Ember 2

Section 9 - Models

Defining Models
Finding Records in the Ember Data Store
Creating, Updating, Deleting
Ember Data Relationships
Pushing Records Into The Store

Section Objectives

- Be able to define Ember models
- Be able to query models using Ember Data helper methods
- Be able to apply CRUD operations to Ember Data
- Be able to define Ember Data relationships

Models

Defining Models

Defining Models
Finding Records in the Ember Data Store
Creating, Updating, Deleting
Ember Data Relationships
Pushing Records Into The Store

Defining Models

- Ember Models are subclasses of DS.Model

  http://emberjs.com/api/data/classes/DS.Model.html

- Ember Models are definitions only and don't store any data

- To access data for your application, you should always go through the Ember Store (DS.Store). This is automatically created for you.

```
let notes = this.get('store').peekAll('note');
```

- Define an adapter to make /GET requests

Lab 10 - Adapters

- Create an application adapter using the Mirage add-on to make /GET requests to retrieve and update model data.
  1. Install the Mirage add-on (HTTP stubbing library)
     ember install ember-cli-mirage
  2. Create an application JSONAPIAdapter
  3. Specify a namespace in both the Mirage config and application adapter
  4. Refactor all peekXYZ() to findXYZ()

Defining Modes

- Attributes

```
export default DS.Model.extend({
    firstName: DS.attr()
});
```

- Transforms – string, number, boolean, date

```
export default DS.Model.extend({
    firstName: DS.attr('string')
});
```

- Options

```
export default DS.Model.extend({
    firstName: DS.attr('string', { defaultValue: 'First name' })
});
```

- Relationships

```
export default DS.Model.extend({
    folder: DS.hasMany('note'),
    note  : DS.belongsTo('folder')
});
```

Models

Finding Records in the Ember Data Store

Defining Models
Finding Records in the Ember Data Store
Creating, Updating, Deleting
Ember Data Relationships
Pushing Records Into The Store

Finding Records

- Retrieving a Single Record

```
let notes = this.get('store').findRecord('notes', 1); //GET /notes/1
```

```
let notes = this.get('store').peekRecord('notes', 1); //no request
```

- Retrieving Multiple Records

```
let notes = this.get('store').findAll('notes'); //GET /notes/
```

```
let notes = this.get('store').peekAll('notes'); //no request
```

- When using the findXYZ(), the result is a DS.PromiseArray that fulfills to a DS.RecordArray

- When using the peekXYZ(), the result is a DS.RecordArray

Models

Creating, Updating, Deleting

Defining Models
Finding Records in the Ember Data Store
Creating, Updating, Deleting
Ember Data Relationships
Pushing Records Into The Store

Creating, Updating

- Creating

```
store.createRecord('note', {
    title: 'New Note',
    body: 'Lorem ipsum'
});
```

- Updating

```
this.get('store').findRecord('note', 1).then(function(note) {
    note.set('title', "My New Note");
});
```

- If you are using the peekXYZ(), then you would create or update directly on the store using the methods available on the DS.RecordArray

Deleting

- Deleting

```
store.findRecord('note', 1).then(
    function(post) {
        post.deleteRecord();
        post.get('isDeleted'); // => true
        post.save(); // => DELETE to /posts/1
    }
); // OR

store.findRecord('note', 2).then(
    function(post) {
        post.destroyRecord(); // => DELETE to /posts/2
    }
);
```

- If you are using the peekXYZ(), then you would remove directly from the store using ***unloadRecord(<record>)***

Models

Ember Data Relationships

Defining Models
Finding Records in the Ember Data Store
Creating, Updating, Deleting
Ember Data Relationships
Pushing Records Into The Store

Relationships

- One-to-One

note.js

```
import DS from 'ember-data';

export default DS.Model.extend({
    folder: DS.belongsTo('folder')
});
```

- One-to-Many

folder.js

```
import DS from 'ember-data';

export default DS.Model.extend({
    notes: DS.hasMany('note')
});
```

Models

Pushing Records Into The Store

Defining Models
Finding Records in the Ember Data Store
Creating, Updating, Deleting
Ember Data Relationships
Pushing Records Into The Store

Pushing Records

notes-folder.js

```
export default Ember.Controller.extend({
    actions: {
        addNote() {
            let notes = this.get('store').peekAll('note');

            let note = this.get('store').push({
                data: [{
                    id: notes.content.length,
                    type: 'note',
                    attributes: {
                        title: "New Note",
                        date: new Date(),
                        content: "",
                        selected: false,
                        folder: this.get('model.folder')
                    }
                }]
            });

            this.send('filterByFolder', this.get('model.folder'));
        }
    }
});
```

Section Review

- Be able to define Ember models
- Be able to query models using Ember Data helper methods
- Be able to apply CRUD operations to Ember Data
- Be able to define Ember Data relationships