Dependency Injection is your friend

Alan Franzoni @ EP2011
Twitter: @franzeur
Tech blog: http://ollivander.franzoni.eu
e-mail: public@franzoni.eu

SOLID

S	O	L	I	D
Single Responsibility	Open-Closed	Liskov Substitution	Interface Segregation	Dependency Inversion

100644 | 908 lines (774 sloc) | 35.621 kb raw | blame | history

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Copyright 2005-2006 Open Source Applications Foundation

    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

        http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License.
-->

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:jee="http://www.springframework.org/schema/jee"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:security="http://www.springframework.org/schema/security"
    xsi:schemaLocation="
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.0.xsd
    http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.0.xsd
    http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-2.0.xsd
    http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security-2.0.xsd">

  <bean id="messageSource"
        class="org.springframework.context.support.ResourceBundleMessageSource">
    <property name="basename" value="MessageResources"/>
  </bean>
```

```
35
36    <!-- load the cosmo properties file and make the cosmo
37         config properties available to other spring beans -->
38    <jee:jndi-lookup id="cosmoConfigPath" jndi-name="java:comp/env/cosmo/config"/>
39
40    <bean id="propertyPlaceholderConfigurer"
41          class="org.osaf.cosmo.spring.CosmoPropertyPlaceholderConfigurer">
42      <property name="locations">
43        <list>
44          <ref local="cosmoConfigPath"/>
45        </list>
46      </property>
47    </bean>
48
49    <!-- resources needed to send mail -->
50    <jee:jndi-lookup id="mailSession" jndi-name="java:comp/env/mail/cosmo"/>
51
52    <bean id="mailSender"
53          class="org.springframework.mail.javamail.JavaMailSenderImpl">
54      <property name="session">
55        <ref local="mailSession"/>
56      </property>
57    </bean>
58
59    <!-- resources needed for database access -->
60    <jee:jndi-lookup id="jdbcDataSource" jndi-name="java:comp/env/jdbc/cosmo"/>
61
62    <bean id="defaultLobHandler"
63          class="org.osaf.cosmo.hibernate.CosmoLobHandler">
64    </bean>
65
66    <bean id="validatePreUpdateEventListener"
67        class="org.hibernate.validator.event.ValidatePreUpdateEventListener"/>
68
69    <bean id="validatePreInsertEventListener"
70        class="org.hibernate.validator.event.ValidatePreInsertEventListener"/>
71
72
73    <bean id="auditableObjectInterceptor"
74        class="org.osaf.cosmo.model.hibernate.AuditableObjectInterceptor"/>
75
76    <bean id="eventStampInterceptor"
```

```
78
79  <bean id="cosmoHibernateInterceptor"
80      class="org.osaf.cosmo.hibernate.CompoundInterceptor">
81      <property name="interceptors">
82        <list>
83            <ref local="auditableObjectInterceptor"/>
84            <ref local="eventStampInterceptor"/>
85        </list>
86    </property>
87  </bean>
88
89  <bean id="cosmoEntityFactory"
90      class="org.osaf.cosmo.model.hibernate.HibEntityFactory"/>
91
92  <bean id="sessionFactory"
93        class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
94    <property name="lobHandler" ref="defaultLobHandler" />
95    <property name="annotatedPackages">
96        <list>
97            <value>org.osaf.cosmo.model.hibernate</value>
98        </list>
99    </property>
100   <property name="annotatedClasses">
101       <list>
102           <value>org.osaf.cosmo.model.hibernate.HibAttribute</value>
103           <value>org.osaf.cosmo.model.hibernate.HibAttributeTombstone</value>
104           <value>org.osaf.cosmo.model.hibernate.HibAvailabilityItem</value>
105           <value>org.osaf.cosmo.model.hibernate.HibBaseEventStamp</value>
106           <value>org.osaf.cosmo.model.hibernate.HibBinaryAttribute</value>
107           <value>org.osaf.cosmo.model.hibernate.HibBooleanAttribute</value>
108           <value>org.osaf.cosmo.model.hibernate.HibCalendarAttribute</value>
109           <value>org.osaf.cosmo.model.hibernate.HibCalendarCollectionStamp</value>
110           <value>org.osaf.cosmo.model.hibernate.HibCollectionItem</value>
111           <value>org.osaf.cosmo.model.hibernate.HibCollectionItemDetails</value>
112           <value>org.osaf.cosmo.model.hibernate.HibCollectionSubscription</value>
113           <value>org.osaf.cosmo.model.hibernate.HibContentData</value>
114           <value>org.osaf.cosmo.model.hibernate.HibContentItem</value>
115           <value>org.osaf.cosmo.model.hibernate.HibDecimalAttribute</value>
116           <value>org.osaf.cosmo.model.hibernate.HibDictionaryAttribute</value>
117           <value>org.osaf.cosmo.model.hibernate.HibEventExceptionStamp</value>
118           <value>org.osaf.cosmo.model.hibernate.HibEventLogEntry</value>
```

```
869     <bean id="protocolExtraTicketProcessingFilter"
870           class="org.osaf.cosmo.acegisecurity.providers.ticket.ExtraTicketProcessingFilter">
871       <property name="securityManager">
872         <ref local="securityManager" />
873       </property>
874       <property name="contentDao">
875         <ref local="contentDao" />
876       </property>
877     </bean>
878
879     <bean id="protocolBasicProcessingFilter"
880           class="org.springframework.security.ui.basicauth.BasicProcessingFilter">
881       <property name="authenticationManager" ref="authenticationManager"/>
882       <property name="authenticationEntryPoint" ref="protocolAuthenticationEntryPoint"/>
883     </bean>
884
885     <bean id="protocolExceptionTranslationFilter"
886           class="org.springframework.security.ui.ExceptionTranslationFilter">
887       <property name="authenticationEntryPoint" ref="protocolAuthenticationEntryPoint"/>
888       <property name="createSessionAllowed" value="false"/>
889     </bean>
890
891     <bean id="protocolWsseProcessingFilter"
892           class="org.osaf.cosmo.acegisecurity.providers.wsse.WsseTokenProcessingFilter"/>
893
894     <!-- Logging resources -->
895     <bean id="httpLoggingFormat" class="java.lang.String">
896       <constructor-arg type="java.lang.String" value="${cosmo.log.HttpLoggingFilter.format}"/>
897     </bean>
898
899     <!-- give JSP tag functions access to config properties -->
900     <bean id="jspConfigProps" class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
901       <property name="targetClass" value="org.osaf.cosmo.ui.TagFunctions"/>
902       <property name="targetMethod" value="setPropertyPlaceholderConfigurer"/>
903       <property name="arguments">
904           <list><ref bean="propertyPlaceholderConfigurer"/></list>
905       </property>
906     </bean>
907 </beans>
908
```

(non-xml DI with Spring)

```
@Configuration
public class AppConfig {
    @Bean
    public MyService myService() {
        return new MyServiceImpl();
    }
}
```

Peer vs Implementation Detail

Peer (injectable)	Usually addresses a different concern from the main one.	Almost anything with a different responsibility
Implementation detail (newable)	Internally employed to perform some task.	Most data-only structures, transformation tools, etc.

Hardcoded dependency

```
from somepackage import UserRepository
from otherpackage import my_db_driver

class UserService(object):
    def __init__(self):
        self.user_repo = UserRepository(
                                my_db_driver)
```

Constructor Injection

```
class UserService(object):
    def __init__(self, user_repository):
        self.user_repository = user_repository
```

Myth 1

"Python needs no stinkin' dependency injection."

Myth 2

"It makes your interfaces dirty and breaks encapsulation."

```
public class SimpleMovieLister {
    // the SimpleMovieLister has a dependency on the
    MovieFinder
    private MovieFinder movieFinder;
    // a setter method so that the Spring container can
    'inject' a MovieFinder
        public void setMovieFinder
        (MovieFindermovieFinder) {
            this.movieFinder = movieFinder;
            }
}
```

What's an interface?

"An interface is a device or system that unrelated entities use to interact."

Car cockpit

Driveable

```
from abc import ABCMeta

class Driveable(object):
    __metaclass__ = ABCMeta
    def steerLeft(self):
        pass
    def steerRight(self):
        pass
```

MySuperCar

```
class MySuperCar(object):
    def turnLeft(self):
        """impl."""
    def turnRight(self):
        """impl."""
    def makeBaconAndEggs(self):
        """impl."""
```

Constructors and interfaces

```
class FilesystemUserRepository(object):
    def __init__(self, base_directory,
                 serializer, deserializer):
        """impl."""
    def save(self, user):
        """impl."""
    def load_by_id(self, user_id):
        """impl."""

class DBBasedUserRepository(object):
    def __init__(self, db_driver):
        """impl."""
    def save(self, user):
        """impl."""
    def load_by_id(self, user_id):
        """impl."""
```

ISO Encapsulation

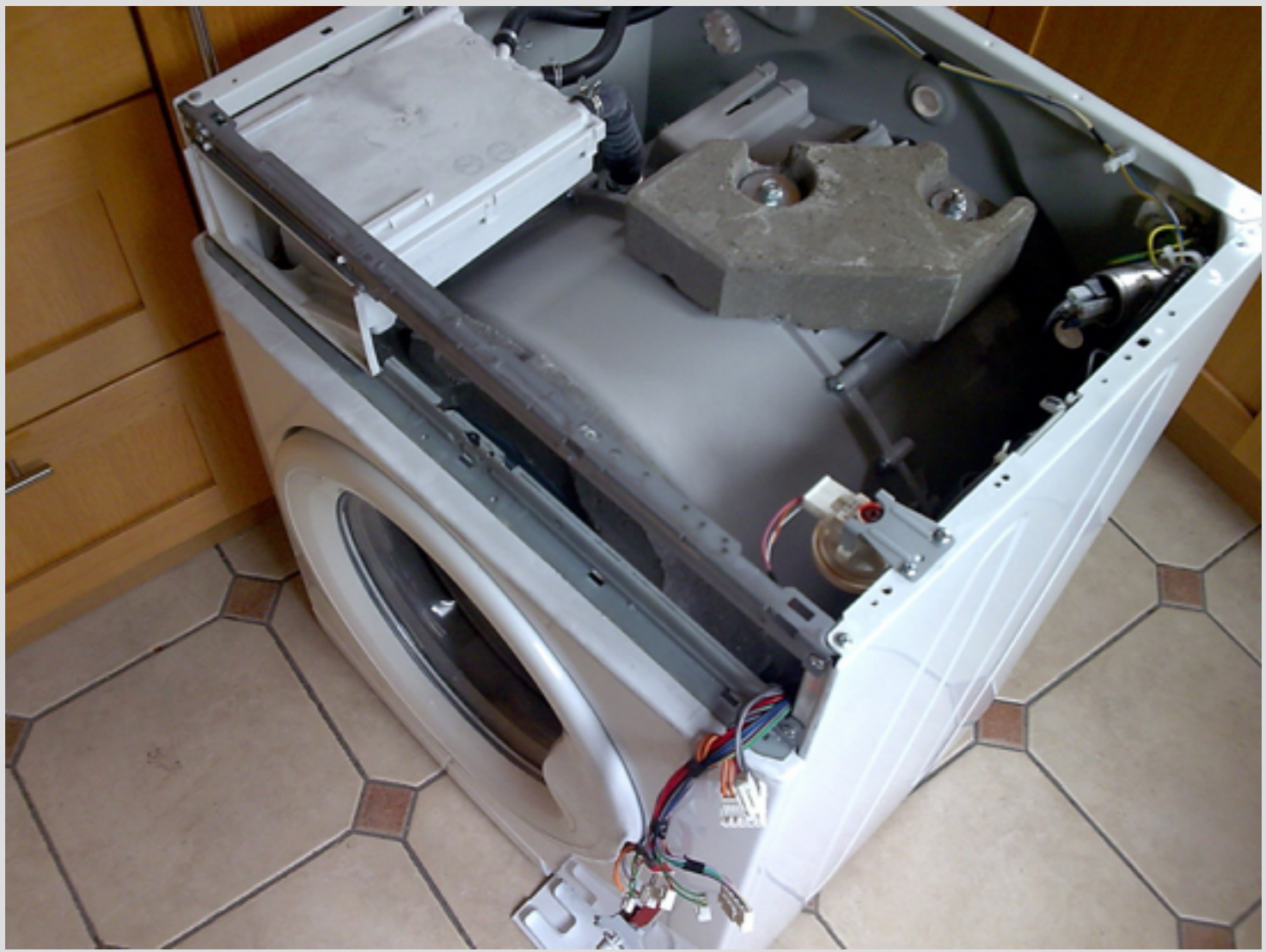
"Encapsulation is the property that the information contained in an object is accessible only through interactions at the interfaces supported by the object."

A proposal

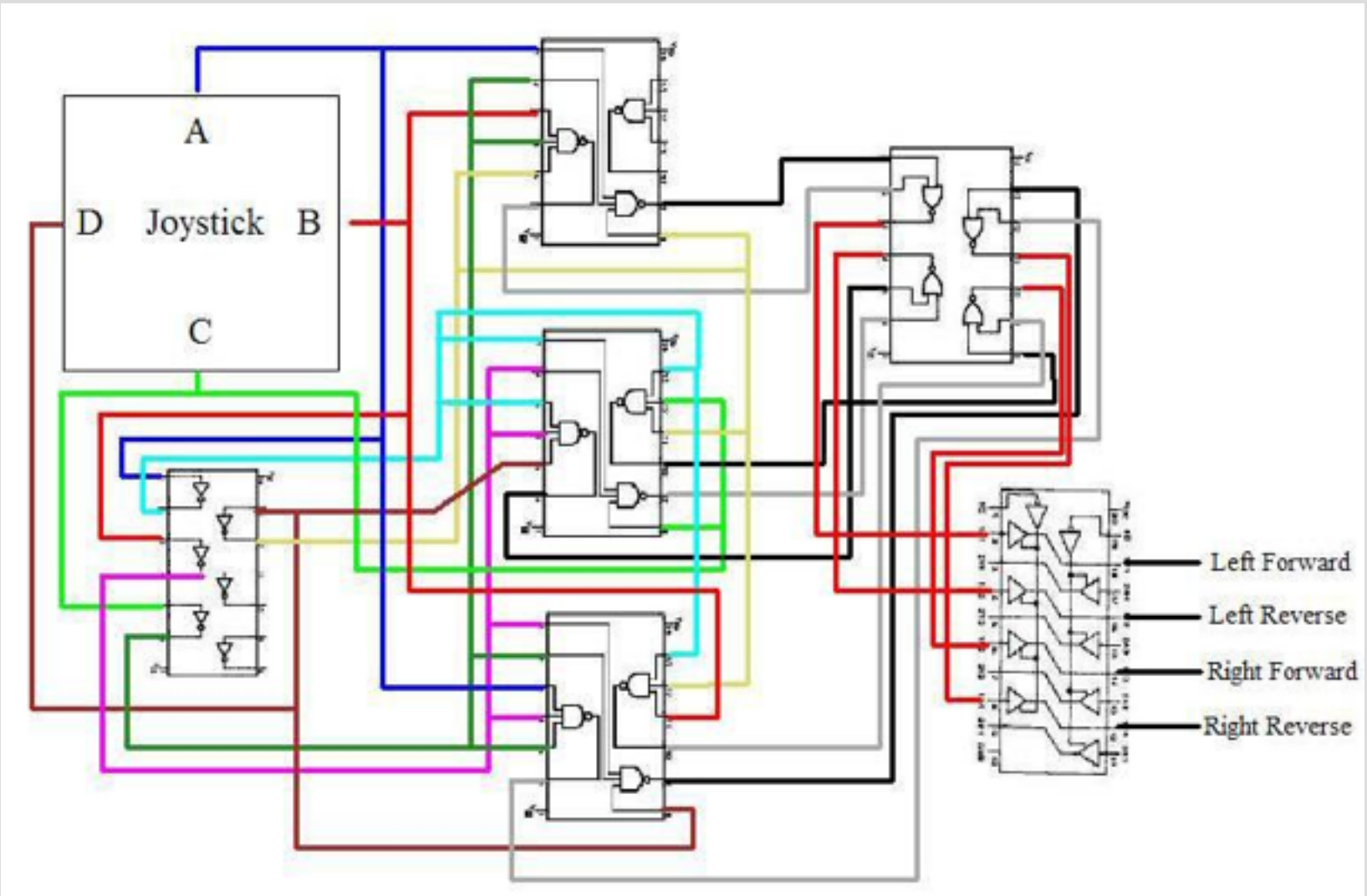
"Encapsulation is the property of a system to work by relying on its interfaces only."
(via @rferranti)

Encapsulated.

Not encapsulated?



Wiring


A
D
Joystick
B
C
Left Forward
Left Reverse
Right Forward
Right Reverse

Pydenji config example

```
from pydenji.config.pythonconfig import Configuration
from pydenji.config.provider import provider, singleton, prototype
from pydenji.importer import NI

class UserConfig(Configuration):
    @provider(scope=singleton, lazy_init=True)
    def user_service(self):
        return NI("ep2011.users.UserService")(
            self._user_repository())
    @provider()
    def _user_repository(self):
        db_driver = self.appcontext.provide(
            "db_driver")
        repo = NI("ep2011.users.UserRepository")(
            db_driver)
        return repo
    def set_app_context(self, appcontext):
        self.appcontext = appcontext
```

Appcontext usage

```
from pydenji.appcontext.context import AppContext
from ep2011.pydenji_conf import UserConfig

appcontext = AppContext()
appcontext.register("db_driver", lambda:
"stub_db_driver")
appcontext.register_anonymous(UserConfig)

appcontext.start()

us = appcontext.provide("user_service")
assert us is appcontext.provide("user_service")
print user_service
```

```
<ep2011.users.UserService object at 0x100655310>
```

Myth 3:

“Just monkey patch.”

Original code

```
class SomeClassWithDefaultDir(object):
    base_path = "/some/dir/"

    def do_something(self):
        with open(self.base_path +
                  "somefilename", "w") as f:
            f.write("something")
```

Unit test

```
@patch("ep2011.monkeypatching.open",
       create=True)
def test_dstn(self, mock_open):
    persist = open(self.tmpfilename, "w")
    mock_open.return_value = persist
    sc = SomeClassWithDefaultDir()
    sc.do_something()
    self.assertEquals("something",
        open(self.tmpfilename).read())
```

Refactor

```
from t.p.fp import FilePath
class SomeClassWFilePath(object):
    _root = FilePath("/some/dir")

    def do_something(self):
        with self._root.child(
            "somefilename").open("w") as f:
            f.write("something")
```

Test fails

```
@patch("ep2011.monkeypatching.open",
       create=True)
def test_dstn(self, mock_open):
    persist = open(self.tmpfilename,
                   "w")
    mock_open.return_value = persist
    sc = SomeClassWFilePath()
    sc.do_something()
    self.assertEquals("something",
        open(self.tmpfilename
             ).read())
```

IOError: [Errno 2] No such file or directory: '/some/dir/somefilename'

Inject

```
class SomeClassInjected(object):
    def __init__(self, resource):
        self.resource = resource

    def do_something(self):
        with self.resource as f:
            f.write(something)
```

Test with injection

```
def test_inj(self):
    f = open(self.tmpfilename, "w")
    s = SomeClassWithInjectedResource(
                    f)
    s.do_something()
    self.assertEquals("something",
            open(self.tmpfilename
                ).read())
```

Pydenji lazy resource

- Same interface as file object (actually a decorator)
- performs existence and permission check at instantiation time
- file is actually opened only when required
- maybe it's not the best idea (not-sensed methods?)

Pydenji

Just used for internal projects so far.
Sources available at http://pydenji.franzoni.eu
Give me some feedback AND user stories (web development anyone?)

References

martinfowler.com/articles/injection.html

misko.hevery.com/2008/09/30/to-new-or-not-to-new/

citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.2927&rep=rep1&type=pdf

www.objectmentor.com/resources/publishedArticles.html

Questions?

Slides, references and examples will be available shortly at
http://ep2011.franzoni.eu