19-Short-lnk-app video

Redirecting Shortened Links

- to send the user to a different website or location
- first you have to set the statusCode to 302, then setHeader to Location and then the url you want that user to redirect

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

import '../imports/api/users';
import '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        console.log('this is from my custom middleware');
        res.statusCode = 302
        // Set HTTP Header
        res.setHeader('Location', 'http://www.google.com')
        // end request
        res.end();
    });
});
```

- the above is just a static version of we want to do, what we want to do is something more dynamic
- the first thing we need to do is parse the url
- so what we want to do is just get the (every thing after the slash) localhost:3000/*xxxxx*
- so we are going to be using the req.url to get this done
- if we console log this:
- and we put on the url localhost:3000/123 this is what gets return on the terminal

```
I20171020-16:09:10.262(-7)? /123
```

- **so what** we want to do now is just get rid of the / backslash
- to store this parse Id lets make a new variable _id and then slice(), and here we want to get rid of the first character, and start from the second so since these are index, we know it start from 0 so that means we will be using 1

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

import '../imports/api/users';
import '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        const _id = req.url.slice(1);
```

- **next** we want to grab our named export Links

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

import '../imports/api/users';
import { Links } from '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        const _id = req.url.slice(1);
```

- now we can grab links to query our collection, lets create a variable for that
- and we want to grab the _id

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

import '../imports/api/users';
import { Links } from '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
```

```
        const _id = req.url.slice(1);
        const link = Links.findOne({_id});
```

- **notice** what we use findOne above, and that is because we are either going to get undefined or one document, and that is exaclty what we want
- next up we are going to see if there was a document if there was a match

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

import '../imports/api/users';
import { Links } from '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        const _id = req.url.slice(1);
        const link = Links.findOne({_id});

        if(link){

        }else{

        }
```

- **if nothing happened we are** just going to do next()

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

import '../imports/api/users';
import { Links } from '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        const _id = req.url.slice(1);
        const link = Links.findOne({_id});
```

```
        if(link){


        }else{
            next();
        }
```

- and then we just copied what we had before inside the if statement

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

import '../imports/api/users';
import { Links } from '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        const _id = req.url.slice(1);
        const link = Links.findOne({_id});


        if(link){
            res.statusCode = 302;
            res.setHeader('Location', link.url);
            res.end();
        }else{
            next();
        }
```

- but instead of sending them to google we want to send them where link has
- to test this refresh your browser
- now get the id by going to chrome tool and chosing the Meteor tab
- then go to minimongo and chose links and copy id
- and paste right after localhost:3000/
- and that should take you directly to the link you have saved, in my case it was google.com
- and this is how server main.js should look like now

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

```
import '../imports/api/users';
import { Links } from '../imports/api/links';
import '../imports/startup/simple-schema-configuration.js';

Meteor.startup(() => {
    WebApp.connectHandlers.use((req,res, next)=>{
        const _id = req.url.slice(1);
        const link = Links.findOne({_id});

        if(link){
            res.statusCode = 302;
            res.setHeader('Location', link.url);
            res.end();
        }else{
            next();
        }
    });
});
```

-