Angular JS 4 (Day 4)

Presentation By: Asfiya Khan (Technical Trainer)

www.cybage.com

Document History

Version No.	Authored/ Modified by	Remarks/ Change History	Date <dd- mon-yy >
1.0	Asfiya Khan	First version of Angular 4	13 March 2018

Course Structure

Target audience	Trainee,SE,SSE
Level	1,2,3
Pre-requisites	Javascript,TypeScript,HTML,CSS
Training methods	Presentation , Demos, Hands-on
Evaluation	Multiple Choice Question

Agenda

- Architecture and Components
- Data Binding and Pipes
- Routing and Navigation
- Templates ,Interpolation and Directives
- Angular Modules
- Services and Dependency Injection
- Ng-Forms
- Retrieving data using HTTP

How Routing Works

```
▼<pm-app>
  ▼<div>
    ▶<nav class="navbar navbar-default">…</nav>
    ▼<div class="container">
        ::before
        <router-outlet></router-outlet>
      ▼<ng-component _nghost-jfk-3>
        ▼<div _ngcontent-jfk-3 class="panel panel-primary">
            <div _ngcontent-jfk-3 class="panel-heading">
                    Product List
                </div>
          ▼<div _ngcontent-jfk-3 class="panel-body">
              ::before
            ▶<div _ngcontent-jfk-3 class="row">…</div>
            ▼<div _ngcontent-jfk-3 class="table-responsive">
              ▼<table _ngcontent-jfk-3 class="table">
                ▶<thead _ngcontent-jfk-3>…</thead>
                ▶<tbody _ngcontent-jfk-3>…</tbody>
                </table>
              </div>
              ::after
            </div>
          </div>
        </ng-component>
        ::after
      </div>
    </div>
  </pm-app>
```

- Configure a route for each component
- Define options/actions
- Tie a route to each option/action
- Activate the route based on user action
- Activating a route displays the component's view

How Routing Works


```
<a routerLink="/products">Product List</a>
```

```
{ path: 'products', component: ProductListComponent }
```

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

product-list.component.ts

```
import { Component } from '@angular/core';

@Component({
  templateUrl: './product-list.component.html'
})
export class ProductListComponent { }
```

Configuring Routes

app.module.ts

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

@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule,
    RouterModule
  ],
  declarations: [
    ...
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Configuring Routes

app.module.ts

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

@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule,
    RouterModule.forRoot([])
  ],
  declarations: [
    ...
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Configuring Routes

app.module.ts

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

@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule,
    RouterModule.forRoot([], { useHash: true })
  ],
  declarations: [
    ...
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Configuring Routes

```
[
 { path: 'products', component: ProductListComponent },
 { path: 'products/:id', component: ProductDetailComponent },
 { path: 'welcome', component: WelcomeComponent },
 { path: '', redirectTo: 'welcome', pathMatch: 'full' },
 { path: '**', component: PageNotFoundComponent }
]
```

Placing the Views

app.component.ts

```
...

@Component({
    selector: 'pm-root',
    template: `
     <ul class='nav navbar-nav'>
       <li><a [routerLink]="['/welcome']">Home</a></li>
       <li><a [routerLink]="['/products']">Product List</a></li>
     </ul>
     <router-outlet></router-outlet>
    `
})
```

DEMO

Service

A class with a focused purpose.

Used for features that:

- Are independent from any particular component
- Provide shared data or logic across components
- Encapsulate external interactions

How Does It Work?

Service

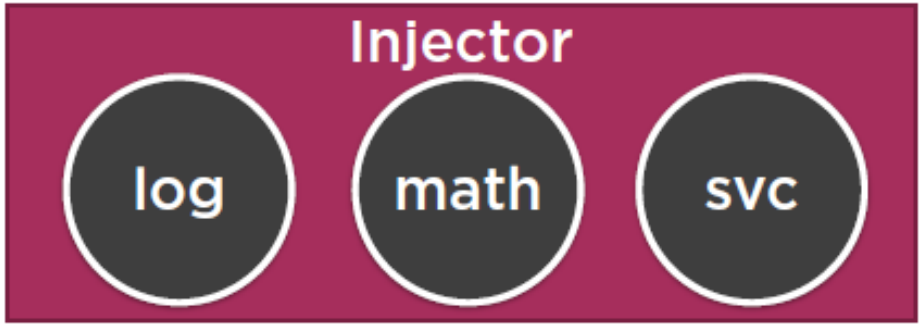
```
export class myService {}
```

Component

```
let svc = new myService();
```

svc

How Does It Work?



Service

```
export class myService {}
```

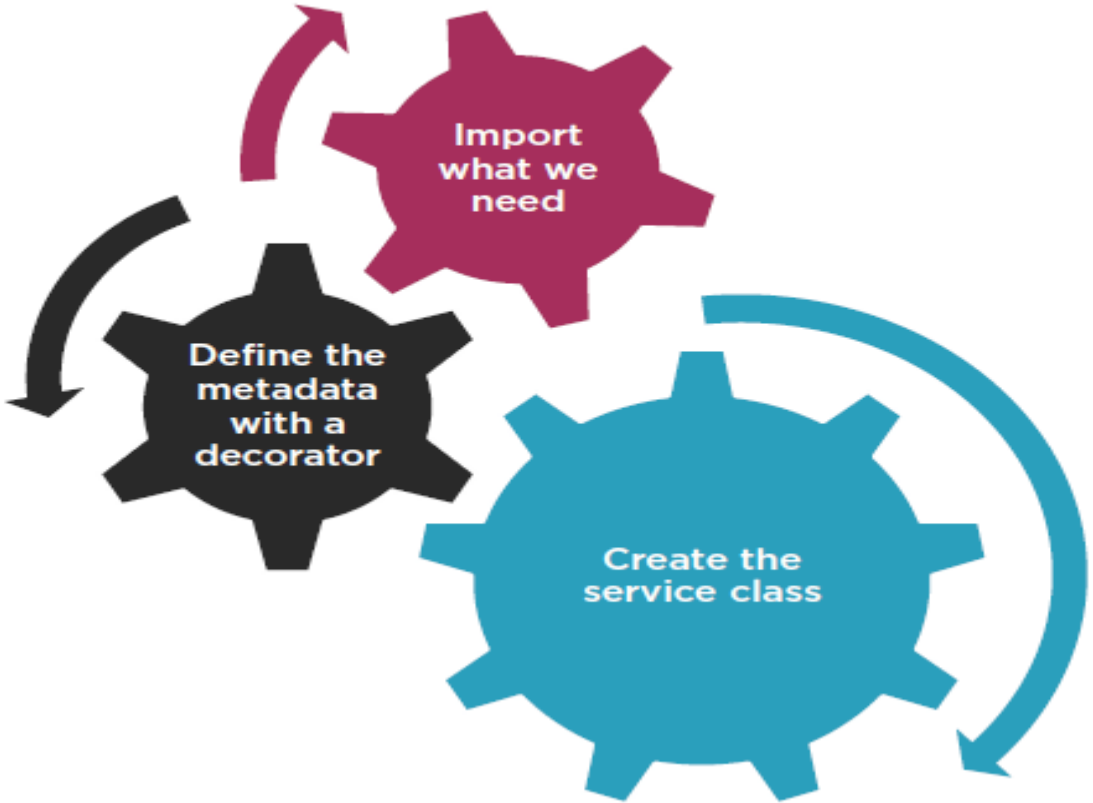
Component

```
constructor(private _myService) {}
```

Dependency Injection

A coding pattern in which a class receives the instances of objects it needs (called dependencies) from an external source rather than creating them itself.

Building a Service



Building a Service

product.service.ts

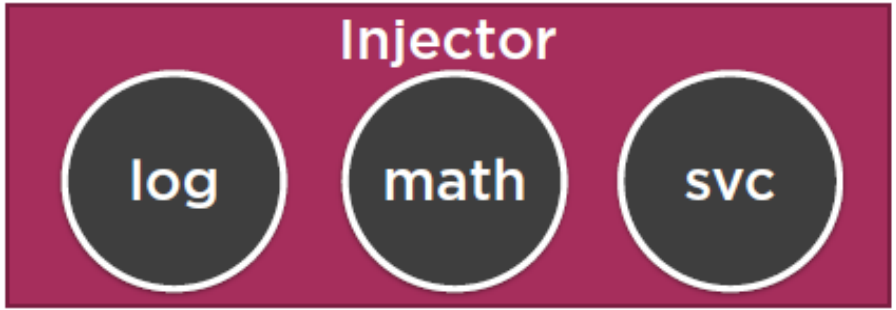
```
import { Injectable } from '@angular/core'

@Injectable()
export class ProductService {

  getProducts(): IProduct[] {
  }

}
```

Registering the Service



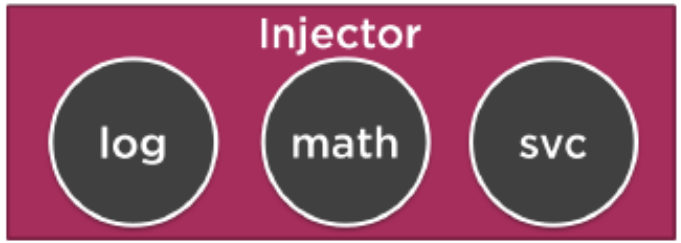
Service

```
export class myService {}
```

Component

```
constructor(private _myService) {}
```

Registering a Service



Register a provider

- Code that can create or return a service
- Typically the service class itself

Define in component OR Angular module metadata

Registered in component:

- Injectable to component AND its children

Registered in Angular module:

- Injectable everywhere in the application

Registering a Provider

app.component.ts

```
...
import { ProductService } from './products/product.service';

@Component({
  selector: 'pm-root',
  template: `
    <div><h1>{{pageTitle}}</h1>
      <pm-products></pm-products>
    </div>
    `,
  providers: [ProductService]
})
export class AppComponent { }
```

Injecting the Service

product-list.component.ts

```
...
import { ProductService } from './product.service';

@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {
 private _productService;
 constructor(productService: ProductService) {
   _productService = productService;
 }

}
```

Steps to create service

Service class

- Clear name
- Use PascalCasing
- Append "Service" to the name
- `export` keyword

Service decorator

- Use Injectable
- Prefix with @; Suffix with ()

Import what we need

Select the appropriate level in the hierarchy

- Root component if service is used throughout the application
- Specific component if only that component uses the service
- Otherwise, common ancestor

Component metadata

- Set the providers property
- Pass in an array

Import what we need

Specify the service as a dependency

Use a constructor parameter

Service is injected when component is instantiated

CYBAGE

Thank You!

www.cybage.com