Intro to Ember.js – Day four cheatsheet

Ember Data

Ember Data guides

Ember Data API docs

JSON API

JSON API website

By default, Ember Data uses the JSON API spec.

Example JSON API document

```
{
    "data": {
        "type": "message",
        "id": "1",
        "attributes": {
            "subject": "hello world",
            "body": "my name is erik"
        },
        "relationships": {
            "to": {
                "data": {
                    "type": "user",
                    "id": "1"
                }
            },
            "from": {
                "data": {
                    "type": "user",
                    "id": "2"
                }
            }
        }
    },
    "included": [{
        "type": "user",
        "id": "1",
        "attributes": {
            "name": "Erik"
        }
    }, {
        "type": "user",
        "id": "2",
        "attributes": {
            "name": "Dan"
        }
    }]
}
```

Architecture

Store: the primary public interface to Ember Data. You use it to create, retreive, update, and delete records. It contains a client-side cache of your model data.

Store API docs

Adapter: determines how data is retreived and persisted from your backend. The store delegates to the adapter to handle all requests to your backend.

Customizing Adapter guide

Serializer: formats the data sent to and received from the backend store. By default, Ember Data serializes data using the JSON API format.

Customizing Serializer guide

Transform: used to serialize and deserialize model attributes when they are saved or loaded from an adapter.

Transform API docs

Generating a model

```
ember generate model <name>
```

Generating an adapter

```
ember generate adapter <name>
```

Setting an API URL prefix

Generate the `application` adapter, which provides configuration for all models in the application:

```
ember generate adapter application
```

Inside your `app/adapters/application.js` :

```
export default DS.JSONAPIAdapter.extend({
  namespace: 'api/v2'
});
```

Frequently used Ember Data Store methods

All of these methods return Promises

`findAll(<type>)`

`findRecord(<type>, <id>)`

`query(<type>, <query>)`

`queryRecord(<type>, <query>)`

`createRecord(<type>, <hash>)`

`deleteRecord(<record>)`

Full Store API docs

Defining models

`DS.attr()` is used to define the schema of a model. For example:

```
export default DS.Model.extend({
  name: DS.attr('string'),
  age: DS.attr('number')
})
```

Relationships between models can be defined using `DS.belongsTo()` and `DS.hasMany()`:

```
export default DS.Model.extend({
  name: DS.attr('string'),
  age: DS.attr('number'),

  group: DS.belongsTo('group'),
  addresses: DS.hasMany('address')
})
```

Creating a new record

```
let newPerson = this.store.createRecord('person', {
  name: "Erik",
  age: 32
});

newPerson.save();
```

Frequently used Ember Data Model APIs

Methods:

All of these methods return Promises

`save()`

`reload()`

`deleteRecord()`

`destroyRecord()`

Properties:

Accessed using `this.get(<property name>)`

`id`

`isNew`

`isLoading`

`isReloading`

`isSaving`

`isDeleted`

`hasDirtyAttributes`

`errors`

Full Model API docs