Services and Dependency Injection

Deborah Kurata
Consultant | Speaker | Author | MVP | GDE

@deborahkurata


Products
Logging

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**

Module Overview

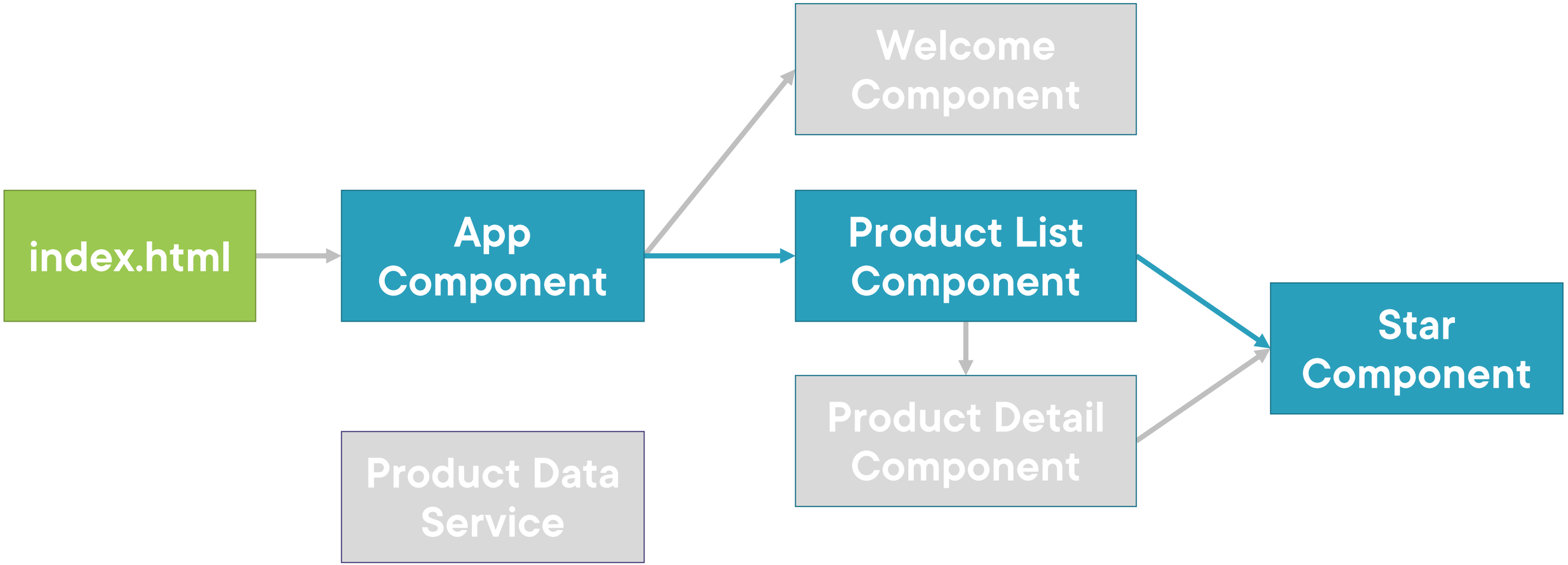
How does it work?

Building a service

Registering the service

Injecting the service

Application Architecture



How Does It Work?

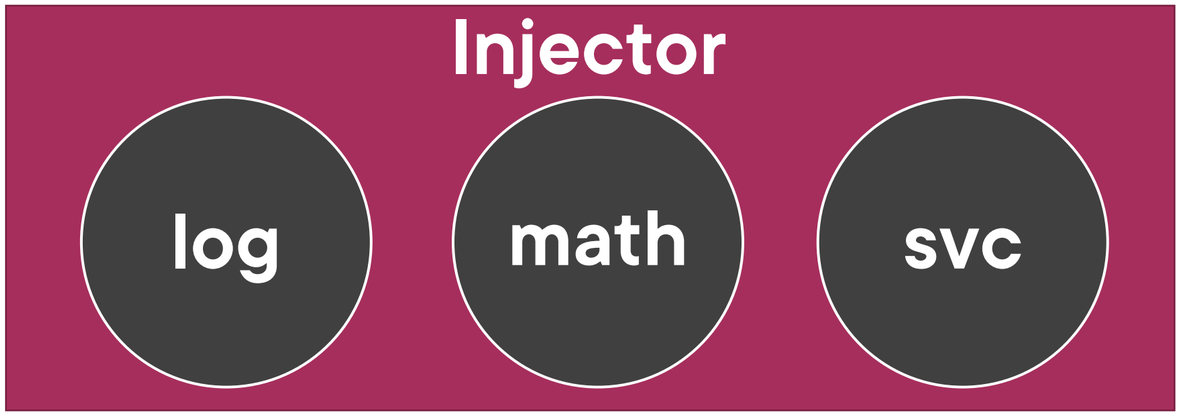
Service

```
export class myService {}
```

Component

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

How Does It Work?



Service

```
export class myService {}
```

Component

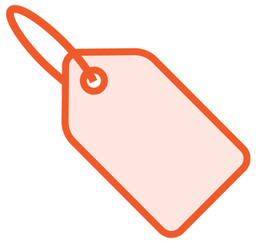
```
constructor(private myService) {}
```

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

Create the service class



Define the metadata with a decorator

Import what we need

Building a Service

product.service.ts

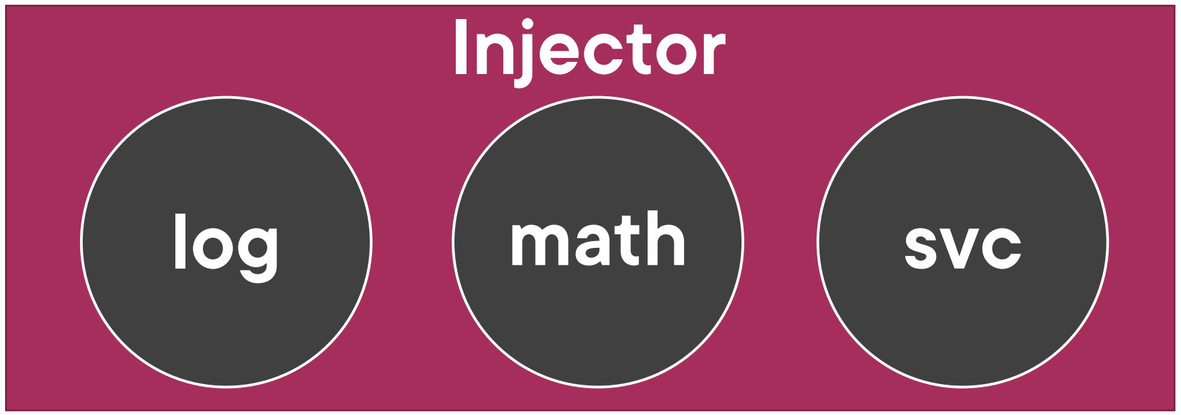
```
import { Injectable } from '@angular/core'

@Injectable()
export class ProductService {

  getProducts(): IProduct[] {
  }

}
```

Registering a Service



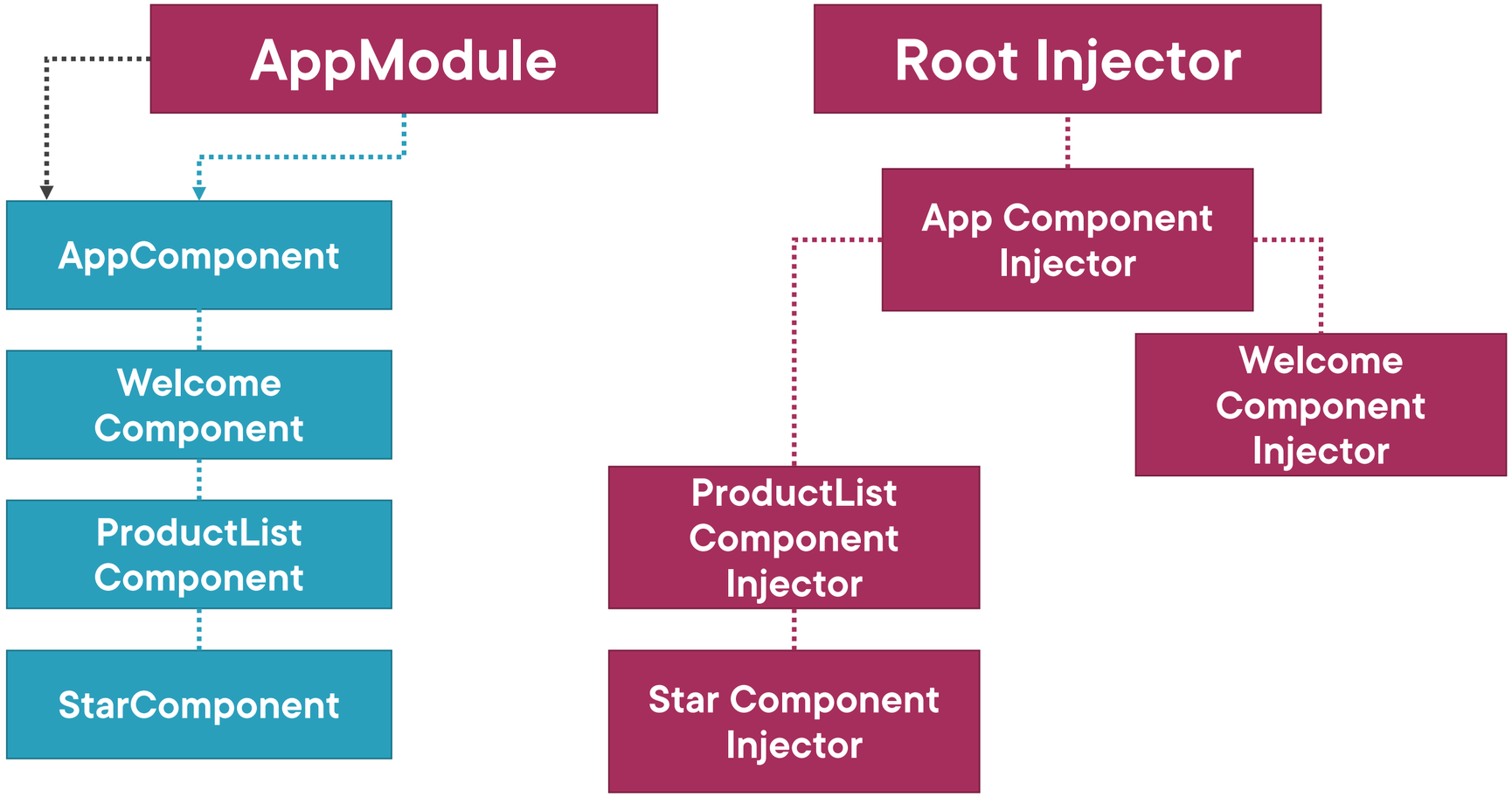
Service

```
export class myService {}
```

Component

```
constructor(private myService) {}
```

Angular Injectors



Registering a Service

Root Injector

Service is available throughout the application

Recommended for most scenarios

Component Injector

Service is available ONLY to that component and its child (nested) components

Isolates a service used by only one component

Provides multiple instances of the service

Registering a Service - Root Application

product.service.ts

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

@Injectable({
  providedIn: 'root'
})
export class ProductService {

  getProducts(): IProduct[] {
  }

}
```

product.service.ts

```
@Injectable({
  providedIn: 'root'
})
export class ProductService { }
```

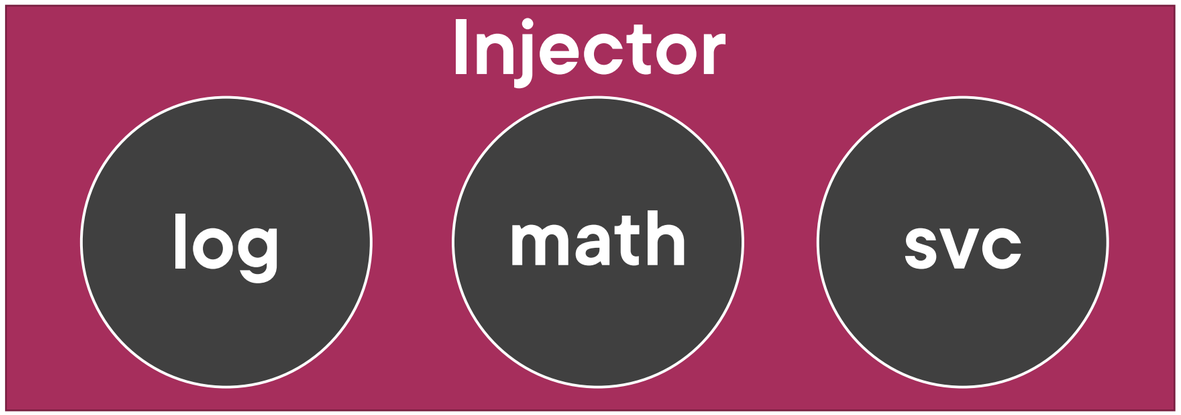
product-list.component.ts

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

app.module.ts

```
@NgModule({
  imports: [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ],
  providers: [ProductService]
})
export class AppModule { }
```

Injecting the Service



Service

```
@Injectable({
  providedIn: 'root'
})
export class myService {}
```

Component

```
constructor(private myService) {}
```

Injecting the Service

product-list.component.ts

```
...


@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {

 constructor() {
 }

}
```

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) {
   this._productService = productService;
 }

}
```

Injecting the Service

product-list.component.ts

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

@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {

 constructor(private productService: ProductService) { }

}
```

Service Checklist: Building a 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

```
import { Injectable } from '@angular/core';
@Injectable({
  providedIn: 'root'
})
export class ProductService {...}
```

Service Checklist: Registering a Service

Select the appropriate level in the hierarchy

- Root application injector if the service is used throughout the application
- Specific component's injector if only that component uses the service

Service Injectable decorator

- Set the `providedIn` property to `'root'`

```
@Injectable({
  providedIn: 'root'
})
export class ProductService {...}
```

Component decorator

- Set the `providers` property to the service

Service Checklist: Dependency Injection

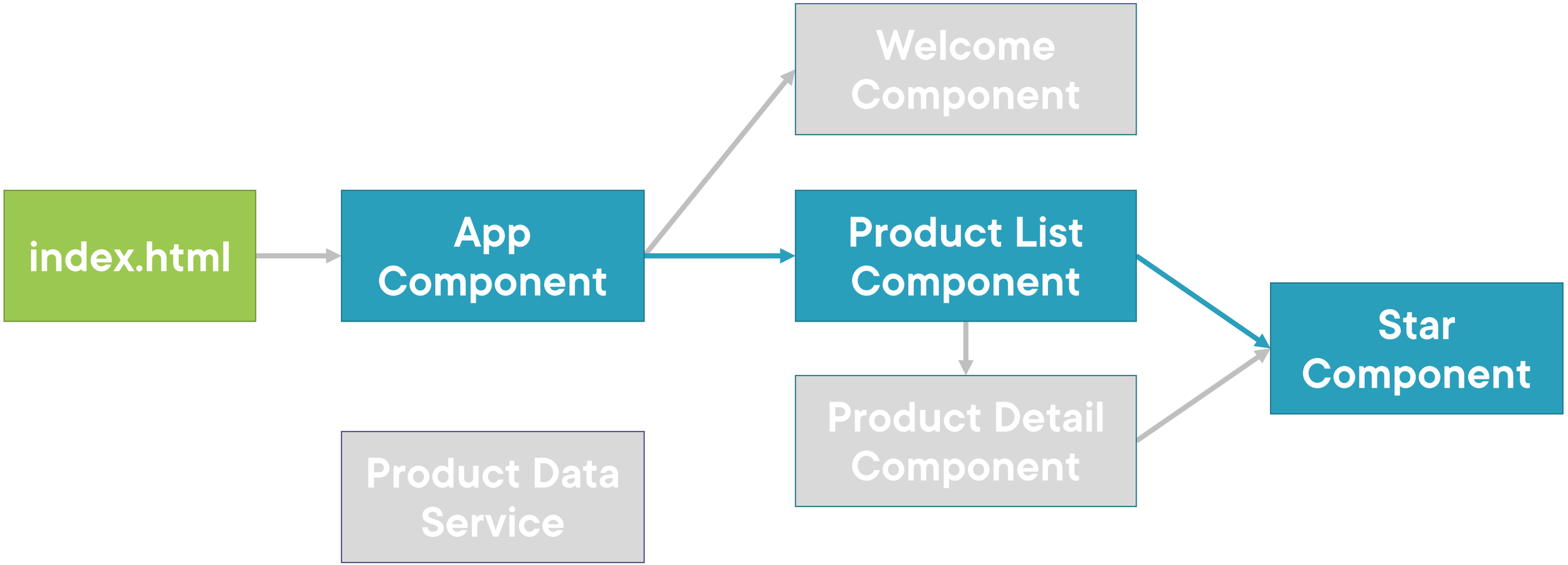
Specify the service as a dependency

Use a constructor parameter

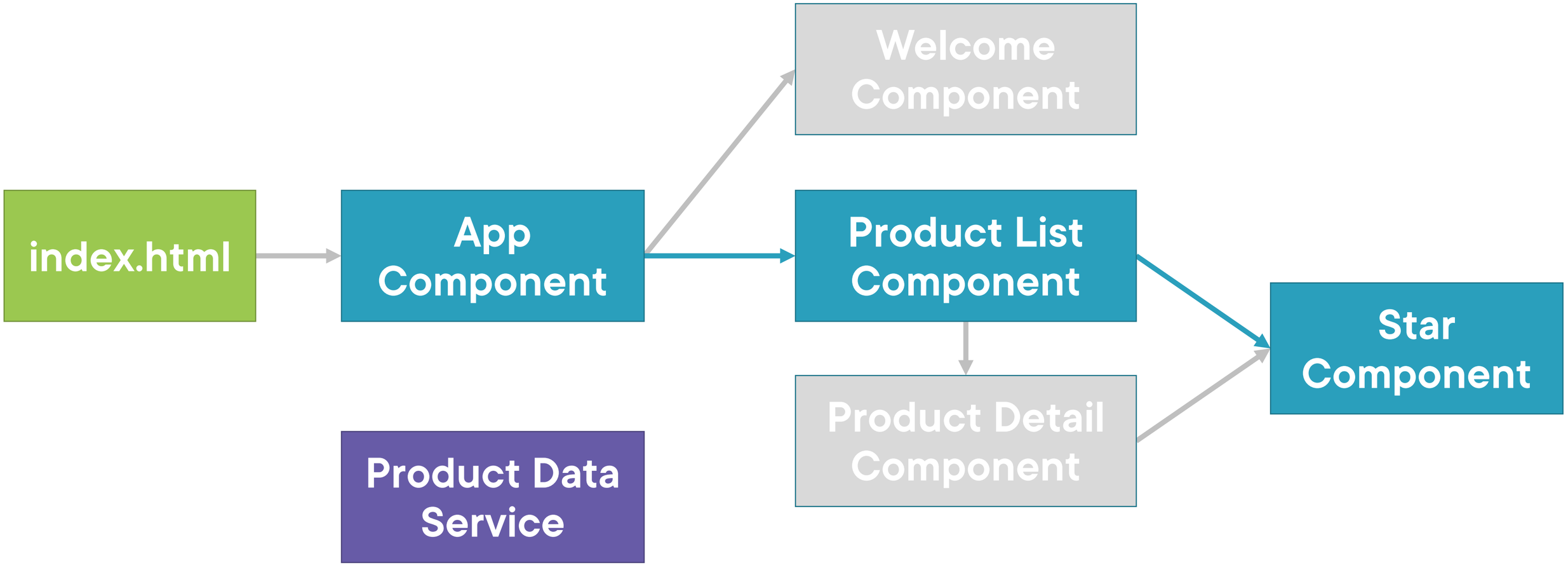
Service is injected when component is instantiated

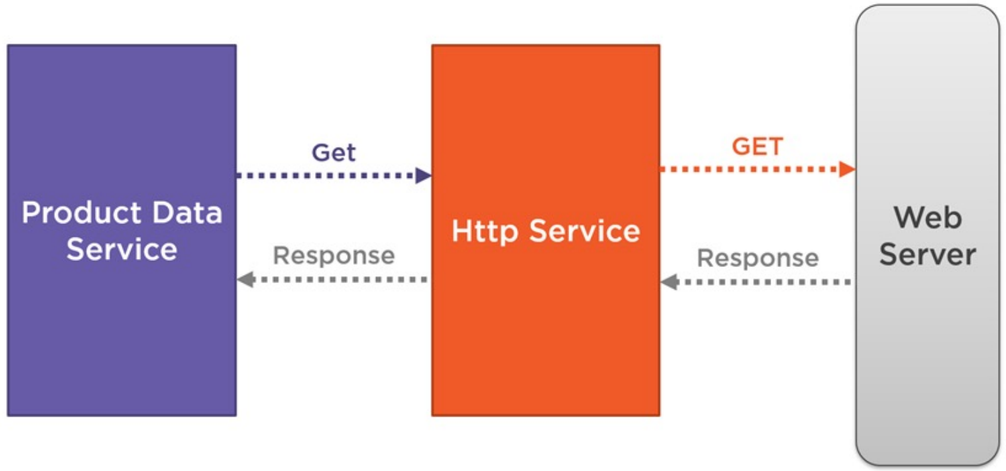
```
constructor(private productService: ProductService) { }
```

Application Architecture



Application Architecture





Coming up next ...

Retrieving Data Using HTTP