17-Short-lnk-app video

Method Validation

- now we are going to validate the url inside of links.js Links.insert
- so we are going to be doing the same thing as in users.js try and catch but with a few minor changes
- here's what links.js should look like now

```
import { Mongo } from 'meteor/mongo';

import { Meteor } from 'meteor/meteor';

import SimpleSchema from 'simpl-schema';

export const Links = new Mongo.Collection('links');


if (Meteor.isServer){

    Meteor.publish('links', function(){

        return Links.find({userId : this.userId});

    })

}

Meteor.methods({

    'links.insert'(url){

        if(!this.userId){

            throw new Meteor.Error('not-authorized')

        }


        try{

            new SimpleSchema({

              url:{

                type: String,

                regEx: SimpleSchema.RegEx.Url

              }

            }).validate({ url });

          }catch (e) {

            throw new Meteor.Error(400, e.message )

          }


        Links.insert({

            url,
            userId: this.userId


        });
    }
});
```

- we are going to be checking the error that simple schema throws
- we notice in chrome tools that is comes from the reason property
- and this gets via the label property - notice that we can only change the word Url
- so over in links.js

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';
export const Links = new Mongo.Collection('links');


if (Meteor.isServer){
    Meteor.publish('links', function(){
        return Links.find({userId : this.userId});
    })
}
Meteor.methods({
    'links.insert'(url){
        if(!this.userId){
            throw new Meteor.Error('not-authorized')
        }


        try{
            new SimpleSchema({
              url:{
                type: String,
                label:'Your link',
                regEx: SimpleSchema.RegEx.Url
              }
            }).validate({ url });
          }catch (e) {
            throw new Meteor.Error(400, e.message )
          }
        Links.insert({
            url,
            userId: this.userId

        });
    }
});
```

- and this the result when you put an invalid url, over in the console

```
errorClass {isClientSafe: true, error: 400, reason: "Your link must be a valid URL", details: un
defined, message: "Your link must be a valid URL [400]", …}
```

- we are going to be creating a new file that is going to take care of the error throwing , because it is just not that efficient having it in multiple places of our app
- so inside of imports create a new folder and call it startup and a file called simple-schema-configuration.js
- now inside of this file we are going to be using the Customize the Error that is thrown
- you can find more info in the documentation
- so over in the simple-schema-configuration.js lets import Meteor, and also SimpleSchema
- and we are going to be using the below method, and it take a function as an arguement

```
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';

SimpleSchema.defineValidationErrorTransform()
```

- and it take a function as an arguement
- this get executed every time simple schemal throw an error, it allow us to change that error
- so we pass in the error

```
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';

SimpleSchema.defineValidationErrorTransform((error)=>{

});
```

- and we return an new error

```
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';


SimpleSchema.defineValidationErrorTransform((error)=>{
    return new Meteor.Error()
});
```

- and we are going to passin the necessary stuff
- which is 400, and the error we are going to pull of from error, like we did before
- so now if we have to call this from client and server main.js
- client/main.js

```
import { Meteor } from 'meteor/meteor';
import ReactDom from 'react-dom';
import { Tracker } from 'meteor/tracker';


import { routes, onAuthChange } from '../imports/routes/routes';
import '../imports/startup/simple-schema-configuration.js';


Tracker.autorun(()=>{
  const isAuthenticated = !!Meteor.userId();
  onAuthChange(isAuthenticated);
});


Meteor.startup(()=>{
  ReactDom.render(routes,  document.getElementById('app'));
});
```

- and we are going to do the same with server main.js
- and now we can remove those try and catch keywords from
- so now links.js should look like this

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';
export const Links = new Mongo.Collection('links');


if (Meteor.isServer){
    Meteor.publish('links', function(){
        return Links.find({userId : this.userId});
    })
}
Meteor.methods({
    'links.insert'(url){
        if(!this.userId){
            throw new Meteor.Error('not-authorized')
        }
        new SimpleSchema({
            url:{
            type: String,
            label:'Your link',
            regEx: SimpleSchema.RegEx.Url
            }
        }).validate({ url });

        Links.insert({
            url,
            userId: this.userId

        });
    }
});
```

- now lets make the same change to the users.js file

```
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';
import { Accounts } from 'meteor/accounts-base';


  Accounts.validateNewUser((user)=>{
    const email = user.emails[0].address

    new SimpleSchema({
        email:{
          type: String,
          regEx: SimpleSchema.RegEx.Email
        }
      }).validate({ email });


    return true;
  });
```

- the last thing we want to do is to wipe out the collection completely so go to your terminal
- over in mongo

```
meteor:PRIMARY> db.links.remove({})
```

-