## Requirement

How to navigate to other pages without page load when clicking the tree nodes rendered inside Sidebar which rendered in Layout page ?

## Solution

**ASP.NET CORE**

To achieve this requirement, we can use partial view in ASP.NET Core. This will make easy to load the one page content in the another page.

In TreeView node clicked event we can load the one page content using ajax method. On the Ajax success event we can set the result page content in main page.

Please refer the below code block.

**[\_RenderPartialView.cshtml]**

|  |
| --- |
| @{  TreeViewFieldsSettings templateFields = new TreeViewFieldsSettings();  TreeviewTemplate treeviewTemplate = new TreeviewTemplate();  templateFields.DataSource = treeviewTemplate.getTreeviewTemplate();  templateFields.HasChildren = "HasChild";  templateFields.Id = "Id";  templateFields.ParentID = "PId";  templateFields.Text = "Name";  }  <script>  function expanding(args) {  //Sidebar will expand along with treeview On clicking the treeview node to expand  var instance = document.getElementById("default-sidebar").ej2\_instances[0];  instance.show();  }  function collapsing(args) {  var instance = document.getElementById("default-sidebar").ej2\_instances[0];  //treeviewInstance = document.getElementById('template').ej2\_instances[0];  if (args.nodeData.id == "1") {  //When on collapsing the parent of treeview, sidebar also collapses to dock state  instance.hide();  }  }  </script>  <script>  var win = window;  win.isManualResizeTrigger = true;  setTimeout(() => { win.dispatchEvent(new Event('resize')); }, 500);  document.addEventListener('DOMContentLoaded', function () {  sidebarInstance = document.getElementById("default-sidebar").ej2\_instances[0];  sidebarInstance.show();  });  function onNodeClick(args) {  if (args.node.outerText != "Home") {  $.ajax({  url: ('/Home/' + args.node.outerText).toString(),  success: function (data) {  //Sidebar will consider the html element which has 'maincontent' class as its main content element  $(".maincontent").html(data);  }  })  } else {  $.ajax({  url: '/Home/',  success: function (data) {  $(".maincontent").html(data);  }  })  }  }  </script>  <script id="treeTemplate" type="text/x-template">  <div>  <img class="eimage" src="/images/**${**EImg**}**.png" alt="**${**EImg**}**" />  <div class="ename">**${**Name**}**</div>  </div>  </script>  <**ejs-sidebar** **id**="default-sidebar" **type**="Push" **dockSize**="80px" **enableDock**=true **target**="maincontent">  <**e-content-template**>  <**ejs-treeview** **id**="template" **nodeTemplate**="#treeTemplate" **nodeExpanded**="expanding" **nodeCollapsing**="collapsing" **cssClass**="custom" **fields**="templateFields" **nodeSelected**="onNodeClick">  </**ejs-treeview**>  </**e-content-template**>  </**ejs-sidebar**> |

Please refer to the below link for the sample.

<https://www.syncfusion.com/downloads/support/directtrac/general/ze/TreeNavigation-279330805>

**Blazor**

For Blazor platform we can simply render our Sidebar with TreeView component. In the TreeView component we have provided **navigationUrl** property using this property we can navigate the pages in Blazor.

Please refer the below code block.

**[MainLayout.razor]**

|  |
| --- |
| @inherits LayoutComponentBase  @using Syncfusion.Blazor  @using Syncfusion.Blazor.Navigations  <**SfSidebar** **Width**="250px">  <**ChildContent**>  <**SfTreeView** **TValue**="MailItem">  <**TreeViewFieldsSettings** **TValue**="MailItem" **Id**="Id" **DataSource**="@MyFolder" **Text**="FolderName" **NavigateUrl**="url"></**TreeViewFieldsSettings**>  </**SfTreeView**>  </**ChildContent**>  </**SfSidebar**>  <div class="main">  <div class="content px-4">  @Body  </div>  </div>  @code{  public class MailItem  {  public string Id { get; set; }  public string FolderName { get; set; }  public string url { get; set; }  }  List<MailItem> MyFolder = new List<MailItem>();  protected override void OnInitialized()  {  base.OnInitialized();  MyFolder.Add(new MailItem  {  Id = "1",  FolderName = "Index",  url = "/"  });  MyFolder.Add(new MailItem  {  Id = "2",  FolderName = "Counter",  url = "counter"  });  MyFolder.Add(new MailItem  {  Id = "3",  FolderName = "fetchdata",  url="fetchdata"  });  }  } |

Please refer to the below link for the sample.

<https://www.syncfusion.com/downloads/support/directtrac/general/ze/SidebarMain-140244166>

**React**

We have used **nodeSelected** event to navigate the other pages. When we select the tree nodes then we have updated the main page content using props. Refer the below code block.

[**TreeView.js**]

|  |
| --- |
| import \* as ReactDOM from 'react-dom';  import { withRouter } from 'react-router-dom'  import createHistory from "history/createBrowserHistory"  import { TreeViewComponent, SidebarComponent } from '@syncfusion/ej2-react-navigations';  import \* as React from 'react';  import { SampleBase } from './sample-base';  class BasicExample extends React.Component {   templateData = [          { "id": 1, "name": "about", "hasChild": true },          { "id": 2,"pid": 1, "name": "angular", },          { "id": 5, "name": "history", "hasChild": true, "expanded": true },          { "id": 6,"pid": 5, "name": "topics",  "selected": true },        ]    constructor() {     super(...arguments);      this.fields = { dataSource: this.templateData, id: 'id', parentID: 'pid', text: 'name', hasChildren: 'hasChild', expanded:'expanded' };      this.tree = null;      this.sidebarobj = null;      this.width = '290px';      this.mediaQuery = '(min-width: 600px)';    }    loadPage(args) {      var data = this.tree.getTreeData(args.node);      let routerLink = data[0].name;      this.props.history.push('/' + routerLink);    }    render() {      return (        <div className="control-section">          <div id="wrapper">            <div className="col-lg-12 col-sm-12 col-md-12">              <SidebarComponent id="sidebar-treeview" ref={Sidebar => this.sidebarobj = Sidebar} width={this.width} target={this.target} mediaQuery={this.mediaQuery}>                <div className='main-menu'>                   <TreeViewComponent id='main-treeview'  fields={this.fields} ref={treeview => { this.tree = treeview;}} nodeSelected={this.loadPage.bind(this)} />                </div>              </SidebarComponent>               <div>               <div>Render the main-content</div>               {this.props.children}               </div>            </div>          </div>        </div>      );    }  }  export default withRouter(BasicExample); |

Please refer to the below link for the sample.

<https://stackblitz.com/edit/react-router-sftree-sidebar>