Сервисы

№ урока:	1	**Курс:**	**Angular 2 Essential**

Средства обучения:	Редактор кода Node.js и npm

Обзор, цель и назначение урока

Цель урока – изучить принцип внедрения зависимостей в Angular 2 и научиться создавать сервисы.

Содержание урока

1. Что такое внедрение зависимостей.
2. Простой сервис
3. Injector Tree
4. Конфигурация провайдеров
5. Использование opaque token

Резюме

Внедрение зависимости (англ. Dependency injection, DI) — процесс предоставления внешней зависимости программному компоненту. Является специфичной формой «инверсии управления» (англ. Inversion of control, IoC). В полном соответствии с принципом единой обязанности объект отдаёт заботу о построении требуемых ему зависимостей внешнему, специально предназначенному для этого общему механизму.

Проблема кода, в котором не используется Dependency Injection

```
export class Car {
   private engine: Engine;
   private tires: Tires;
      constructor() {
      this.engine = new Engine();
      this.tires = new Tires();
   }

   drive() {
      return `Автомобиль с двигателем ${this.engine.cylinders} цилиндров и резиной
${this.tires.name}`;
   }
}
```

Проблема данного класса в том, что он хрупкий, не гибкий, тяжело расширяется и не поддается тестированию. Если измениться конструктор класса Engine или Tires (например, нужно будет указать дополнительные параметры) то нарушиться работа класса Car. Это делает класс хрупким. Если мы захотим поместить другой тип резины в свойство tires, то столкнемся с проблемой, связанной с тем, что тип шин заранее определен конструктором и поменять его тяжело. Это делает класс негибким. Если мы захотим использовать один и тот же экземпляр зависимостей, для разных экземпляров автомобилей в текущей версии конструктора — это невозможно будет сделать. Данному классу не хватает гибкости.
Для данного класса тяжело написать Unit тест, так как нет контроля над зависимостями. Нет возможности контролировать какие значения будут возвращаться экземплярами классов Engine и Tires.

CyberBionic systematics

CyberBionic Systematics ®
19 Yevgena Sverstuka Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Один из вариантов решения проблемы убрать создание зависимостей из кода класса и передавать их через конструктор, метод или свойство.

В следующем классе, зависимости передаются через конструктор. Это дает возможность определить какие экземпляры зависимостей должны использоваться в конкретный момент времени.

```
export class Car {

    constructor(private engine: Engine, private tires: Tires) {
    }

    drive() {
        return `Автомобиль с двигателем ${this.engine.cylinders} цилиндров и резиной
${this.tires.name}`;
    }
}
let c1 = new Car(new Engine(), new Tires());
let c2 = new Car(new Engine2(), new Tires());
```

Желтим цветом выделен код выполняющий внедрение зависимостей. Проблемой данного кода является то что созданием экземпляров зависимостей занимается клиент, которому необходимо получить экземпляр класса Car. Есть разные фреймворки для внедрения зависимостей. В этом уроке будет рассмотрен механизм внедрения зависимостей в Angular 2.

@Injectable() декоратор, которые определяет что класс, является доступным для создание через Injector. При создании сервисов, класс сервиса должен быть декорирован @Injectable()

Сервис – объект выполняющий общие для всего приложения задачи. Например, отправка запроса на сервер, логирование, проверка данных, проверка прав доступа и т.д.

Пример сервиса в Angular 2

```
@Injectable()
export class DataService {

    getData() {
        return “shared information”;
    }

}
```

Для того чтобы сервис можно было использовать его нужно зарегистрировать. Для этого нужно указать инжектору (компоненту системы, который занимается созданием сервисов) провайдер, который создает экземпляр сервиса. Для того, чтобы установить провайдер, нужно указать значение свойства providers для модуля, компонента или другой части приложения. Если провайдер будет установлен для модуля, то экземпляр сервиса будет доступен всем объектам, которые находятся в модуле и этот экземпляр будет общим для всех. Если провайдер установить в компоненте, то сервис будет доступен данному компоненту и его дочерним компонентам.

```
@NgModule({
    imports: [CommonModule],
    declarations: [DataListComponent],
    providers: [DataService]
})
export class SimpleServiceModule {
```


CyberBionic Systematics ®
19 Yevgena Sverstuka Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

```
}
```

Для того, чтобы получить экземпляр сервиса, который будет создан провайдером, нужно определить конструктор с особыми параметрами, в том классе, который требует внедрения зависимостей. В нашем примере выше сервис называется DataService. Для того, чтобы получить доступ к экземпляру этого сервиса, в компоненте, который находиться в модуле SimpleServiceModule, необходимо написать следующий код.

```
export class DataListComponent {

   constructor(private dataService: DataService) {
   }

}
```

Injector используя конструктора класса DataService создаст экземпляр и поместит ссылку на него в параметр конструктора DataListComponent.

Закрепление материала

- Что такое внедрение зависимости?
- Что такое сервис?
- Как создать сервис в Angular 2 приложении?
- Для чего нужно использовать декоратора @Injectable()?
- Как получить экземпляр необходимого сервсиа?

Самостоятельная деятельность учащегося

Задание 1
Проведите рефакторинг кода, сделанного по любому заданию из урока №3. Перепишите код таким образом, чтобы значения массива products предоставлялись компоненту через сервис.

Задание 2
Создайте объект

```
let data = {
        getData: function() {
                // вернуть массив products
        }
}
```

Измените код первого задания так, чтобы он использовал объект data в качестве сервиса.

Рекомендуемые ресурсы

Dependency Injection in Angular2
https://angular.io/docs/ts/latest/guide/dependency-injection.html

Hierarchical dependency injectors
https://angular.io/docs/ts/latest/guide/hierarchical-dependency-injection.html

CyberBionic systematics

CyberBionic Systematics ®
19 Yevgena Sverstuka Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com