Designing With Dependency Injection

Alex Miller
MetaMatrix
Overview

- Definition
- Types
- Consequences
- Patterns / Smells
- Architecture
- Q&A
Definition

● Other terms
  – Hollywood principle

● My definition
  – Technique to reduce coupling by moving configuration and dependency wiring outside a component

● Configuration as dependency
Types

- Service locator
- Interface injection (Type 1)
- Setter injection (Type 2)
- Constructor injection (Type 3)
- Getter injection
- Context IOC?
Service locator

- A facility to bind and look up components by name or type
- Not dependency injection (component still controls the wiring)
- Example: JNDI
Interface Injection (Type 1)

- Define interface used to inject a dependency
- Container: Apache Avalon
Setter Injection
(Type 2)

- All dependencies declared using setter methods
- Probably the most common type
- Containers: Spring, Pico
Constructor Injection (Type 3)

- Declare dependencies completely in one or more constructors (if different possible sets)
- Next most common after Type 2
- Containers: Pico, Spring
Getter Injection

- Specify a getter to be used within the class to retrieve the dependency
- Use AOP to inject a dependency by changing the implementation of the getter OR subclass and override the method
- Example: proposed by Bob Lee and prototyped using dynaop
Context IOC

- Proposed this week on TheServerSide.com
- Component defines an inner interface with getters for all dependencies
- Constructor takes an instance of this interface

```
Component
- Contract contract
+ Component(Contract contract)

Component.Contract
+ getFoo() : Foo
+ getBar() : Bar
```
Consequences

- Flexible
- Testable
- Maintainable
Example
## Patterns

<table>
<thead>
<tr>
<th>Category</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>Separated Interface</td>
</tr>
<tr>
<td>Base</td>
<td>Plugin</td>
</tr>
<tr>
<td>Base</td>
<td>Service Stub</td>
</tr>
<tr>
<td>Base</td>
<td>Registry</td>
</tr>
<tr>
<td>Creational</td>
<td>Factory Method</td>
</tr>
<tr>
<td>Creational</td>
<td>Builder</td>
</tr>
<tr>
<td>Creational</td>
<td>Scripted configuration</td>
</tr>
<tr>
<td>Creational</td>
<td>Declarative wiring</td>
</tr>
<tr>
<td>Structural</td>
<td>Adapter</td>
</tr>
<tr>
<td>Structural</td>
<td>Decorator / Proxy</td>
</tr>
</tbody>
</table>
Base Patterns

- **Separated Interface** - put interface in separate package than implementation
- **Plugin** - link classes during configuration rather than compilation
- **Service Stub** - removes dependence on a service during testing
- **Registry** - basically a service locator
Creational Patterns

- **Factory Method** - method to create concrete instances of an abstract type
- **Builder** - object used to configure and create concrete instances
- **Scripted configuration** - move component assembly and wiring to an interpreted script
- **Declarative wiring** - assemble components using a lightweight container and config file
Structural Patterns

- **Adapter** - adapts one interface to another
- **Decorator** - add behavior to an interface by wrapping it
- **Proxy** - change behavior by inserting a proxy
Smells

- Components without interfaces
- Untestable component
- Singletons
- Property hell
Architecture

- When to use service locators
- Open vs closed designs
- External APIs
- Pervasive dependencies (logging)
Links

Books
- Fowler, Martin, Patterns of Enterprise Application Architecture, http://martinfowler.com/books.html#eaa

Blogs
- Miller, Alex, http://www.jroller.com/page/metalex
- Oberg, Rickard, Dependency injection and open vs. closed designs, http://jroller.com/page/rickard/20040814#dependency_injection_and_open_vs
- Lee, Bob, Getter-Based Dependency Injection, http://weblogs.java.net/blog/crazybob/archive/2004/05/getterbased_dep.html
- Weirich, Jim, Dependency Injection in Ruby, http://onestepback.org/index.cgi/Tech/Ruby/DependencyInjectionInRuby.rdoc
- Thomas, Dave, Transparent Inversion of Control, http://blogs.pragprog.com/cgi-bin/pragdave.cgi
- Mathew, Sony, Examining the Validity of Inversion of Control, http://stage.theserverside.com/articles/article.tss?l=IOCandEJB

Software Projects
- Spring, http://www.springframework.org/
- Pico, http://www.picocontainer.org/