**Anonymous REST API Call from SPFx React WebPart with properties from web part**

## Solution: ThirdPartyAPICall-2

In this article we will see the steps to create a SPFx web part which calls a third-party anonymous REST API to get data from the API. Some properties like URL and item id will be passed from web part properties to change the data in the web part.

**Step 1:** Open you React SPFx web part and update the **interface** **IThirdpartyApiCallProps**.**ts** inside your **components** folder. Update the interface with following 3 properties and import **WebPartContext** to create property for web part context inside react component.

import { WebPartContext } from '@microsoft/sp-webpart-base';

export interface IThirdpartyApiCallProps {

  apiURL: string;

  userID: string;

  context: WebPartContext;

}

**Step 2:** Open web part file **ThirdpartyApiCallWebPart.ts** and add below 2 properties in interface **IThirdpartyApiCallWebPartProps**. These properties will be passed to component from the web part.

export interface IThirdpartyApiCallWebPartProps {

  apiURL: string;

  userID: string;

}

**Step 3:** Update **render()** method to set the properties of react component. **Context** property of react component is set from web part context. Other **2 properties** are set from web part properties.

const element: React.ReactElement<IThirdpartyApiCallProps> = React.createElement(

      ThirdpartyApiCall,

      {

        context: this.context,

        apiURL: this.properties.apiURL,

        userID: this.properties.userID

      }

    );

**Step 4:** Define 2 new text box properties for property pane of web part in **getPropertyPaneConfiguration()** method. These 2 properties will be passed to react component properties from web part.

groupFields: [

                PropertyPaneTextField('apiURL', {

                  label: "New API URL"

                }),

                PropertyPaneTextField('userID', {

                  label: "User ID"

                })

              ]

**Step 5:** Define default value for the 2 new properties in web manifest json file. **ThirdpartyApiCallWebPart.manifest.json**

 "properties": {

      "description": "Third Party API call",

      "apiURL": "https://jsonplaceholder.typicode.com/users",

      "userID": "1"

    }

**Step 6:** Create a new .ts file in components folder with name **IThirdpartyApiCallState**. This interface will have all the values which we fetch from single API record <https://jsonplaceholder.typicode.com/users/1>. All the properties of API record are stored in react component state which is define in this interface. We will define all those properties which we want to fetch from API.

export interface IThirdpartyApiCallState {

    id:string;

    name:string;

    username:string;

    email:string;

    address:string;

    phone:string;

    website: string;

    company: string;

  }

**Step 6:** Now open react component .tsx file **ThirdpartyApiCall.tsx** and do following:

1. Access state interface **IThirdpartyApiCallState** in tsx file as state management. Import the interface and add as state in **React.Component** class constructor.

import { IThirdpartyApiCallState } from './IThirdpartyApiCallState';

export default class ThirdpartyApiCall extends React.Component<IThirdpartyApiCallProps, IThirdpartyApiCallState>

2. Add construct for react component class **ThirdpartyApiCall** in tsx file. This constructor will be communicating with properties and state through interfaces. Here were initialing all our state properties to null.

  public constructor(props: IThirdpartyApiCallProps, state: IThirdpartyApiCallState) {

    super(props);

    this.state={

      id:null,

      name:null,

      username:null,

      email:null,

      address:null,

      phone:null,

      website: null,

      company: null

    };

}

3. Update **render()** method code to show values of properties received from API.

          <span className={ styles.welcome }>User Details:</span>

          <div><strong>ID: </strong>{this.state.id}</div><br/>

          <div><strong>User Name: </strong>{this.state.username}</div><br/>

          <div><strong>Name: </strong>{this.state.name}</div><br/>

          <div><strong>Address: </strong>{this.state.address}</div><br/>

          <div><strong>Email: </strong>{this.state.email}</div><br/>

          <div><strong>Phone: </strong>{this.state.phone}</div><br/>

          <div><strong>Web site: </strong>{this.state.website}</div><br/>

          <div><strong>Company: </strong>{this.state.company}</div><br/>

4. Add **componentDidMount()** method. This method is called after **constructor** (with null values) 🡺 **render**(with printing null values) 🡺 **componentDidMount** (Invoke API and Set Data Into State). Inside this method we are calling **InvokeAPIAndSetDataIntoState.** This is getting user details from API using **getUserDetails()** method and adding to **state of react component** using **setState()** method. **setState()** again calls render method internally and print values received from API. Import **HttpClient** and **HttpClientResponse**.

import { HttpClient, HttpClientResponse } from '@microsoft/sp-http';

  public componentDidMount() {

    this.InvokeAPIAndSetDataIntoState();

  }

  public InvokeAPIAndSetDataIntoState(){

    this.getUserDetails().then(response =>{

      this.setState({

        id: response.id,

        name: response.name,

        username: response.username,

        email:response.email,

        address:'Street: '+ response.address.street+ ' Suite: '+ response.address.suite+ ' City'+response.address.city+' Zip Code:'+response.address.zipcode,

        phone:response.phone,

        website: response.website,

        company: response.company.name

      });

     });

  }

public getUserDetails(): Promise<any> {

    let url = this.props.apiURL + "/" + this.props.userID;

    return this.props.context.httpClient.get(

      url, HttpClient.configurations.v1

    )

      .then((response: HttpClientResponse) => {

        return response.json();

      })

      .then(jsonResponse => {

        return jsonResponse;

      }) as Promise<any>;

  }

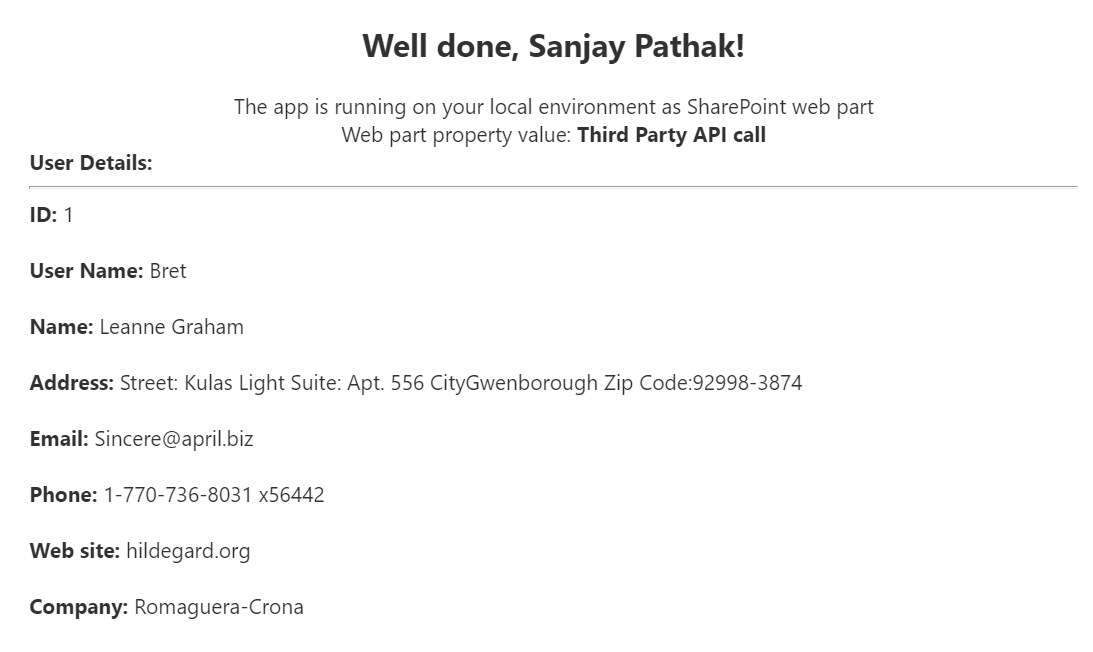
5. When properties of web part changed, values in web part needs to be re-rendered based on the changed values. For this we need to implement **componentDidUpdate()** method. This method is called when properties of react component are changed. Inside this method we are calling **InvokeAPIAndSetDataIntoState()** again to get data from API.

public componentDidUpdate(prevProps: IThirdpartyApiCallProps, prevState: IThirdpartyApiCallState, prevContext: any): void {

    this.InvokeAPIAndSetDataIntoState();

  }

Now **gulp serve** and check your web part with data from external API.



You can change the properties like **User ID** in property pane.

