11-Short-lnk-app

Overview of Methods, Publiclations, Subscriptions,

- Subscriptions, Publications and Methods are the tools Meteor gives you to secure and lockdown your applications data
  - Publications and Subscriptions allow you to lock down the data that gets sync to the client
  - MiniMongo was clone of MongoDb, if something got added to MongoDb, DDP would sync that up with MiniMongo
  - But when we are working with production applications we only sync a subset of that data
  - Methods are responsible for creating, updating, and removing data a document

Setting up an Insecure System - Part 1

- first lets do a reset by typing

```
joses-MacBook-Pro:short-lnk mendoza$ meteor reset
```

- now run meteor

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

- now in your api folder create file called links.js for our links collection
- now in that file lets import the named export Mongo
- then create a new Collection constructor-it takes one string, the name of your collection, we are going to name our collections 'links' and we are going to need return results so we are going to create a const variable for it called Links, and we are going to export it so the files that import Links can have access to i

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


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

- lets include this Links variable in the client and server/main.js
- first our server/main.js

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


import '../imports/api/users';
import '../imports/api/links';


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

```
});
```

- then our client main.js, and on this one we will be grabbing our named export Links

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

import { routes, onAuthChange } from '../imports/routes/routes';
import { Links } from '../imports/api/links';



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

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

- lets go to the Link.js and we are going to be creating a form that inserts new links

```
import React from 'react';
import { Accounts } from 'meteor/accounts-base'



export default class Link extends React.Component{
    onLogout(){
        Accounts.logout();
    }
    render(){
        return (
            <div>
                <h1>Your Links</h1>
                <button onClick={this.onLogout.bind(this)}>Logout</button>
                <p>Add Link</p>
```

```
                <form onSubmit={this.onSubmit.bind(this)}>
                    <input type="text" ref="url" placeholder="URL"/>
                    <button>Add Link</button>
                </form>
            </div>
        );

    }
}
```

- now lets create that method up above and then we are going to fetch the inputs value by using refs

```
export default class Link extends React.Component{
    onLogout(){
        Accounts.logout();
    }
    onSubmit(e){
        const url = this.refs.url.value.trim();
        e.preventDefault();
    }
    render(){
        return (
            <div>
```

- now lets do an if statement to check if there is a url in the input
- then we want to use the insert method to Links, the insert method takes an object for now we are going to give it a url property using the ES6 shorthand
- lets import Links above

```
import { Accounts } from 'meteor/accounts-base'

import { Links } from '../api/links';

export default class Link extends React.Component{
    onLogout(){
        Accounts.logout();
    }
    onSubmit(e){
```

```
        const url = this.refs.url.value.trim();
        e.preventDefault();

        if(url){
            Links.insert({url})
        }
    }
```

- now we want to clear the input field after the form has been submitted

```
export default class Link extends React.Component{
    onLogout(){
        Accounts.logout();
    }
    onSubmit(e){
        const url = this.refs.url.value.trim();
        e.preventDefault();

        if(url){
            Links.insert({url});
            this.refs.url.value = '';

        }
    }
```

- now over in your browser create a new account
- in the console click on Meteor developer tools to check what's inside that links collection
- and as you can see in the Minimongo db you there is the user you created
- now type Test value here in the input in the browser
- click on links on the left column, and in the right you should be able to see what you just created
- now type something else with spaces and then type testing 123
- and you will the result on the right without spaces because of trim()
- as a challange we needed to use the find() and fetch() of the links and print them to the console in the client/main.js
- here is my solution

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

```
import { routes, onAuthChange } from '../imports/routes/routes';
import { Links } from '../imports/api/links';



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

Tracker.autorun(()=>{
  console.log('Here are the Links ',Links.find().fetch());
});
```

- **here is the** the teachers solution, he just put it in a variable

```
Tracker.autorun(()=>{
  const links = Links.find().fetch()
  console.log('Here are the Links ', links);
});
```

-