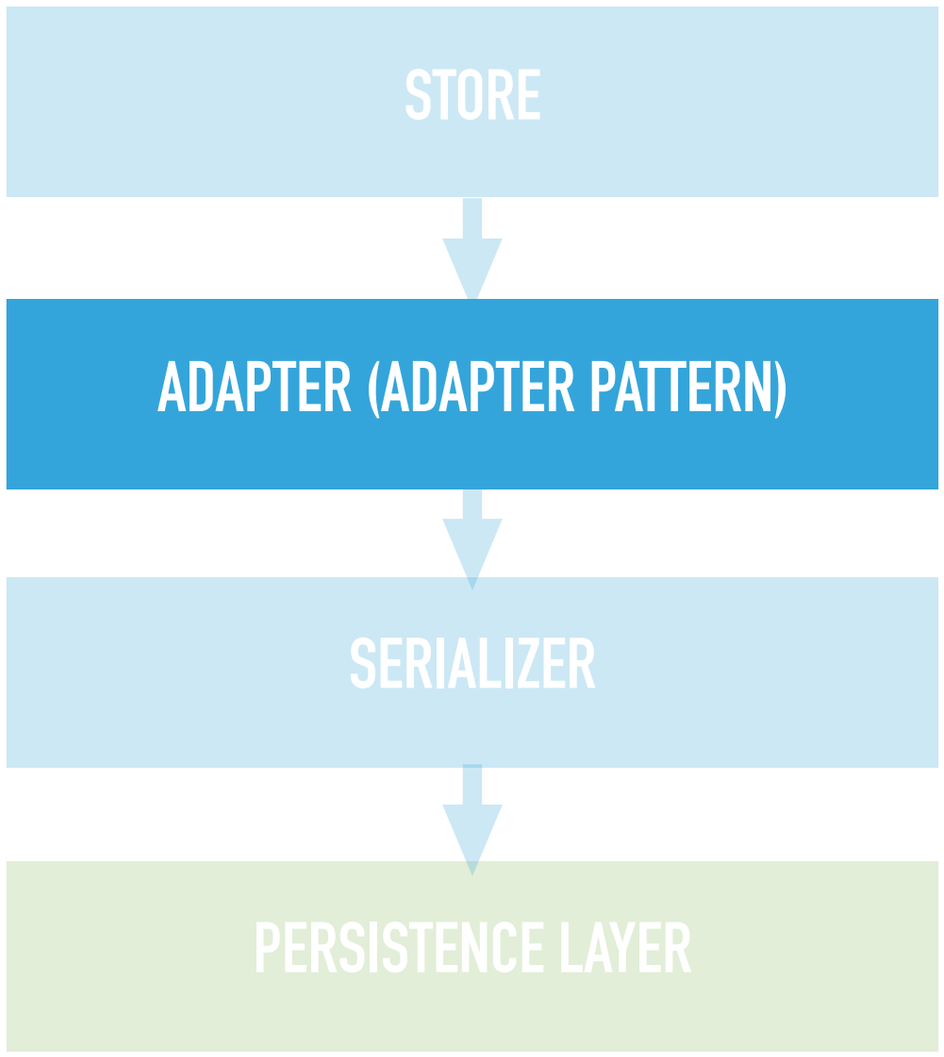
Data store (models)

Communicating with persistence layer

Format data to and from persistence layer

Server, localStorage, Firebase, Parse, etc

ADAPTERS



Data store (models)

Communicating with persistence layer

Format data to and from persistence layer

Server, localStorage, Firebase, Parse, etc

BUILT-IN ADAPTERS

JSON-API

`DS.JSONAPIAdapter`

REST

`DS.RESTAdapter`

Base Adapter

`DS.Adapter`

COMMON ADAPTER CUSTOMIZATIONS

HOST AND NAMESPACE

```
import ENV from 'mycatapp/config/environment';

// app/adapters/application.js
export default DS.RESTAdapter.extend({
  host: ENV.APP.apiEndpoint,
  namespace: 'api/v1'
});
```

PATH FOR TYPE

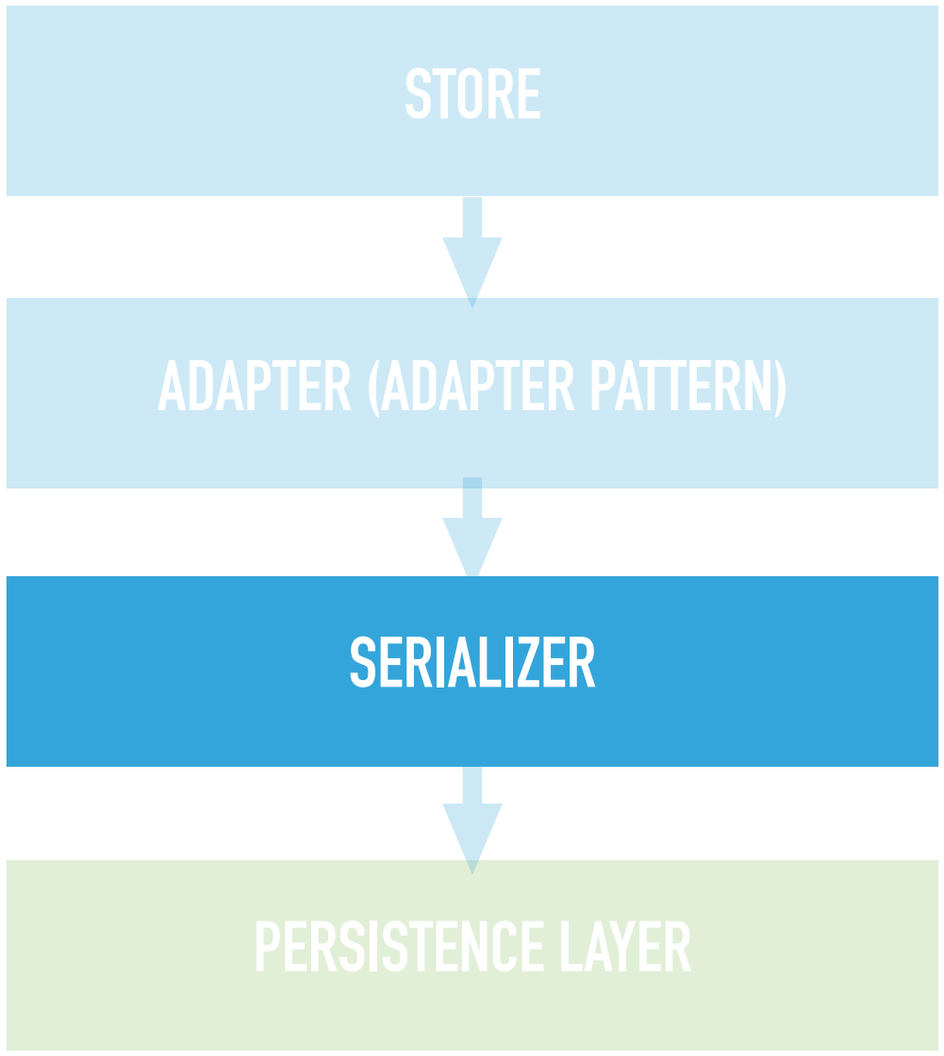
YOUR API	EMBER
`POST /api/user`	`POST /api/users`
`GET /api/video-games`	`GET /api/videoGames`

```
export default ApplicationAdapter.extend({
  pathForType(modelName) {
    return Ember.String.pluralize(modelName);
  }
});
```

CUSTOMIZING THE URL

- urlForCreateRecord
- urlForDeleteRecord
- urlForFindAll
- urlForFindRecord
- urlForQuery
- urlForQueryRecord
- urlForUpdateRecord

SERIALIZERS



Data store (models)

Communicating with persistence layer

Format data to and from persistence layer

Server, localStorage, Firebase, Parse, etc

Index | Methods | Properties

Show: ☑ Inherited ☐ Protected ☐ Private ☐ Deprecated

METHODS

extractAttributes
extractErrors
extractId
extractMeta
extractPolymorphicRelationship
extractRelationship
extractRelationships
keyForAttribute
keyForLink
keyForPolymorphicType
keyForRelationship
modelNameFromPayloadKey
normalize
normalizeArrayResponse
normalizeCreateRecordResponse
normalizeDeleteRecordResponse
normalizeFindAllResponse
normalizeFindBelongsToResponse
normalizeFindHasManyResponse
normalizeFindManyResponse
normalizeFindRecordResponse
normalizeQueryRecordResponse
normalizeQueryResponse
normalizeResponse
normalizeSaveResponse
normalizeSingleResponse
normalizeUpdateRecordResponse
payloadKeyFromModelName
pushPayload
serialize
serializeAttribute
serializeBelongsTo
serializeHasMany
serializeIntoHash
serializePolymorphicType

PROPERTIES

attrs
primaryKey
store

BUILT-IN SERIALIZERS

JSON-API

`DS.JSONAPISerializer`

JSON

`DS.JSONSerializer`

REST

`DS.RESTSerializer`

Base Serializer

`DS.Serializer`

JSON SERIALIZER

PAYLOADS AND RESPONSES

```
GET /api/users/1
{ "id": 1, "name": "David Tang" }
```

```
GET /api/users
[
  { "id": 2, "name": "Michael Westen" },
  { "id": 3, "name": "Fiona Glenanne" }
]
```

RELATIONSHIPS

```
{
   "id": 99,
   "name": "David Tang",
   "pets": [ 1, 2, 3 ],
   "company": 7
}
```

REST SERIALIZER

PAYLOADS AND RESPONSES

```
GET /api/users/1
```

```
{
   "user": {
      "id": 1,
      "name": "David Tang",
      "pets": [ 1, 2, 3 ],
      "company": 7
   }
}
```

PAYLOADS AND RESPONSES

```
GET /api/users
```

```
{
   "users": [
      { "id": 1, "name": "Steve McGarret" },
      { "id": 2, "name": "Danny Williams" }
   ]
}
```

RELATIONSHIPS AND SIDELOADING

```
GET /api/users/1
```

```
{
  "user": {
    "id": 1, "name": "David Tang", "company": 7
  },
  "companies": [
    { "id": 7, "name": "Company A" }
  ]
}
```

JSON-API SERIALIZER

http://thejsguy.com/2015/12/05/which-ember-data-serializer-should-i-use.html

COMMON SERIALIZER CUSTOMIZATIONS

Tip: Extend a serializer that matches your API as close as possible

NORMALIZING RESPONSES

```
GET /api/cats
```

```
{
  "data": [
    { "id": 1, "name": "Tubby" },
    { "id": 2, "name": "Frisky" },
    { "id": 3, "name": "Tabitha" }
  ]
}
```

NORMALIZING RESPONSES

```
// app/serializers/cat.js
export default DS.RESTSerializer.extend({
 normalizeResponse(a, b, payload, c, d) {
    payload = { cats: payload.data };
    return this._super(a, b, payload, c, d);
 }
});
```

NORMALIZING RESPONSES

```
// app/serializers/cat.js
export default DS.JSONSerializer.extend({
 normalizeResponse(a, b, payload, c, d) {
    payload = payload.data;
    return this._super(a, b, payload, c, d);
 }
});
```

NORMALIZING BY STORE CALL

- normalizeCreateRecordResponse
- normalizeDeleteRecordResponse
- normalizeFindAllResponse
- normalizeFindHasManyResponse
- normalizeFindRecordResponse
- normalizeQueryRecordResponse
- normalizeQueryResponse

attrs

```
{ "id": 1, "first_name": "Tubby", "years": 4 }
```

```
// app/serializers/cat.js
export default DS.RESTSerializer.extend({
  attrs: {
    firstName: 'first_name',
    age: 'years'
  }
});
```

RELATIONSHIP ATTRIBUTES

```
{ "id": 1, "home_id": 3, "owner_id": 2 }
```

```
// app/serializers/application.js
export default DS.RESTSerializer.extend({
  keyForRelationship(key, relationship) {
    if (relationship === 'belongsTo') {
      return `${key}_id`;
    }
  }
});
```

EMBEDDED RECORDS

```
{
  "id": 5,
  "name": "David Tang",
  "skills": [
    { "id": 2, "name": "JavaScript" },
    { "id": 9, "name": "Ember" }
  ]
}
```

EMBEDDED RECORDS

```
// app/serializers/user.js
let Mixin = DS.EmbeddedRecordsMixin;

export default DS.JSONSerializer.extend(Mixin, {
  attrs: {
    skills: { embedded: 'always' }
  }
});
```

SETTING THE PRIMARY KEY

```
// app/serializers/user.js
export default DS.RESTSerializer.extend({
  primaryKey: 'socialSecurityNumber'
});
```

SERIALIZING DATA ON SAVE

```
// app/serializers/user.js
export default DS.RESTSerializer.extend({
  serialize(snapshot) {
    return [
      { name: snapshot.attr('name') }
    ];
  }
});
```

DS.SNAPSHOT

```
// snapshot for a user model

snapshot.id;

snapshot.attr('name')

snapshot.hasMany('interests') // interestsSnapshot

snapshot.belongsTo('company') // companySnapshot
```

TESTING IT ALL

THE DEFAULT SERIALIZER TEST

```
import { moduleForModel, test } from 'ember-qunit';
moduleForModel('cat', 'Unit | Serializer | cat', {
  needs: ['serializer:cat']
});

test('it serializes records', function(assert) {
  var record = this.subject();
  var serializedRecord = record.serialize();
  assert.ok(serializedRecord);
});
```

1. moduleFor() INSTEAD OF moduleForModel()

```
import { moduleFor, test } from 'ember-qunit';


moduleFor(
  'serializer:cat', 'Unit | Serializer | cat', {}
);


test('it serializes records', function(assert) {
  let serializer = this.subject();
});
```

2. USE THE STORE

Test serializers and adapters through the store with an HTTP mocking library

TESTING WITH THE STORE – 1. SETUP

```
moduleForModel('cat', 'Unit | Serializer | cat', {
  needs: [ 'serializer:cat', 'adapter:cat' ],
  beforeEach() {
    // next slide
  },
  afterEach() {
    // next slide
  }
});
```

TESTING WITH THE STORE – 2. PRETENDER

```
// beforeEach()
this.server = new Pretender(function() {
  this.get('/api/cats', function() {
    let response = JSON.stringify(/* data */);
    return [ 200, {}, response ];
  });
});

this.server.shutdown(); // afterEach()
```

TESTING WITH THE STORE – 3. THE TEST

```
test('array responses', function(assert) {
  let store = this.store();
  return store.findAll('cat').then((cats) => {
    assert.equal(cats.get('length'), 3);
  });
});
```

TESTING THE DATA YOUR SERVER RECEIVES

```
// app/serializers/cat.js
export default DS.RESTSerializer.extend({
  serialize(snapshot) {
    /* implementation */
  }
});
```

TESTING THE DATA YOUR SERVER RECEIVES

```
let [ request ] = this.server.handledRequests;
let body = request.requestBody;
let requestPayload = JSON.parse(body);
let expectedJSON = /* JSON */;
assert.deepEqual(requestPayload, expectedJSON);
```

EMBER DATA RESOURCES

- Introduction to Ember Data 2.0 by Christoffer Persson
- My Blog - thejsguy.com
- Ember Data and Custom APIs - 5 Common Serializer Customizations
- Which Ember Data Serializer Should I Use?
- Working with Nested Data in Ember Data Models
- Handling Errors with Ember Data

THANKS!

THEJSGUY.COM

@SKATERDAV85