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");
    }
}

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

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

Injection

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";
    }
}

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

Injection

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:

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

    return self.text
```

The place for injection

Injection

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

Dependency Injection Containers in web2py

```python
class Dispatcher(object):
    namespace = "dispatcher"

    def __init__(self, namespace=None):
        self._registry_ = {}
        if namespace:
            self.namespace = namespace

    def register_for(self, target):
        def wrap(dispatch_class):
            self._registry_[target] = dispatch_class
            return dispatch_class
        return wrap

    def get_for(self, obj):
        targets = type(obj).__mro__
        for target in targets:
            if target in self._registry_:
                return self._registry_[target](obj)
        else:
            raise ValueError(
                "no %s found for object: %s" % (self.namespace, obj))
```

```
# Create a container
parsers = Dispatcher("parser")

# register it
@parsers.register_for(Postgre)
class PostgreParser(...

# use it
def _load_dependencies(self):
  self.parser = parsers.get_for(self)
```

Are Controllers Managed by DI?

- Rough workflow for using controllers
  - Find the controller by parsing the URL
  - Compile the controller as *.pyc
  - Load the pyc file
- Filesystem path as the key?

```
cpath = pjoin(folder, 'compiled')
if os.path.exists(cpath):
    filename = pjoin(cpath, 'controllers.%s.%s.pyc' % (controller, function))
    try:
        ccode = getcfs(filename, filename, lambda: read_pyc(filename))
    except IOError:
        raise HTTP(404,
                   rewrite.THREAD_LOCAL.routes.error_message % badf,
                   web2py_error=badf)
```

Dependency Injection Containers in web2py

- Registration of dependencies

```
@adapters.register_for('sqlite', 'sqlite:memory')

class SQLite(SQLAdapter):
```

- Usage of dependencies

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