16-Short-lnk-app - video

Building "links.create" method

- first delete the methods we created greetUser and addNumber

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

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

if (Meteor.isServer){
    Meteor.publish('links', function(){
        return Links.find({userId : this.userId});
    })
}
Meteor.methods({

});
```

- and also remove the calls from 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';


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

Meteor.startup(()=>{
```

```
  ReactDom.render(routes,  document.getElementById('app'));
});
```

- **we** are going to be removing a package that enables this insecure behavior
- to explore this package lets run Meteor list
- and the package we are looking for is called insecure

```
joses-MacBook-Pro:short-lnk mendoza$ Meteor list
accounts-password      1.4.1  Password support for accounts
blaze-html-templates   1.1.2  Compile HTML templates into reactive UI with Meteor Blaze
ecmascript             0.8.3  Compiler plugin that supports ES2015+ in all .js files
es5-shim               4.6.15  Shims and polyfills to improve ECMAScript 5 support
insecure               1.0.7  (For prototyping only) Allow all database writes from the client
jquery                 1.11.10  Manipulate the DOM using CSS selectors
meteor-base            1.0.4* Packages that every Meteor app needs
mobile-experience      1.0.5  Packages for a great mobile user experience
mongo                  1.2.2  Adaptor for using MongoDB and Minimongo over DDP
reactive-var           1.0.11  Reactive variable
shell-server           0.2.4  Server-side component of the `meteor shell` command.
standard-minifier-css  1.3.5  Standard css minifier used with Meteor apps by default.
standard-minifier-js   1.2.3* Standard javascript minifiers used with Meteor apps by default.
tracker                1.1.3  Dependency tracker to allow reactive callbacks
```

- when we remove the insecure package the app will break, we will no longer be able to use Links.insert from some sort of react call back
- lets go to the terminal to remove the package insecure

```
joses-MacBook-Pro:short-lnk mendoza$ meteor remove insecure
```

- **now** if you want to add a new link it won't work
- we get an access denied
- go to links.js inside Meteor.methods
- now we use a naming convention, you can name it what ever you like, for example for this method, we will be calling links.insert, and links has nothing to do with anything else on that page, links it's not tied to the actual collection name
- so next, we are you using the dot character and is usually not allowed on an objects property name but we can by pass that by wrapping the property name inside quotes

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

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


if (Meteor.isServer){
    Meteor.publish('links', function(){
        return Links.find({userId : this.userId});
    })
}
Meteor.methods({
    'links.insert'(){


    }
});
```

- and now we have our first Meteor method define
- we then want to make sure who ever calls this method is logged in
- **rememeber that this where all of the security takes place**

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


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


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

- this is going to stop the execution, so anything below this wont run

- now lets lets do Links.insert and we want to insert a document

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

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')
        }
        Links.insert({
            url,
            userId: this.userId

        });
    }
});
```

- above we are using url in es6 shorthand
- and now we are ready to do a method call
- go to Links.js
- on the onSubmit method
- and it also expects the url argument

```
import React from 'react';
import { Accounts } from 'meteor/accounts-base'
import { Meteor } from 'meteor/meteor';
import { Links } from '../api/links';
import  LinksList  from './LinksList';

export default class Link extends React.Component{
    onLogout(){
```

```
        Accounts.logout();
    }
    onSubmit(e){
        const url = this.refs.url.value.trim();
        e.preventDefault();

        if(url){
            Meteor.call('links.insert', url);
            // Links.insert({url, userId: Meteor.userId()});
            this.refs.url.value = '';

        }
    }
```

- now go to your browser and insert a new link
- and you can see that it works
- and if you go to your chrome tools, over in Meteor mongo you will be able to see the link you created
- now you can check in the terminal as well
- and there it is the new link we created

```
meteor:PRIMARY> db.links.find()
{ "_id" : "A8v5uuvAvQMyuvGdy", "url" : "this is from jose@jjmendoza.com", "userId" : "dWcqFGiaur
6gwfpej" }
{ "_id" : "z2CDbf5pFPsEQpi9k", "url" : "new link", "userId" : "dWcqFGiaur6gwfpej" }
meteor:PRIMARY>
```

- we are going to be looking more in the ddp tab in the chrome tools
- go ahead and clear everything from that tab
- uncheck subscriptions and collections
- now add a new link in our app
- and this is what shows now in that tab

```
calling method links.insert with mendoza2 {"msg":"method","method":"links.inse ... }
 10:08:03
107 B
got result for method links.insert {"msg":"result","id":"3"}
 10:08:03
25 B
updated {"msg":"updated","methods":["3"]}
```

- the frist is an outbound event that is going from the clien to the server, and that is initiates our Method links insert
- this what it looks like

```
MessageStack Trace

root:{} 5 keys

msg:"method"

method:"links.insert"

params:[] 1 item

id:"3"

randomSeed:"1aaf66b963afe8caea35"
```

- now we are going to take a look at the second line

```
got result for method links.insert "{"msg":"result","id":"3"}"
```

- it is the result of the method call
- last up we have the updated message

```
updated "{"msg":"updated","methods":["3"]}"
```

- it is also coming from the server to the client