HEXAWARE

ANGULAR – HTTP SERVICE

Session Objective

- To understand the following features of http service with hands-on experience in Angular .

  - HTTP Service
    - RXJS - Observables
    - Http Error Handling

 | © Hexaware Technologies. All rights reserved.

HTTP With RxJS OBSERVABLE

HTTP

- HTTP is available under the ‘@angular/http’ package.
- Import the package and inject it into the component/service.
- Get method is used to send an HTTP Request and subscribe to the response Asynchronously.
- The response is mapped to the desired object and result is displayed.

 | © Hexaware Technologies. All rights reserved.

Observables

- Observables help us manage asynchronous data such as data coming from a backend service.
- It handles multiple values over time makes them a good candidate for working with real-time data
- It treat events as a collection.
    - An array whose items arrive asynchronously over time.

 | © Hexaware Technologies. All rights reserved.

Observables (cont..)

- Observables are a proposed feature of ES2016, the next version of JavaScript.
- To use observables angular uses the third party library called Reactive Extensions(RxJS).
- Observables are used within Angular itself including Angular's event system and its HTTP client service.
- It allow us to manipulate sets of events with operator.

 | © Hexaware Technologies. All rights reserved.

Observable operators

- RxJS also provides Observable operators which can used to manipulate the data being emitted. Some of these operators are:
  - Map -Map operator allows to transform the incoming data.
  - Filter - emit only those items from an Observable that pass a predicate test
  - Take - emit only the first *n* items emitted by an Observable
  - Skip - suppress the first *n* items emitted by an Observable
  - Debounce -only emit an item from an Observable if a particular timespan has passed without it emitting another item

 | © Hexaware Technologies. All rights reserved.

Observables cont…

- **Filter** operator filters an Observable by only allowing items through that pass a test that you specify in the form of a predicate function.
- For example:

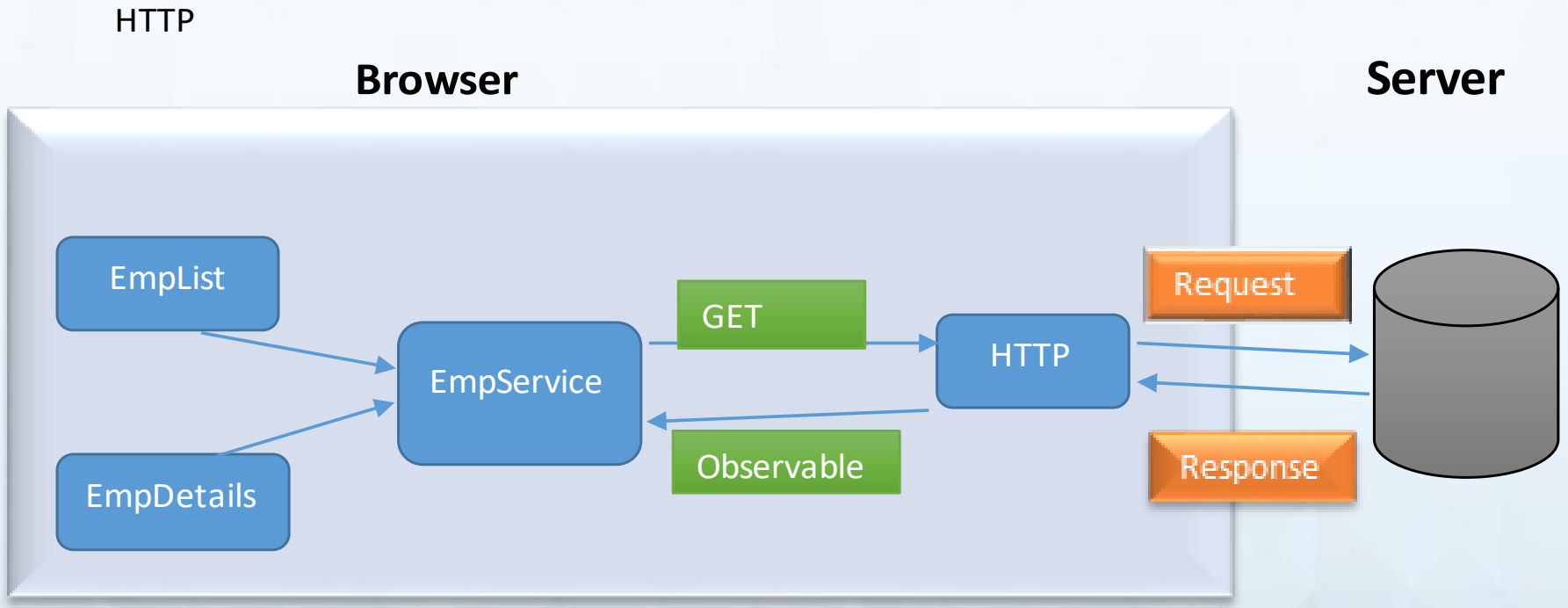
```
filter(x => x > 10)
```

- **Map** transform the items emitted by an Observable by applying a function to each item.
- For example:

map(x => 10*x) The argument to the arrow function that says to take each data item and transform it to 10 times its value.

 | © Hexaware Technologies. All rights reserved.

Observables cont…



 | © Hexaware Technologies. All rights reserved.

Observable - HTTP

- Angular HTTP library provides Http client for server communication.
- **get()** is the Http client method that uses HTTP GET method to communicate server using HTTP URL.
- Pass HTTP URL to Http.get() and it will return the instance of RxJS Observable. To listen values of Observable we need to subscribe it.

 | © Hexaware Technologies. All rights reserved.

Observable

Observable

- An observables emits multiple values over time.
- observables is lazy by default.(will not emit values until they subscribed to)
- Observable can be cancellable by unsubscribing. It supports map, filter, reduce and similar operators.

 | © Hexaware Technologies. All rights reserved.

Service Using HTTP and Observable

HEXAWARE

Import HTTP Module in Root Module

- Import it in the root module **app.module**
- Add it to the imports metadata array

```
import { HttpModule } from '@angular/http'
@NgModule({
  imports: [BrowserModule, HttpModule],
})
```

 | © Hexaware Technologies. All rights reserved.

Service Using HTTP and Observable cont…

Import the Required module in Component/Service

```
import { Http, Response } from '@angular/http';
import { Observable } from 'rxjs/Observable';
```

 | © Hexaware Technologies. All rights reserved.

Service Using HTTP and Observable cont…

Inject HTTP service into the component/service

```
constructor(private _http: Http) {    }
```

 | © Hexaware Technologies. All rights reserved.

Service Using HTTP and Observable cont…

Call the service method

```
ngOnInit() {
    this._employeeService.getEmp().subscribe((employeeData) =>
this.employee = employeeData);
    console.log(this.employee);
  }
```

 | © Hexaware Technologies. All rights reserved.

Service Using HTTP and Observable cont…

- The method getEmp () return Observable
- Subscribe is defined to retrieve the Observable, it also uses the callback function as argument to access the data
- The callback function is notified when Observable emits the array of IEmployee

 | © Hexaware Technologies. All rights reserved.

Service Using HTTP and Observable cont…

HEXAWARE

Subscribe Callback function

Callback Method	Purpose
onNext	The Observable calls this method whenever the Observable emits an item. The emitted item is passed as a parameter to this method
onError	The Observable calls this method if there is an error
onCompleted	The Observable calls this method after it has emitted all item. If it is calling onNext for the final time

 | © Hexaware Technologies. All rights reserved.

Service Using HTTP and Observable cont…

HEXAWARE

Call the HTTP Service

```
getEmp(): Observable<IEmployee[]> {
        return
this._http.get("https://jsonplaceholder.typicode.com/posts")
            .map((response: Response) =>
<IEmployee[]>response.json())
}
```

 | © Hexaware Technologies. All rights reserved.

Service Using HTTP and Observable cont…

- Get() method returns Observable of Response
- Transform the response to an array of IEmployee using map operator
- The map operator of observables is to extract the data of the response as JSON content.

 | © Hexaware Technologies. All rights reserved.

HTTP Error Handling

- Error handling in HTTP is done using catch operator
- Import the required
  - Import ‘rxjs/add/operator/catch’

 | © Hexaware Technologies. All rights reserved.

HTTP Error Handling cont…

Mark the exception in the **Service** class

```
getEmp(): Observable<IEmployee[]> {
        return
this._http.get("https://jsonplaceholder.typicode.com/postss")
            .map((response: Response) =>
<IEmployee[]>response.json())
            .catch(this.handleError);

    }
    handleError(error: Response) {
        console.error(error);
        return Observable.throw(error);
    }
```

 | © Hexaware Technologies. All rights reserved.

HTTP Error Handling cont…

Handle the exception in **Component** class

```
ngOnInit() {
    this._employeeService.getEmp()
      .subscribe((employeeData) => this.employee =
employeeData, (error) => {
        this.message = 'Service is not available Please try
later'
      });
}
```

 | © Hexaware Technologies. All rights reserved.

HTTP Error Handling cont…

Display the message using **HTML**

```
<tr *ngIf="!employee">
                {{message}}
            </tr>
```

 | © Hexaware Technologies. All rights reserved.

HEXAWARE

Innovative Services

Passionate Employees

Delighted Customers

Thank you

www.hexaware.com