Angular Training

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com

Schedule for Angular Training

Day#	Date	Topic
Day-1	21-Feb-2024	• Introduction to Angular, Components
Day-2	24-Feb-2024	• Components • Templates
Day-3	25-Feb-2024	• Template Driven & Reactive Forms • Pipes & Data Formatting
Day-4	**26-Feb-2024**	• **HTTP Client** • **Services & Dependency Injection**
Day-5	27-Feb-2024	• Angular Routing • Angular Modules
Day-6	28-Feb-2024	• Observables & RxJS Library

Server Communication Using HTTP Client

Narasimha

Sr. IT Trainer/Consultant

Why do we need Server Communication?

- Most front-end applications need to communicate with a server over the HTTP protocol, to send or receive data and access other back-end services.
- Angular provides a client HTTP API for Angular applications, the HttpClient service class in @angular/common/http.

Why do we need Server Communication?

- Most front-end applications need to communicate with a server over the HTTP protocol, to send or receive data and access other back-end services.
- Angular provides a client HTTP API for Angular applications, the HttpClient service class in @angular/common/http.

Pre-Requisite Concepts

- AJAX
- JSON
- XmlHttpRequest / Fetch / $.ajax() / axios / HttpClient
- Server Side Code : Web API
- JSON Server

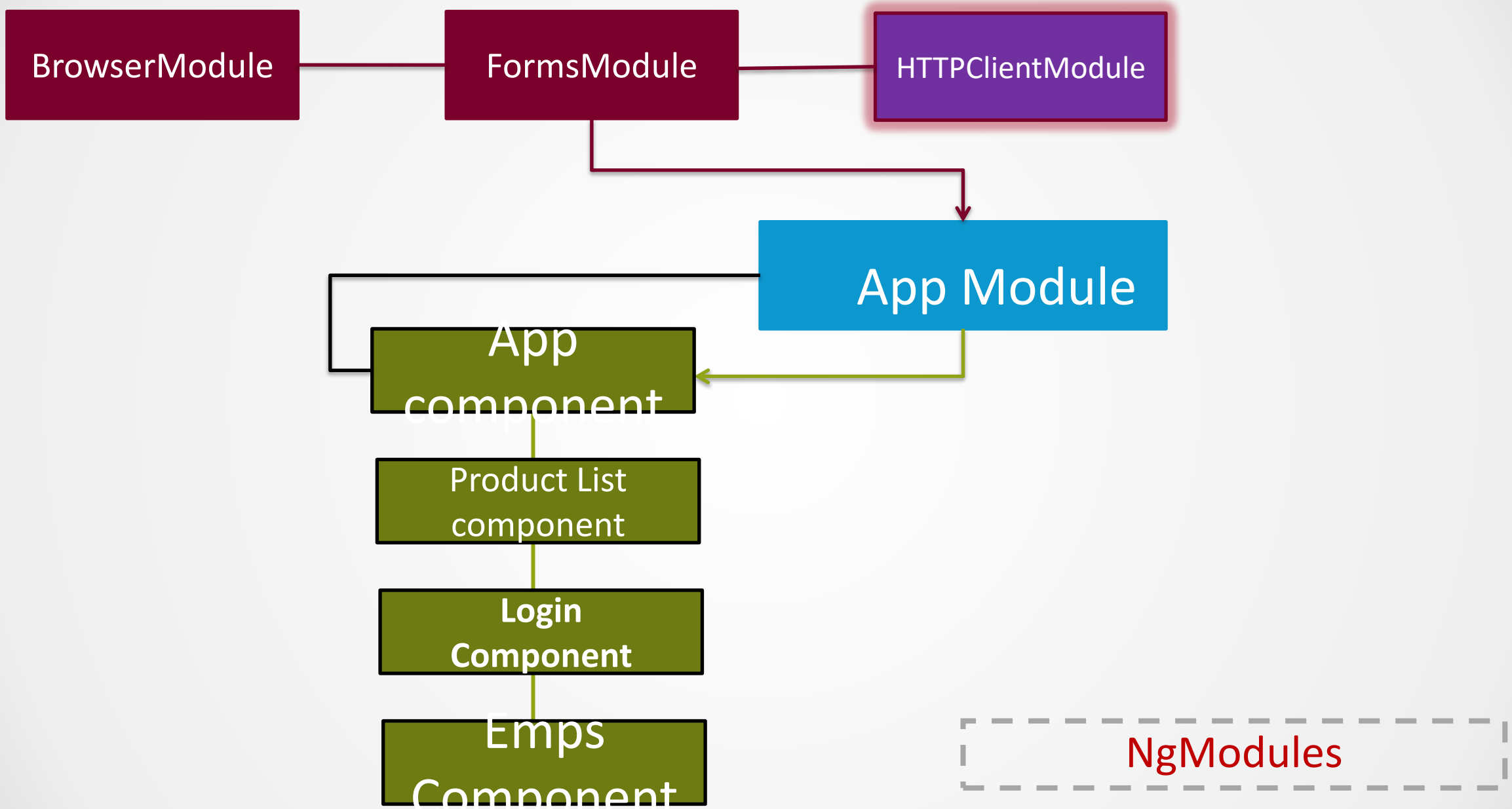
Understanding Http Calls

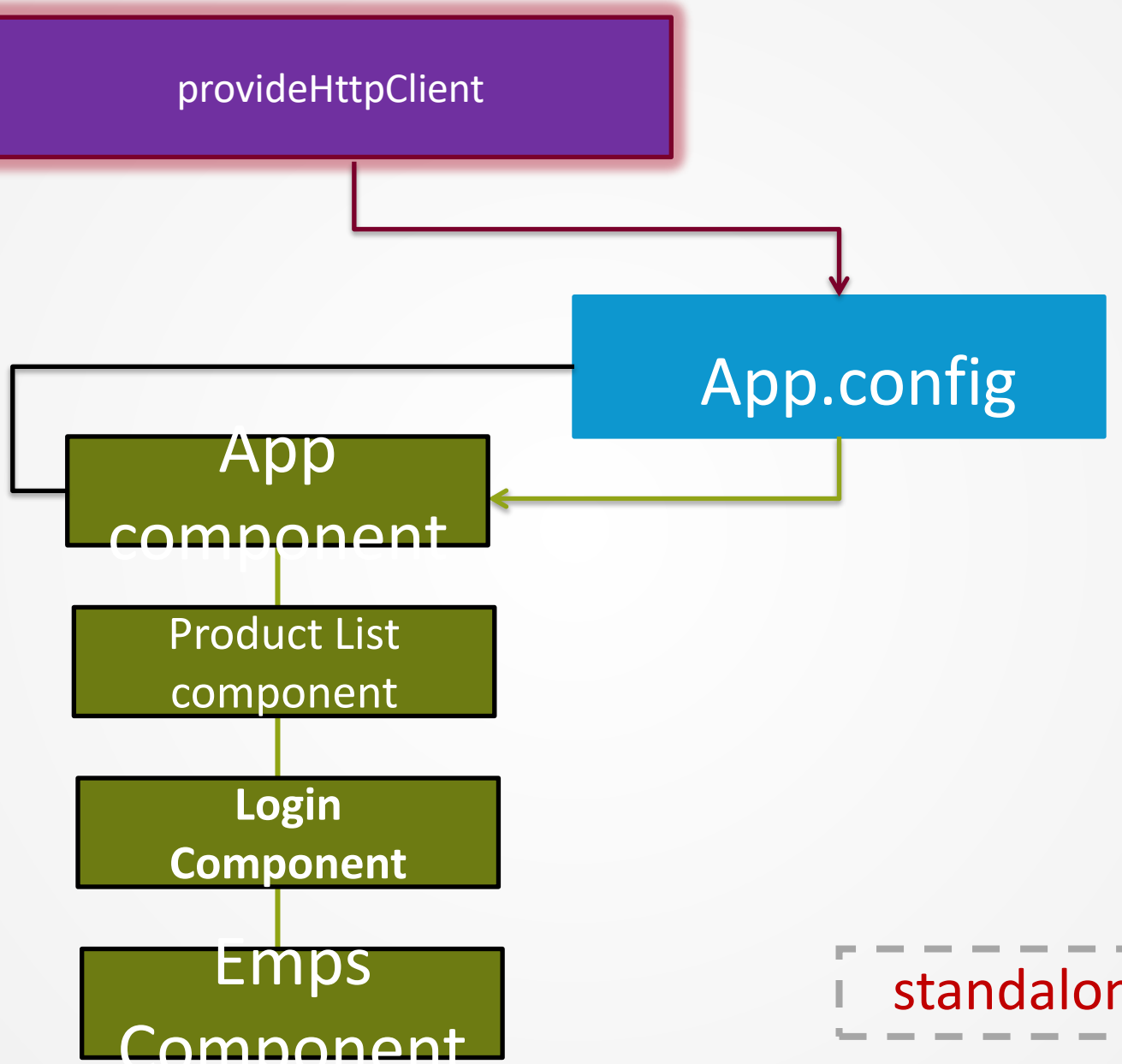
Narasimha

Sr. IT Trainer/Consultant

Angular Library : HttpClient

- Angular provides a client HTTP API for Angular applications, the **HttpClient service class** in @angular/common/http.
- Before you can use HttpClient, you need to import the Angular HttpClientModule
- You can then inject the HttpClient service as a dependency of an application class


BrowserModule
FormsModule
HTTPClientModule
App Module
App component
Product List component
Login Component
Emps Component
NgModules


provideHttpClient
App.config
App component
Product List component
Login Component
Emps Component
standalone Components

Usage of HttpClient in Component

```
@Component({
  selector: 'app-customer-list',
  templateUrl: './customer-list.component.html',
  styleUrls: ['./customer-list.component.css’]
})
export class CustomerListComponent {

  public customersArray:any[] = [];

  constructor(private httpObj:HttpClient)
  {
  }
}
```

Usage of HttpClient in Component

```
@Component({
  selector: 'app-customer-list',
  standalone: true,
  imports: [CommonModule],
  providers: [HttpClient],
  templateUrl: './customer-list.component.html',
  styleUrl: './customer-list.component.css'
})
export class CustomerListComponent {

  public customersArray:any[] = [];

  constructor(private httpObj:HttpClient){
  }
}
```

Usage of HttpClient in Component

```
constructor(private httpObj:HttpClient){

  }


this.httpObj.get(url).subscribe((resData) =>
{
        // handle the response
});
```

Usage of HttpClient in Component

```
export class HttpDemo1Component {

      public customersArray:any[] = [];

      constructor(private httpObj:HttpClient)  {    }

     getDataButtonClick()  {
              let url = "https://www.w3schools.com/angular/customers.php";
             this.httpObj.get(url).subscribe( (resData:any) =>
               {
                  this.customersArray = resData.records;
               } );
  }
}
```

Usage of HttpClient in Component

Nuget Package : Microsoft.AspNet.WebApi.Cors

```
builder.Services.AddCors(o => o.AddPolicy("AllowOrigin", builder =>
{
    builder.AllowAnyOrigin()
           .AllowAnyMethod()
           .AllowAnyHeader();
}));
```

```
app.UseCors("AllowOrigin");
```


Practical
HandsOns

Performing CRUD operations

Narasimha

Sr. IT Trainer/Consultant

CRUD Opeations

- GET : this.httpObj.get(url)
- POST : this.httpObj.post(url, obj)
- PUT : this.httpObj.put(url, obj)
- DELETE : this.httpObj.delete(url)

CRUD Opeations

1.	Read All	-	get(url)	-	/depts
2.	Read Single	-	get(url)	-	/depts/20
3.	Create	-	post(url, obj)	-	/depts
4.	Update	-	put(url, obj)	-	/depts/20
5.	Delete	-	delete(url)	-	/depts/20

Perform AJAX Operations using Angular Services

Narasimha

Sr. IT Trainer/Consultant


Practical
Hands-Ons

Dependency Injection

(Using Angular Services)

Narasimha

Sr. Corporate Trainer, Mentor

tnrao.trainer@gmail.com

What is Dependency Injection ?

Narasimha

Sr. IT Trainer/Consultant

What is Dependency Injection

- Dependency Injection(DI) is a design pattern and mechanism for creating and delivering some parts of an application to other parts of an application that require them.
- DI is one of the fundamental concepts in Angular.
- Angular supports this design pattern and you can use it in your applications to increase flexibility and modularity.

How does DI work in Angular?

Narasimha

Sr. IT Trainer/Consultant

How does DI work in Angular?

- Two main roles exist in the DI system: dependency consumer and dependency provider.
- Angular facilitates the interaction between dependency consumers and dependency providers using **Injector**.
- When a dependency is requested, the injector checks its registry to see if there is an instance already available there.
- If not, a new instance is created and stored in the registry.

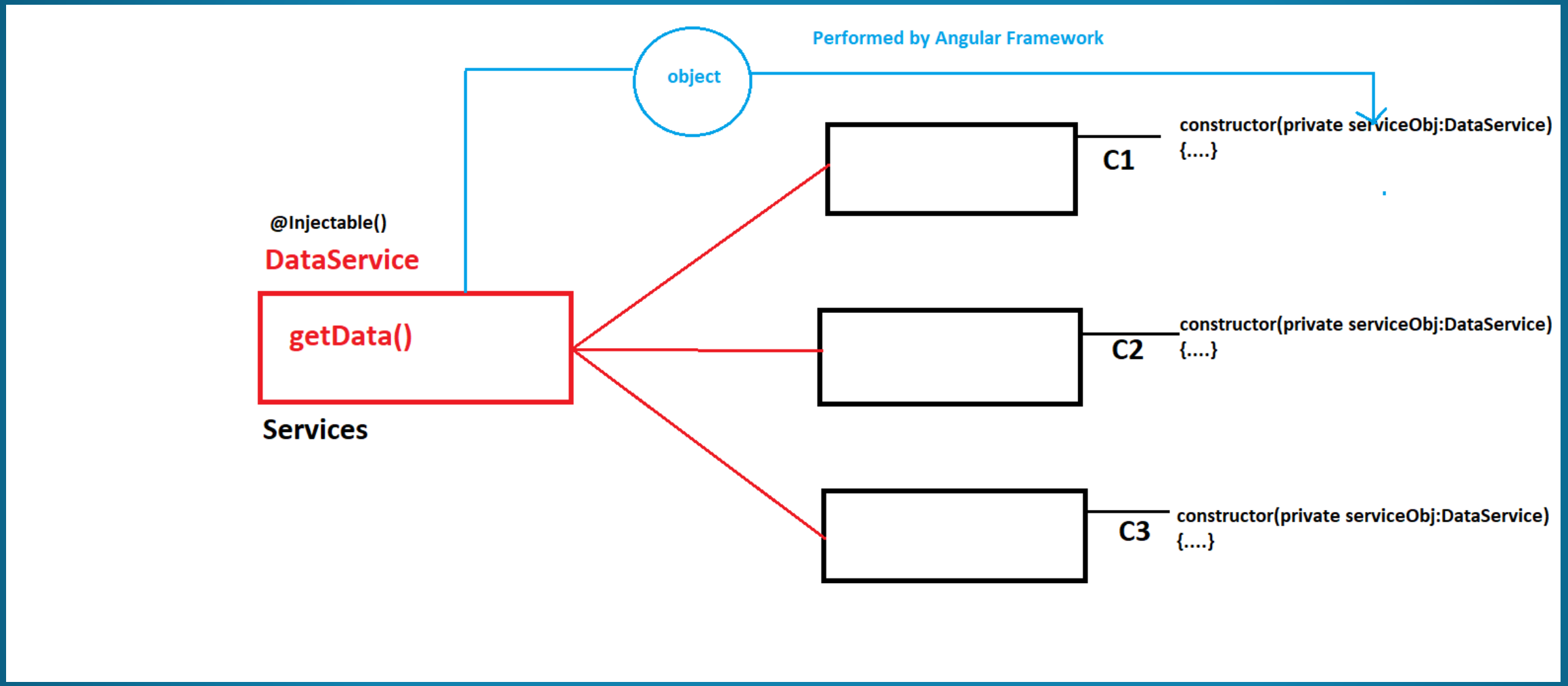
What and Why services?

Narasimha

Sr. IT Trainer/Consultant

What and Why services?

- In Angular, dependencies are typically services.
- A service is typically a class with a narrow, well-defined purpose.
- A component is one type of class that can use DI.
- Angular distinguishes components from services to increase modularity and reusability.
- By separating a component's view-related features from other kinds of processing, you can make your component classes lean and efficient.


Performed by Angular Framework
object
@Injectable()
DataService
getData()
Services
C1
constructor(private serviceObj:DataService)
{....}
C2
constructor(private serviceObj:DataService)
{....}
C3
constructor(private serviceObj:DataService)
{....}

How to develop the Services?

Narasimha

Sr. IT Trainer/Consultant

Working with services

Steps

1. Create a service using angular CLI.
2. Add the required logic in service class
3. Provide the services using Providers
4. Inject the service object in component
5. Access the members of services to perform operation.

How to create services?

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

@Injectable({
     providedIn: 'root'
})
export class LogService
{
        log(msg: any) { console.log(msg); }
        error(msg: any) { console.error(msg); }
        warn(msg: any) { console.warn(msg); }
}
```

How to use services?

```
export class AppComponent
{
        constructor(private logger: LogService){ }

        button_click()
        {
                this.logger.log("Hello World");
        }

}
```


Practical
HandsOns


Q & A