

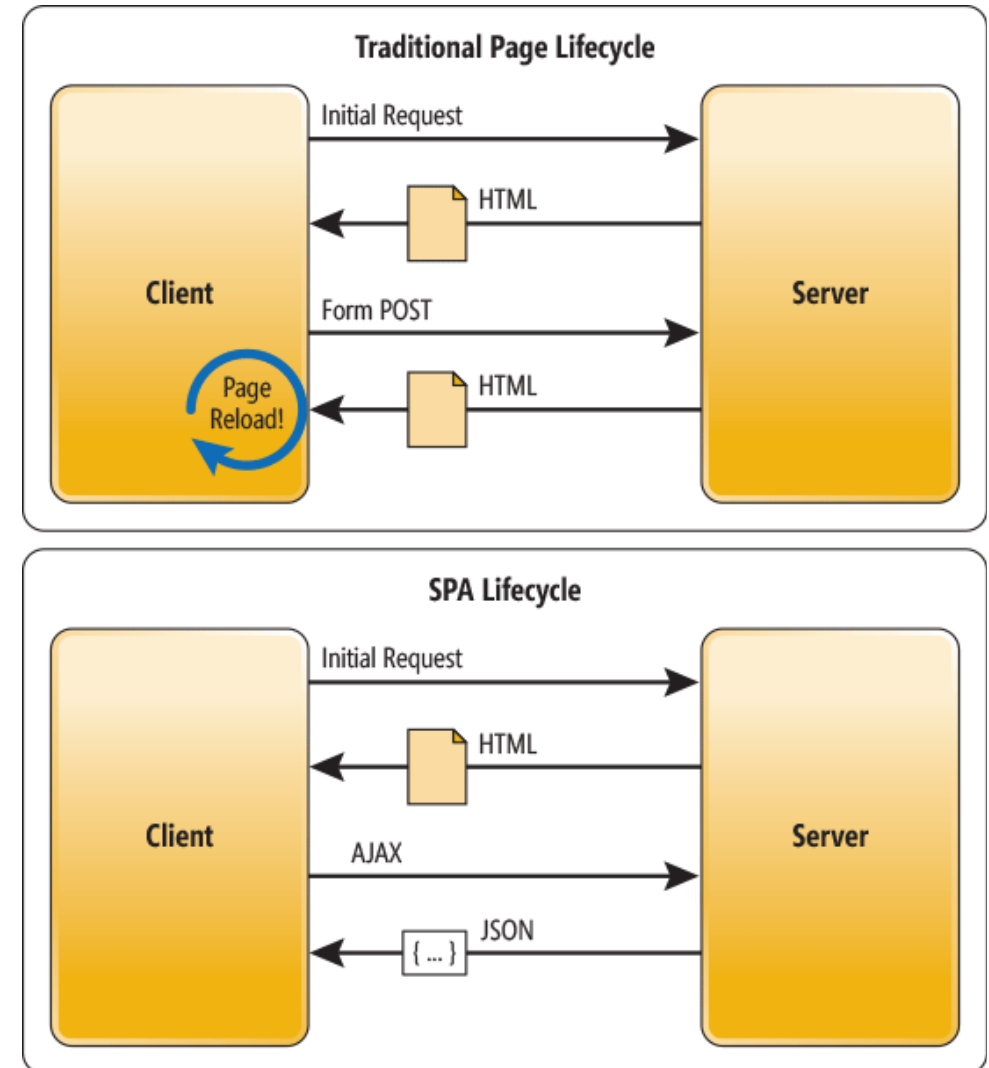


Day 35



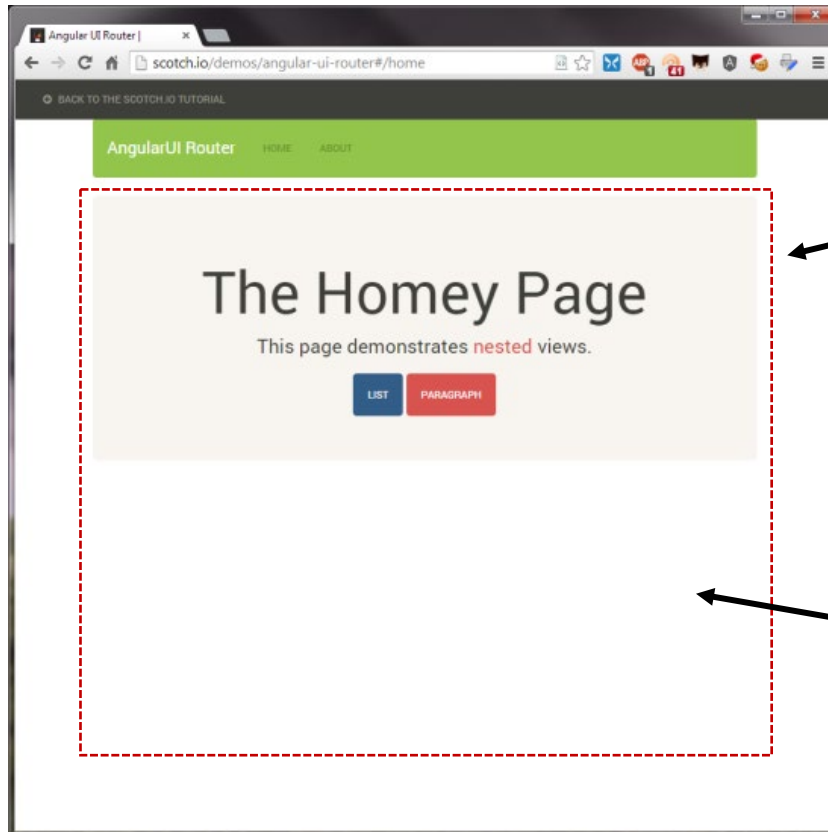
Single Page Application (SPA)

- SPA are web applications that loads a single HTML called the app shell
- It then dynamically update the contents of app shell as the user interacts with the application
- The app shell is not reloaded when its content is refreshed





SPA - App Shell

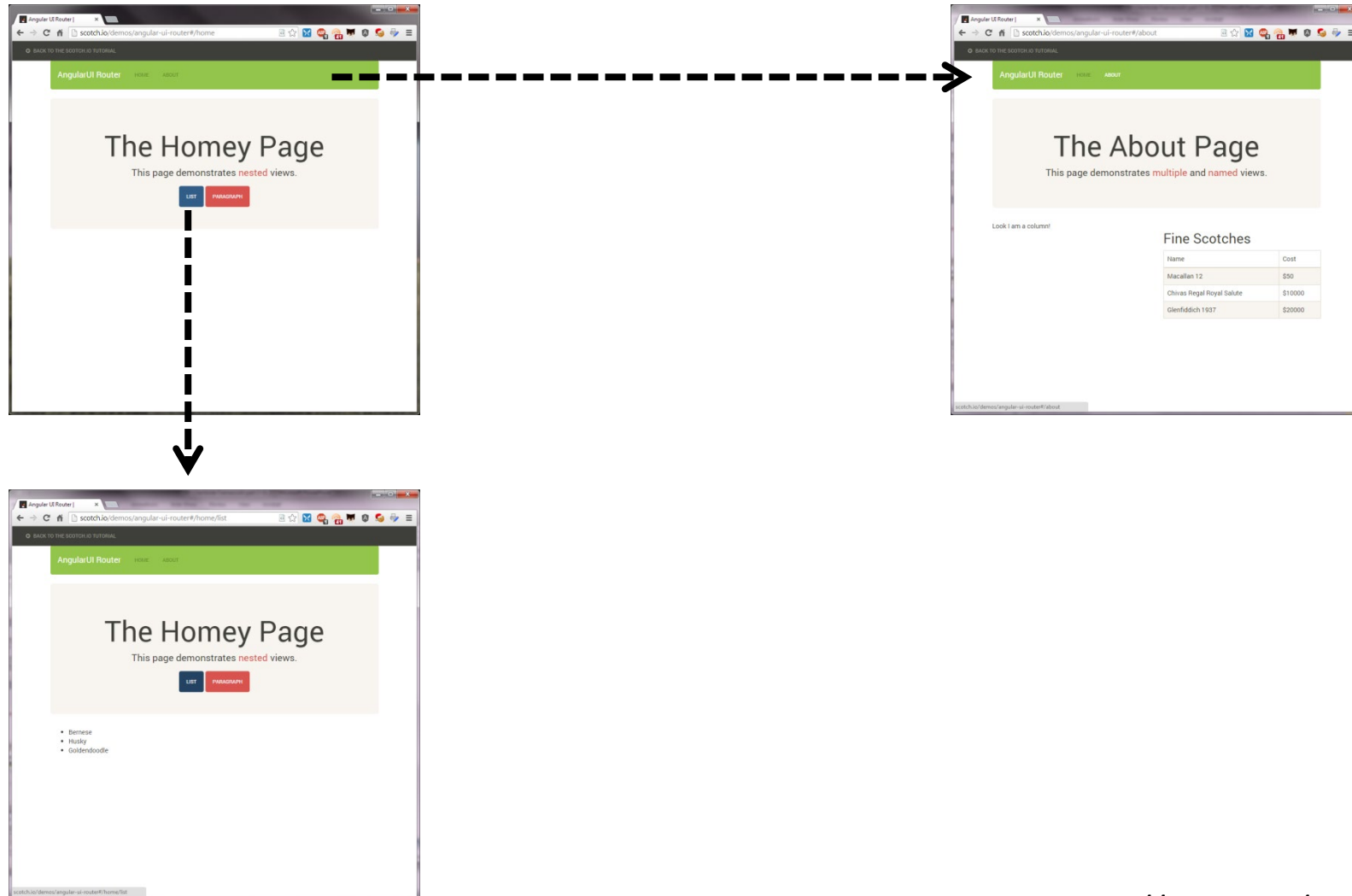


The 'application shell' that host the different views/components for different routes.

The component changes depending on the route. This is the outlet
Note the app shell is not reloaded



SPA - Client Side Routing



From <http://scotch.io/demos/angular-ui-router>



Changing Client Routes

- A change in the client route viz. content of the app shell is triggered by changing the URL
 - Angular monitors the address bar
- `RouterModule` manages the client side routes and maps the URL to a component according to the current URL in the address bar

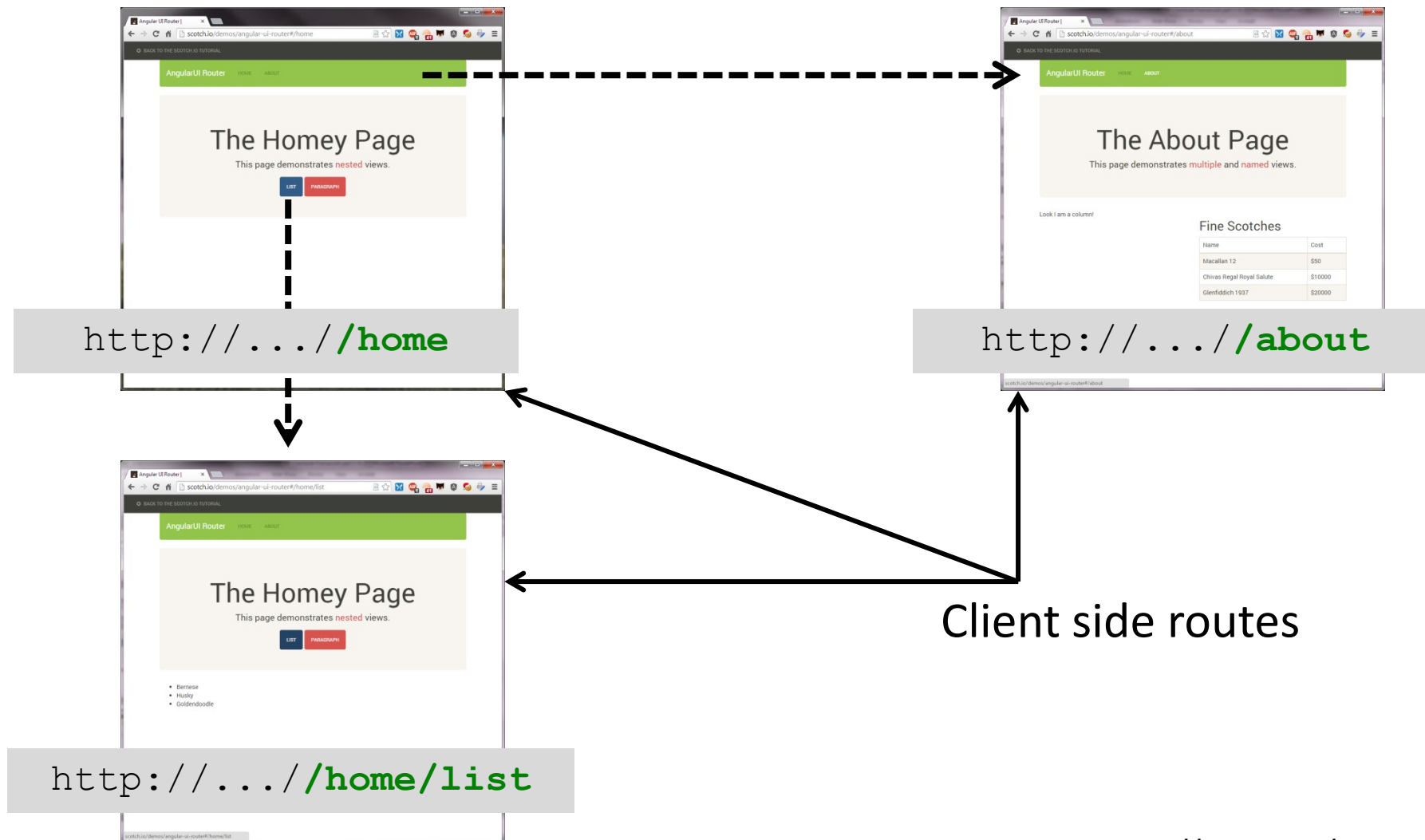
```
import { RouterModule } from '@angular/router';
```

```
@NgModule({  
  imports: [  
    ...,  
    RouterModule.forRoot({ ... })  
  ]  
})
```

When loading `RouterModule`, need to define the all the routes in the current application



SPA - Client Side Routing



From <http://scotch.io/demos/angular-ui-router>



Route

- A route consists of
 - URL
 - Component - this is the component that is loaded when the URL is activated
 - Pre-condition - for activating the route. Optional
 - Post-condition - for leaving the route. Optional

```
const appRoutes: Routes = [  
  { path: "dog", component: DogComponent },  
  ...  
]  
RouterModule.forRoot(appRoutes)
```

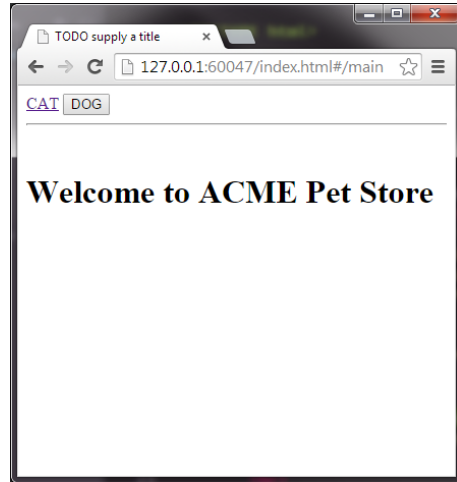
One route

Path, /dog. Don't include /

Component to activate

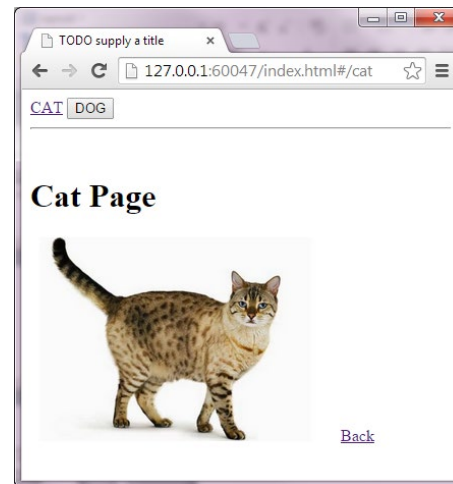


Example - Defining Routes



`url: main`
`component: HomeComponent`

`url: dog`
`component: DogComponent`



`url: cat`
`component: CatComponent`



Example - Defining Routes

```
import { Routes, RouterModule } from '@angular/router';
const appRoutes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'home', component: HomeComponent },
  { path: 'cat', component: CatComponent },
  { path: 'dog', component: DogComponent },
  { path: "**", redirectTo: '/', pathMatch: 'full' }
];
```

} Same routes

```
@NgModule({
  imports: [
    ...
    RouterModule.forRoot(appRoutes)
  ]
})
export class AppModule {
  ...
}
```

Catch-all route; will be activated
when no route matches



Outlet

- Outlet is the location where the components for the routes to display their respective components
- `<router-outlet>` element is used to demarcated where in the app shell should the component be displayed
- A typical place to is `app.component.html`



Router Outlet

```
<div>  
  <router-outlet>  
</div>
```

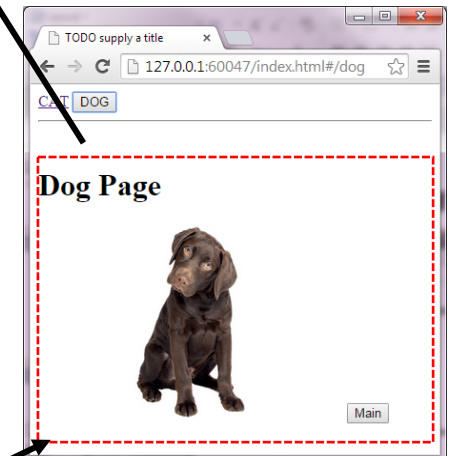
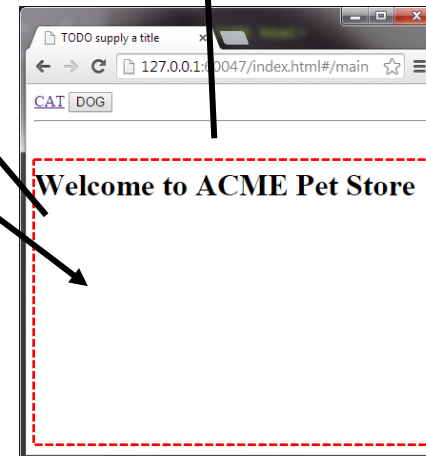
app.component.html

home.component.html

```
<h1>Welcome to ACME Pet Store</h1>
```

dog.component.html

```
<h1>Dog Page</h1>  
  
<button type="button">  
  Back  
</button>
```





Changing Client Side Route

- A change of client's view is triggered by changing the URL
- URL can be changed by
 - User interactivity - eg. user clicking on a link
 - Programmatically with TypeScript
- Use `routerLink` directive instead of `href` attribute in `<a>` or any clickable component eg. button

```
<a [routerLink]="['/home']">Home</a>
```

```
const appRoutes: Routes = [  
  { path: '', component: HomeComponent },  
  { path: 'home', component: HomeComponent },  
  ...  
];
```

Clicking the link will trigger the HomeComponent to be display at the router outlet



Changing Client Side Route Programmatically

- Router service exposes methods to programmatically to change routes
 - Router service is provided by RouterModule
 - Inject into your component
- Use `navigate()` method
 - Parameters is exactly the same as `routerLink` directive

```
this.router.navigate([ '/home' ] );
```

router is of type Router.
Injected into component



Changing Client Side Route Programmatically

app.component.ts

```
import { Router } from '@angular/router';  
@Component({  
  ...  
})
```

```
  ...
```

```
})
```


```
export class DogComponent {
```

```
  constructor(private router: Router) { }
```

Inject router service
into component

An arrow pointing from the text 'Inject router service into component' to the 'router' parameter in the constructor.

Click event
handler

An arrow pointing from the text 'Click event handler' to the 'toDog' method.

```
    toDog() {  
      this.router.navigate([ '/dog' ] );  
    }  
  }
```



Client Side Routes as Application States

- Routes can be used to represent the current state of the application
 - Eg. `/customer/3` might represent the customer with customer id 3 is current being displayed
- Parts of the route is then static and are dynamic
 - Eg. `/customer/1`, `/customer/2`, `/customer/3`
- Think of the dynamic portion as 'parameters'



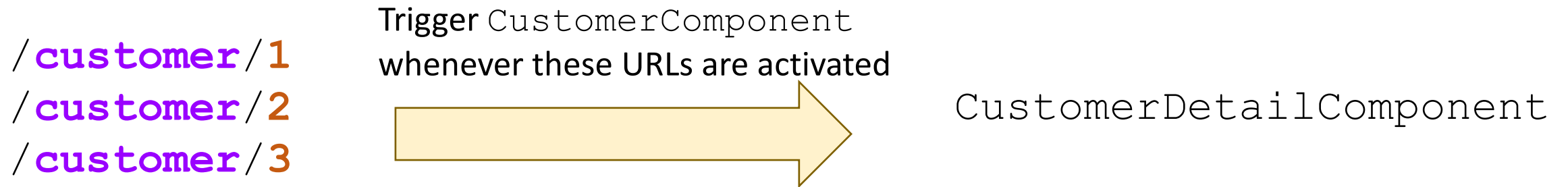
Passing Data Between Routes

- Data can be passed between routes as
 - path variable - the data is part of the route
 - Eg. `/customer/1`
 - query parameters - key/value pairs
 - Eg. `/customer/1?view=simple`
 - data held by a service
 - Since service are singletons, data can be 'deposited' by one route, and retrieved from another route
- Path variable and query parameters data can be access with the `ActivatedRoute` service



Path Variable - Parameterizing Routes

- Path variables in routes are parameterized with a colon (:)



```
const appRoutes: Routes = [  
  { path: 'customer/:custId', component: CustomerDetailComponent },  
  ...  
]
```

Route parameter - the variable part
of the route captured in :custId



Activating Parameterized Route

- By clicking

```
<a *ngFor="let c of customers"  
  [routerLink]="[ '/customer', c.custId ]">  
  {{ c.name }}  
</a>
```

- Programmatically

```
this.router.navigate([ '/customer', custId ] );
```



Query Parameters

- Programmatically

```
this.router.navigate([ '/customer', custId ]  
    , { queryParams: { view: 'simple' } } );
```

- By clicking

```
<a *ngFor="let c of customers"  
    [routerLink]="[ '/customer', c.custId ]"  
    [queryParams]="{ view: 'simple' }">  
    {{ c.name }}  
</a>
```



Retrieving the Path Variables and Query Parameters

- Use `ActivatedRoute` to retrieve the route parameter
 - `ActivatedRoute` is a service from `RouterModule`

```
@Component({ ... })
export class CustomerDetailComponent implements OnInit {
  constructor(private activatedRoute: ActivatedRoute,
               private title: Title) { }

  ngOnInit() {
    const custId = this.activatedRoute.snapshot.params.custId;
    const view = this.activatedRoute.snapshot.queryParams.view;
    title.setTitle(`Customer ${custId}`)
    // ...
  }
}
```

Retrieve the route parameter

title service

Sets the view's title



Parameter Routes Illustrated

```
<a [routerLink]="['/customer', 1]>Jumbo Eagle Corp</a>
```

```
RouterModule.forRoot([  
  ...  
  { path: '/customer/:custId',  
    component: CustomerDetailsComponent }  
])
```

ActivatedRoute.snapshot.params.**custId**


```
@Component({...})  
export class CustomerDetailComponent  
  implements OnInit {  
  constructor(private activatedRoute: ActivatedRoute,) { }
```



Binding Route Parameters to @Input


```
@Component({ ... })  
export class CustomerDetailComponent implements OnInit {
```

```
  @Input()  
  custId: string  
}
```

A curved arrow originates from the @Input() decorator and points to the custId property, indicating that the property is decorated with the @Input() decorator.

```
const appRoutes: Routes = [  
  ...  
]  
RouterModule.forRoot(appRoutes, { bindToComponentInputs: true })
```

Enable input binding in router

A straight arrow points from the text "Enable input binding in router" to the bindToComponentInputs property in the RouterModule.forRoot method call.



Route Guards

- Controls the entry and exit from a route
 - Eg. cannot navigate to a view if you are not login
 - Eg. prompt user to save work before leaving a view
- Route guards can
 - Prevent user from leaving a view by returning a `boolean`
 - Navigate to another view by returning a `UrlTree`
 - Can return the above as `Promise` or `Observable`
- Implement 2 functions
 - `CanActivateFn`
 - `CanDeactivateFn`





CanActivate Example

Going to a view

```
export const checkIfAuthenticated =  
  (route: ActivatedRouteSnapshot, state: routerStateSnapshot) => {  
    const authSvc = inject(AuthenticationService)  
    const router = inject(Router)  
  
    if (authSvc.isAuthenticated())  
      return true  
    return router.parseUrl("/help")  
  }
```

Assume a service that holds
the authentication state

Returns true/false depending on
user's authentication state

Create a UrlTree to navigate to /help.
/help must be configured as a route

```
const routes: Routes = [  
  { path: '', component: LoginComponent },  
  { path: 'customers', component: CustomerListComponent,  
    canActivate: [ checkIfAuthenticated ]  
  }  
]
```

Add one or more CanActivateFn to
the canActivate attribute of a route



CanDeactivate Example

The component the router
is navigating away

```
export const hasSaved: CanDeactivateFn<OrderFormComponent> =  
  (orderForm: OrderFormComponent, route: ActivatedRouteSnapshot  
    , state: routerStateSnapshot) => {
```

```
    if (orderForm.form.dirty)  
      return confirm('You have not saved the order.\nAre you  
should you want to leave?')
```

JavaScript function that returns true or false

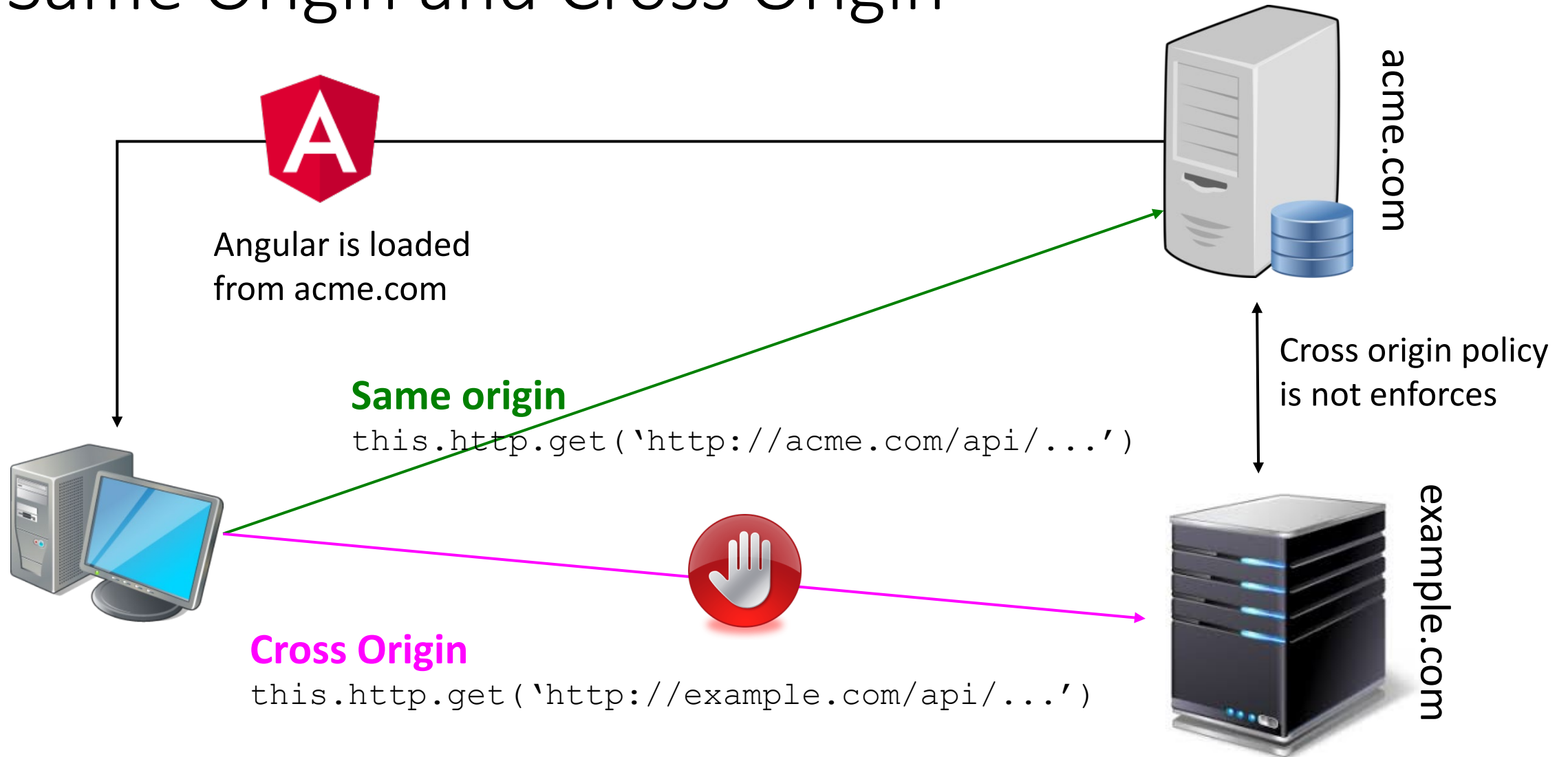
```
    return true  
  }
```

```
const routes: Routes = [  
  { path: '', component: LoginComponent },  
  { path: 'order', component: OrderFormComponent,  
    canDeactivate: [ hasSaved ]  
  }  
]
```

Add one or more CanDeactivateFn to
the canDeactivate attribute of a route



Same Origin and Cross Origin





Setting Angular Development for Same Origin

- Access cross origin resource, Angular and the REST backend is running on 2 different servers
 - Angular makes HTTP call to a separate server

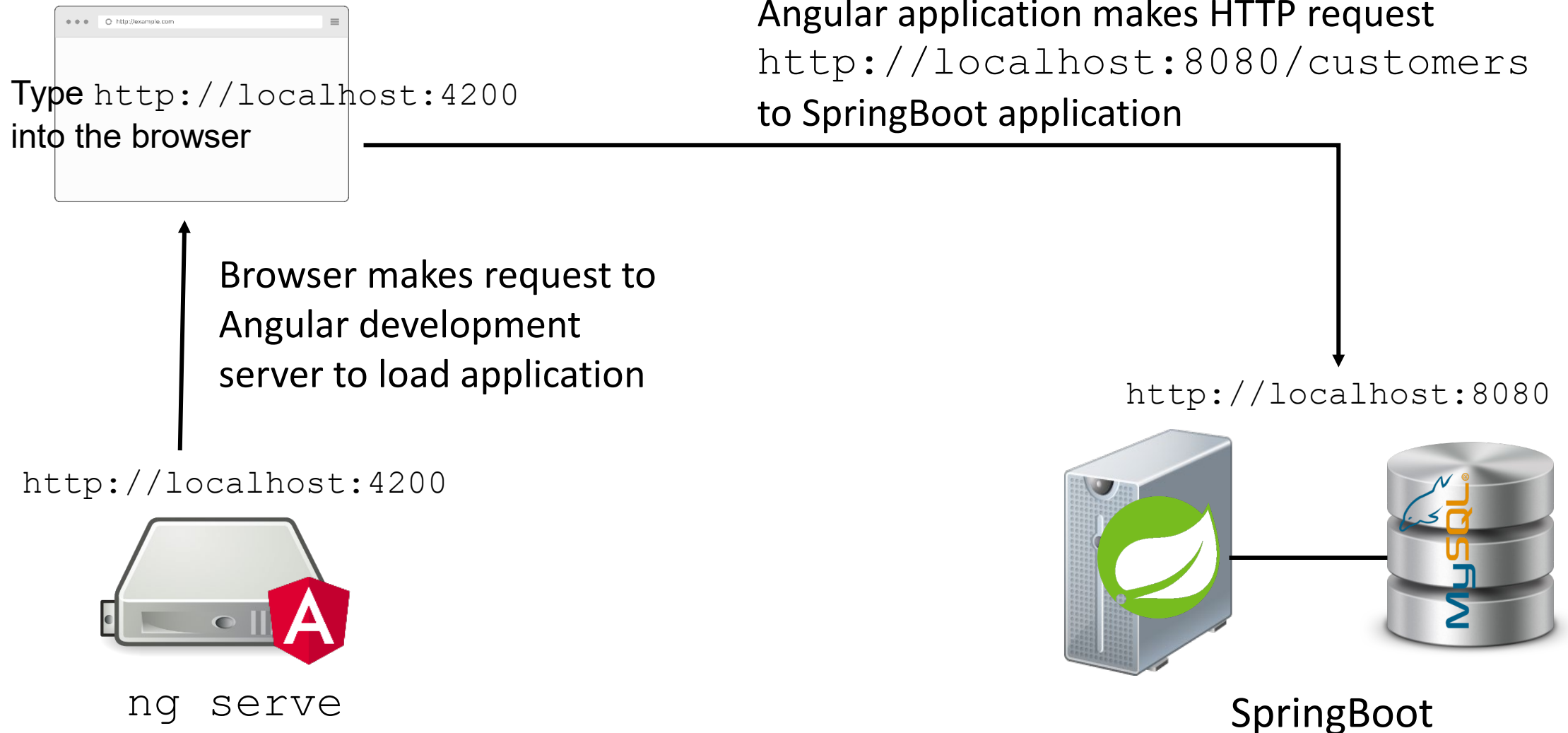
```
this.http.get('https://acme.com/api/customer/1')
```

- Same origin XHR calls are made to the same server from which the Angular application is loaded from
 - Problem because the call goes back to the Angular development server

```
this.http.get('/api/customer/1')
```



Development Environment



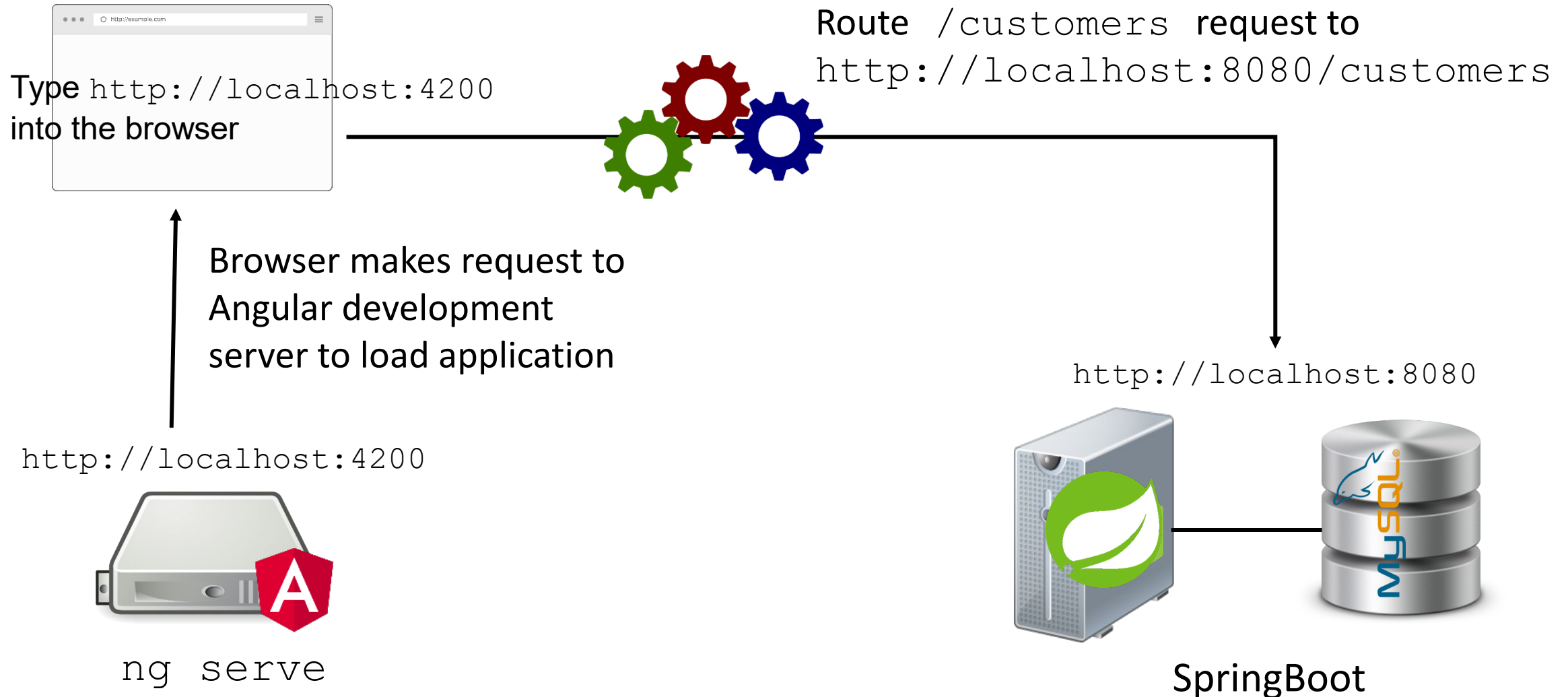


Serving Angular Application from SpringBoot

- Change all HTTP request from `http://<server>/customers` to just `/customers`
 - Make all request independent of the deployment since the Angular application is part of the SpringBoot application
- Create a proxy to route all request from `/customers` to `http://<server>/customers`
 - Run development server with proxy
 - During development, its still 2 servers
- Build Angular for production once the client-side application completes
- Copy Angular application artefacts to SpringBoot's application static folder
 - Eg. `public`



Development Environment





Change All HTTP Request

```
import { HttpClient, HttpParams } from '@angular/common/http';

import { firstValueFrom } from 'rxjs'
@Injectable()
export class CustomerService {

  constructor(private http: HttpClient) { }

  getAllCustomers(offset = 0, limit = 50): Promise<any> {
    const qs = new HttpParams()
      .set('offset': offset)
      .set('limit': limit);
    return (
      firstValueFrom(
        this.http.get('/customers', { params: qs })
      )
    );
  }
}
```

http://localhost:8080/customers

A black arrow points from the text 'http://localhost:8080/customers' to the path '/customers' in the code snippet, indicating that the path is relative to the base URL.



Proxy File

Forward all HTTP request starting with **/api** to
http://localhost:8080

proxy.config.json

```
{
  "/api": {
    "target": "http://localhost:8080",
    "secure": false
  }
}
```




Configure CLI to Use Proxy

ng **serve**

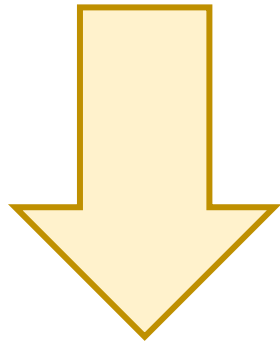
```
{
  "projects": {
    "<app_name>": {
      "architect": {
        "serve": {
          "builder": "@angular-devkit/build-angular:dev-server",
          "options": {
            "proxyConfig": "proxy.conf.json"
          }
        }
      }
    }
  }
}
```

Add these lines

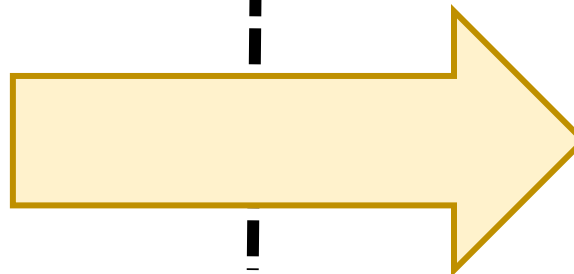


Build and Copy Artefacts to SpringBoot

ng build



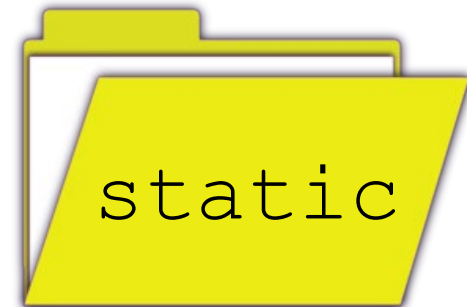
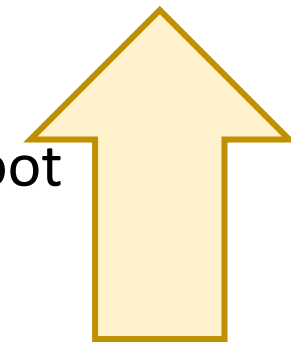
Compile Angular application



Copy all files to public

mvn package

Angular application served from SpringBoot





Unused



Material List

Star Wars

Luke Skywalker

C-3PO

R2-D2

Darth Vader

Leia Organa

Owen Lars

Beru Whitesun lars

R5-D4

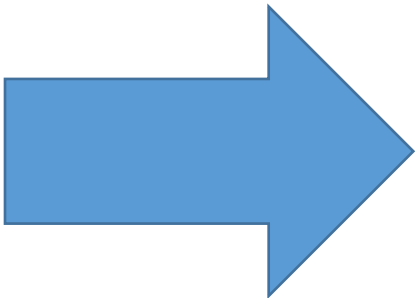
Biggs Darklighter

Obi-Wan Kenobi

PeopleComponent

/films/1
/films/2
/films/3

People id



Star Wars

The Phantom Menace

Episode: 1

Turmoil has engulfed the Galactic Republic. The taxation of trade routes to outlying star systems is in dispute. Hoping to resolve the matter with a blockade of deadly battleships, the greedy Trade Federation has stopped all shipping to the small planet of Naboo. While the Congress of the Republic endlessly debates this alarming chain of events, the Supreme Chancellor has secretly dispatched two Jedi Knights, the guardians of peace and justice in the galaxy, to settle the conflict....

Attack of the Clones

Episode: 2

There is unrest in the Galactic Senate. Several thousand solar systems have declared their intentions to leave the Republic. This separatist movement, under the leadership of the mysterious Count Dooku, has made it difficult for the limited number of Jedi Knights to maintain peace and order in the galaxy. Senator Amidala, the former Queen of Naboo, is returning to the Galactic Senate to vote on the critical issue of creating an ARMY OF THE REPUBLIC to assist the overwhelmed Jedi....

FilmComponent



Material List

```
@NgModule({
  imports: [
    RouterModule.forRoot([
      { path: '', component: PeopleComponent },
      { path: '/people', component: PeopleComponent },
      { path: 'films/:pId', component: FilmComponent }
    ])
  ]
})
export class AppModule {
```



Material List - PeopleComponent

```
<mat-nav-list>
  <mat-list-item *ngFor="let p of people; let i = index;"
    (click)="getFilms(i)">
    {{ p.name }}
  </mat-list-item>
</mat-nav-list>
```

Get all the film ids that the character appeared in; save this list in an array

```
getFilms(idx) {
  const films = //an array of films by idx
  this.router.navigate(['/films', idx],
    { queryParams: { fids: films.join('|') } });
}
```

Join all the film ids to a string delimited by |. Pass these over to the next route query parameter

Star Wars

Luke Skywalker

C-3PO

R2-D2

Darth Vader

Leia Organa

Owen Lars

Beru Whitesun lars

R5-D4

Biggs Darklighter

Obi-Wan Kenobi

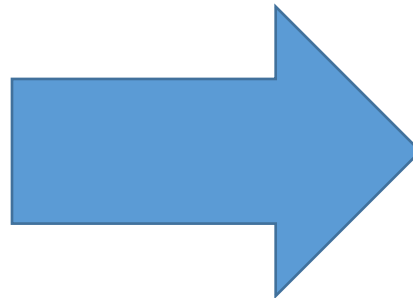


Material List

Star Wars
Luke Skywalker
C-3PO
R2-D2
Darth Vader
Leia Organa
Owen Lars
Beru Whitesun lars
R5-D4
Biggs Darklighter
Obi-Wan Kenobi

/films/**10**?**fids**=2|5|4|6|3|1

Additional parameters can
be passed as query
parameter from one route
to the next



Star Wars
The Phantom Menace Episode: 1 Turmoil has engulfed the Galactic Republic. The taxation of trade routes to outlying star systems is in dispute. Hoping to resolve the matter with a blockade of deadly battleships, the greedy Trade Federation has stopped all shipping to the small planet of Naboo. While the Congress of the Republic endlessly debates this alarming chain of events, the Supreme Chancellor has secretly dispatched two Jedi Knights, the guardians of peace and justice in the galaxy, to settle the conflict....
Attack of the Clones Episode: 2 There is unrest in the Galactic Senate. Several thousand solar systems have declared their intentions to leave the Republic. This separatist movement, under the leadership of the mysterious Count Dooku, has made it difficult for the limited number of Jedi Knights to maintain peace and order in the galaxy. Senator Amidala, the former Queen of Naboo, is returning to the Galactic Senate to vote on the critical issue of creating an ARMY OF THE REPUBLIC to assist the overwhelmed Jedi....



Material List

```
<mat-card *ngFor="let f of films">
  <mat-card-header>
    <mat-card-title>
      <h2>{{ f.title }} </h2>
    </mat-card-title>
    <mat-card-subtitle>
      {{ f.episode_id }}
    </mat-card-subtitle>
    <mat-card-content>
      <p>{{ f.opening_crawl }}</p>
    </mat-card-content>
  </mat-card>
```

```
ngOnInit() {
  const pId = this.activatedRoute.snapshot.params.pId;
  const fids = this.activatedRoute.snapshot.queryParams
    .fids.split('|');
  this.films = //get the films
}
```

Star Wars

The Phantom Menace

Episode: 1

Turmoil has engulfed the Galactic Republic. The taxation of trade routes to outlying star systems is in dispute. Hoping to resolve the matter with a blockade of deadly battleships, the greedy Trade Federation has stopped all shipping to the small planet of Naboo. While the Congress of the Republic endlessly debates this alarming chain of events, the Supreme Chancellor has secretly dispatched two Jedi Knights, the guardians of peace and justice in the galaxy, to settle the conflict...

Attack of the Clones

Episode: 2

There is unrest in the Galactic Senate. Several thousand solar systems have declared their intentions to leave the Republic. This separatist movement, under the leadership of the mysterious Count Dooku, has made it difficult for the limited number of Jedi Knights to maintain peace and order in the galaxy. Senator Amidala, the former Queen of Naboo, is returning to the Galactic Senate to vote on the critical issue of creating an ARMY OF THE REPUBLIC to assist the overwhelmed Jedi...