React – Routing

In this lab, you will create a routing React application in a Kraken app.

# Objectives

In this lab, you will

* Use an existing Kraken app named, ‘react’ in the /lab folder
  + Or use the /solution-kraken folder
* Add React pages
* Add React Routes
* Run the app

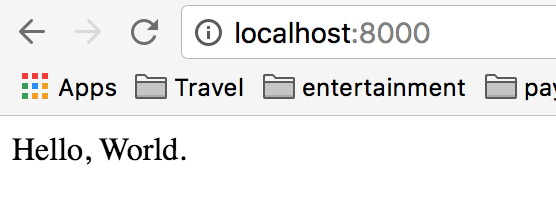
# Run the Kraken app to verify correctness

1. Change to the lab/react folder (or the solution-kraken/react folder) and run

npm install

npm start

1. Open the browser to localhost:8000 and the page should look like this:



1. The output comes from public/views/layout.jsx as shown.

**'use strict'**;  
  
**var *React*** = ***require***(**'react'**);  
  
***module***.exports = ***React***.**createClass**({  
  
 render: **function** *render*() {  
  
 **return** (  
 <**html**>  
 <**head**>  
 <**meta charSet='utf-8'** />  
 <**title**>  
 {**this**.props.**title**}  
 </**title**>  
 </**head**>  
 <**body**>  
 Hello, World.  
 <**script src='/bundle.js'**></**script**>  
 </**body**>  
 </**html**>  
 );  
 }  
});

1. This displays the page content. The component just renders the initial page with Hello World.

# Create the Home Page

1. Create the /public/views/home.jsx page with the following content:

**'use strict'**;  
  
**var *React*** = ***require***(**'react'**);  
**var *Link*** = ***require***(**'react-router/lib/Link'**);  
  
***module***.exports = ***React***.**createClass**({  
  
 **displayName**: **'home'**,  
  
 render: **function** *render*() {  
 **return** (  
 <**div id='home'**>  
 <**h1**>This is the home page</**h1**>  
 <**Link to="/"**>Go to the Home Page</**Link**>  
 <**h6**>I am a React Router rendered view</**h6**>  
 <**Link to='/first'**>Click to go to First</**Link**><**br** />  
 <**Link to='/second'**>Click to go to Second</**Link**><**br** />  
 <**Link to='/third'**>Click to go to Third</**Link**><**br** />  
 <**Link to='/fourth'**>Click to go to Fourth</**Link**><**br** />  
 </**div**>  
 );  
 }  
});

1. This page will link to the other four pages created in the next steps.

# Create an Index Route for the Main Page

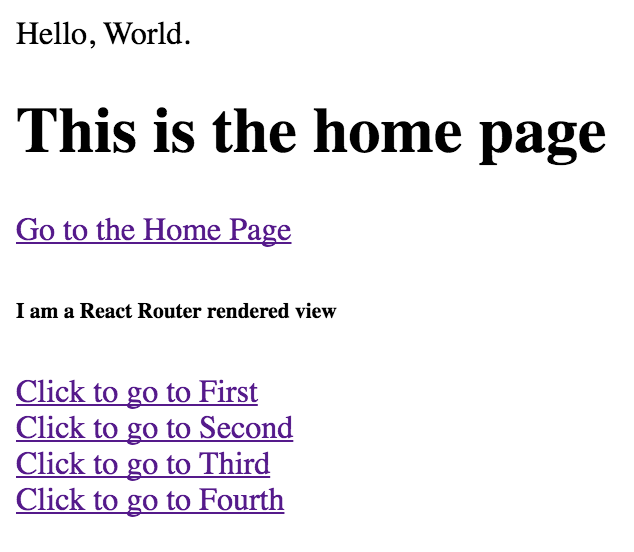
1. Edit the /routes.jsx page, add the above home.jsx page as the <IndexRoute> for the main page as highlighted below:

**var *Layout*** = ***require***(**'./public/views/layout.jsx'**);  
**var *Home*** = ***require***(**'./public/views/home.jsx'**);  
  
**var *routes*** = ***module***.exports = (  
 <**Router**>  
 <**Route path='/' component=**{***Layout***}>  
 <**IndexRoute component=**{***Home***} />  
 </**Route**>  
 </**Router**>  
);

1. Modify the /public/views/layout.jsx page to include its children. This tells React Router to add the additional routes (including the Index Route) to the page when rendering.

</**head**>  
<**body**>  
Hello, World.  
{**this**.props.children}  
<**script src='/bundle.js'**></**script**>  
</**body**>

1. Stop and restart the app. The page should look like the following:



# Create Four Pages

1. Create four pages named, first.jsx, second.jsx, third.jsx and fourth.jsx in the public/views folder. Each component should display its name and include a Link to the next page. It should look similar to this:

**'use strict'**;  
  
**var *React*** = ***require***(**'react'**);  
**var *Link*** = ***require***(**'react-router/lib/Link'**);  
  
***module***.exports = ***React***.**createClass**({  
  
 **displayName**: **'first'**,  
  
 render: **function** *render*() {  
  
 **return** (  
 <**div id='first'**>  
 <**h1**>This is the first page</**h1**>  
 <**Link to="/"**>Go to the Home Page</**Link**>  
 <**h6**>I am a React Router rendered view</**h6**>  
 <**Link to='/second'**>Click to go to Second</**Link**>  
 </**div**>  
 );  
 }  
});

1. Modify the /routes.jsx routed definition to include the four new components inside the main route. Similar to the way we included the home.jsx component above, but using the <Route> tag instead of the <IndexRoute> tag.
2. Stop and start the app again. Click on the links and observe the pages.
3. Observe the output of the Node application. It only loads the page once. All the routing is done in the browser by React. This is a Single Page Application!

# EXTRA CREDIT:

# Add the Header, Footer, Nav, and Middle Components

1. Change the contents of the layout.jsx file to the following:

**'use strict'**;  
  
**var *React*** = ***require***(**'react'**);  
**var *Link*** = ***require***(**'react-router/lib/Link'**);  
  
  
**var *Header*** = ***React***.**createClass**({  
 render() {  
 **return** (  
 <**div className="header"**><**h1**>{**this**.**props**.**header**}</**h1**></**div**>  
 );  
 }  
});  
  
**var *Footer*** = ***React***.**createClass**({  
 render() {  
 **return** (  
 <**div className="footer"**><**h1**>{**this**.**props**.**footer**}</**h1**></**div**>  
 );  
 }  
});  
  
**var *Nav*** = ***React***.**createClass**({  
 render() {  
 **return** (  
 <**div className="nav"**>  
 <**h1**>{**this**.**props**.**nav**}</**h1**>  
 </**div**>  
 );  
 }  
});  
  
**var *Middle*** = ***React***.**createClass**({  
 render() {  
 **return** (  
 <**div className="middle"**>{**this**.**props**.**text**}</**div**>  
 );  
 }  
});  
  
  
***module***.exports = ***React***.**createClass**({  
  
 getInitialState() {  
 **return** {  
 **header**: **'This is the Header'**,  
 **footer**: **'This is the Footer'**,  
 **nav**: **'Nav'** };  
 },  
  
 render: **function** *render*() {  
  
 **return** (  
 <**html**>  
 <**head**>  
 <**meta charSet='utf-8'**/>  
 <**title**>  
 {**this**.props.**title**}  
 </**title**>  
 <**link rel="stylesheet" href="css/style.css"**/>  
 </**head**>  
 <**body**>  
 <**div className="fullPageBox"**>  
 <**Header header=**{**this**.state.**header**}/>  
 <**div className="content"**>  
 <**Nav nav=**{**this**.state.**nav**}/>  
 <**Middle text="Hello, world."**>  
 </**Middle**>  
 </**div**>  
 <**Footer footer=**{**this**.state.**footer**}/>  
 </**div**>  
 <**script src='/bundle.js'**></**script**>  
 </**body**>  
 </**html**>  
 );  
 }  
});

1. Create the file, /public/css/style.css and insert the following:

**body** {  
 **padding**: 50**px**;  
 **font**: 14**px "Lucida Grande"**, **Helvetica**, **Arial**, **sans-serif**;  
}  
  
.**content** {  
 **width**: 100%;  
 **background-color**: **lightblue**;  
 **vertical-align**: **top**;  
}  
  
.**header**, .**footer** {  
 **background-color**: **cornflowerblue**;  
 **padding**: 15**px**;  
}  
  
.**middle**, .**nav** {  
 **vertical-align**: **top**;  
 **display**: **inline-block**;  
 **padding**: 20**px**;  
}  
  
.**nav** {  
 **border-right**: 1**px solid black**;  
 **height**: 260**px**;  
}

1. Restart the server. You should see the following:



1. Now, modify the <Nav> portion of layout.jsx to include <Links>s to the Components like we did before.

**var *Nav*** = ***React***.**createClass**({  
 render() {  
 **return** (  
 <**div className="nav"**>  
 <**h1**>Nav</**h1**>  
 <**Link to='/first'**>First</**Link**><**br** />  
 <**Link to='/second'**>Second</**Link**><**br** />  
 <**Link to='/third'**>Third</**Link**><**br** />  
 <**Link to='/fourth'**>Fourth</**Link**><**br** />  
 </**div**>  
 );  
 }  
});

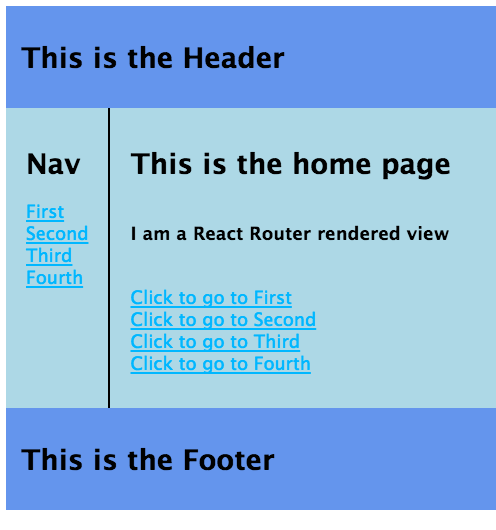
1. Change the <Middle> portion of layout.jsx to include the children:

**var *Middle*** = ***React***.**createClass**({  
 render() {  
 **return** (  
 <**div className="middle"**>{**this**.**props**.**children**}</**div**>  
 );  
 }  
});

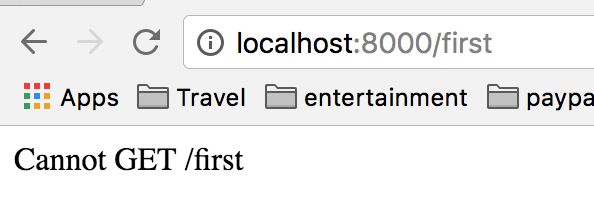
1. Finally, change the main render method of layout.jsx to include the children inside the <Middle> component:

<**div className="fullPageBox"**>  
 <**Header header=**{**this**.state.**header**}/>  
 <**div className="content"**>  
 <**Nav nav=**{**this**.state.**nav**}/>  
 <**Middle text="Hello, world."**>  
 {**this**.props.**children**}  
 </**Middle**>  
 </**div**>  
 <**Footer footer=**{**this**.state.**footer**}/>

1. Restart the server and refresh the browser. You should see:



1. Click on the links and watch the page change.
2. Click on the First link and show the first page:
3. Refresh the browser and notice the following error:



1. Wow. The server does not know how to do Server Side Rendering for the path, /first. Let’s edit the ./controllers/index.js file to add ALL paths that start with /. Notice the wild card in the path for the router.get() function.

router.get(**'/\*'**, **function** (req, res) {  
  
 res.render( req.**url**, model );  
  
});

1. The above tells Kraken to render the PATH from the request. Kraken looks in the routes.jsx file to determine HOW to render the path, then it renders the layouts.jsx with the first.jsx file as the current child.
2. To create your React app, you usually only need to update the views and the routes.jsx files.

Congratulations. You have completed this lab.