

---

---

# Dependency Injection

— SED 2019 Team 2 —  
11/14

---

---

# Dependency Injection

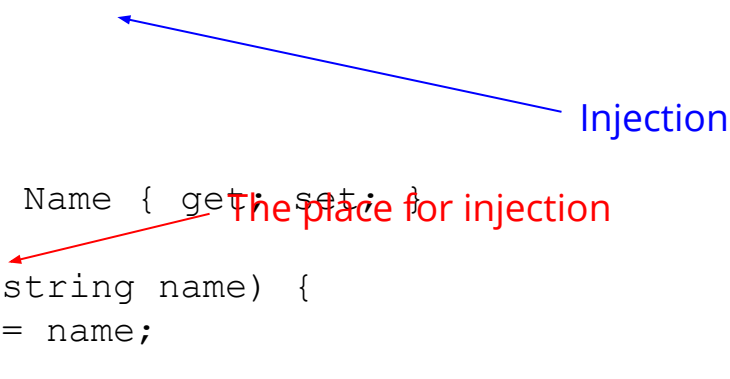
- To decouple dependencies and dependents
- For easily replacing dependencies without changing lots of classes
- Achieve **Inversion Of Control** (IoC): dependencies are not actively created, but passively received
- Three approaches of dependency injection
  - Constructor Injection
  - Method Injection
  - Property Injection
- Dependency Injection Containers (Injectors) to further simplify dependency injection

# Three Approaches of Dependency Injection

```
// Constructor Injection
public class Program{
    static void Main(string[] args) {
        Person p = new Person("Roberson");
    }
}

public class Person {
    private string Name { get; set; }

    public Person(string name) {
        this.Name = name;
    }
}
```



Injection

The place for injection

# Three Approaches of Dependency Injection

// Method Injection

```
public class Program {  
    static void Main(string[] args) {  
        Person p = new Person();  
        p.SetName("Roberson");  
    }  
}
```

Injection




```
public class Person {  
    private string Name { get; set; }  
  
    public void SetName(string name) {  
        this.Name = name;  
    }  
}
```

The place for injection




# Three Approaches of Dependency Injection

```
// Property Injection
public class Program {
    static void Main(string[] args) {
        Person p = new Person();
        p.Name = "Roberson";
    }
}
```



Injection

```
public class Person {
    public string Name { get; set; }
}
```



The place for injection

# Dependency Injection Containers (Injectors)

- Solve the problem that dependency classes are still referenced
- Provides a central registry of dependencies
- Example

```
Container.register('Auth', 'DbAuth', new Credentials('mysql://localhost',  
'user', 'pass'));
```

```
Auth auth = Container.get('Auth');
```

```
Session session = new Session();
```

```
App app = new App(auth, session);
```

# Dependency Injection in web2py

```
def flatten(self, render=None): # gluon/html.py
```

```
    """returns the text stored by the XML object rendered by the `render`  
    function"""
```

```
    if render:
```

The place for injection



```
        return render(self.text, None, {})
```

```
    return self.text
```

Injection



```
markmin = TAG(html).element('body').flatten(markmin_serializer) #  
gluon/contrib/generics.py
```

# References

- [鐵人賽Day08] - Dependency Injection概念介紹  
<https://ithelp.ithome.com.tw/articles/10204404>
- 理解 Dependency Injection 實作原理  
<https://jaceju.net/php-di-container/>
- Learn Dependency Injection By Building an Injector  
<https://itnext.io/learn-dependency-injection-by-building-an-injector-fb48408af6a>
- 控制反轉 (IoC) 與 依賴注入 (DI)  
<https://notfalse.net/3/ioc-di>