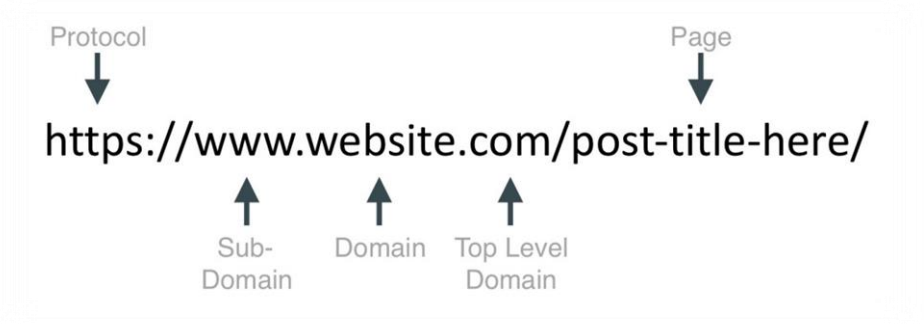
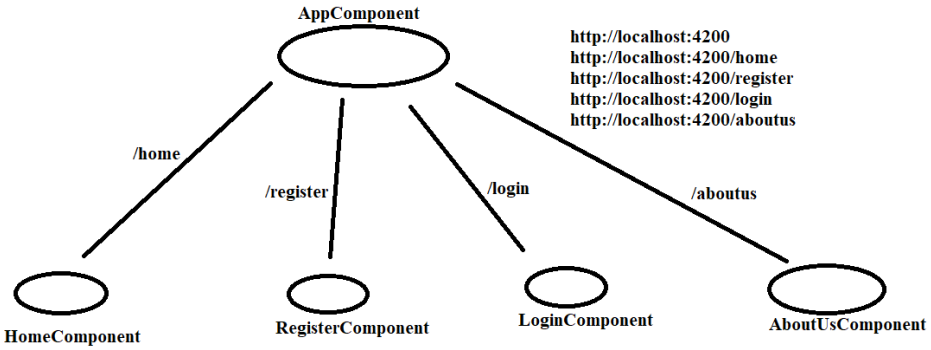
Routing & Navigation

A web application contains multiple resources in it. An user can access those resources by using URL.It stands for Uniform Resource Locator.



We know that Component is one of the building blocks of Angular Application.An App can have hierarchy of components. If , a user want to access a specific page, then the application should provide facility to navigate to that specific page.This can be done with “Routing & Navigation”.

Let us consider the following component hierarchy of application.



Here, The root component/parent component provides paths for its children. By navigating through the path, user can reach its associated component.Let us implement Routing & Navigation for above diagram.

Step-1: Create components

ng generate component home

ng generate component register

ng generate component login

ng generate component aboutus

Step-2: Once the components are created, let us create links to those by mentioning path at parent component's view(here ,it is app.component.html).

For this ,**RouterLink** directive can be used.

<a routerLink="/home">Home

<a routerLink="/register">Register

<a routerLink="/login">Login

<a routerLink="/aboutus">AboutUs

Now ,the links are associated with paths.When user click on those links,that perticular need to be loaded.For that, we can use "RouterOutlet" directive which can act like place holder for component selectors.

<router-outlet></router-outlet>

Step-3: Configure routes(Connecting components to paths)

It is time to connect above paths to corresponding components respectively.

For this, Angular app contains a special module for routing (app-routing.module.ts)

In that an empty array of type "Routes" like below

const routes:Routes=[];

This array represents objects which contains configuration of routes

Syntax:

{ **path:" path-of-component", component:class-name-of-component }**

Regarding to our application, 4 objects can be placed into that array

```
const routes: Routes = [
    {path:'home',component:HomeComponent},
    {path:'register',component:RegisterComponent},
    {path:'login',component:LoginComponent},
    {path:'aboutus',component:AboutusComponent},
  ];
```

Step-4:Now , the user can access the pages either by clicking on hyperlinks or by entering a specific url from the below list into address bar of browser.

http://localhost:4200/home

http://localhost:4200/register

http://localhost:4200/login

http://localhost:4200/aboutus

It is possible to enter invalid url(i.e. with the path not avaialable in our app) like

http://localhost:4200/contactus

This is invalid URL with respect to out application,because this path is not existed.

To deal with such invalid URLs, let use create a new component “PageNotFound”

ng generate component pagenotfound

And to deal with empty path, redirect to a specific path. So after creating these special Routes , the routes array look like below

```
const routes: Routes = [
                    {path:'home',component:HomeComponent},
                    {path:'register',component:RegisterComponent},
                    {path:'login',component:LoginComponent},
                    {path:'aboutus',component:AboutusComponent},
                   {path: ' ',redirectTo:'home',pathMatch:'full'},//to deal with empty path
                   {path:"**", component : PageNotFoundComponent}//to deal with invalid path
          ];
```

Note: The path-match strategy 'full' matches against the entire URL

Note: We should add wild card path("**") always as last obejct to the array.Otherwise,it can match with all strings and paths after that wont get chance to execute.