Inversion of Control (IoC)

Daniel Rinehart

BFPUG Jun 6, 2007

Definition

- Externalizes the creation and management of component dependencies
- IoC has two subtypes and four flavors
  - Dependency Lookup
    - Dependency Pull
    - Contextualized Dependency Lookup
  - Dependency Injection
    - Constructor Dependency Injection
    - Setter Dependency Injection
- IoC usually talked about in the context of a container

Motivation

- Reduce glue code
- Externalize dependencies
- Manage dependencies in a single place
- Improve testability
- Foster good application design

Traditional

- Class Foo needs a reference to an instance of a class that implements IBar
- public function doSomething()
  {
  var bar:IBar = new Bar();
  }
- Very tight coupling between classes

Dependency Pull

- Centralized registry
- public function doSomething()
  {
  var registry:IRegistry = getRegistry();
  var bar:IBar =
  IBar(registry.getInstance(“bar”));
  }
- Requires common way to get access to registry (Singleton, Extension)

Contextualized Dependency Lookup

- Lookup done against a container that manages registry
- public function lookup(registry:IRegistry)
  {
  this.bar = IBar(registry.getInstance(“bar”));
  }
  public function doSomething()
  {
  // instance level access to bar
  }
- Requires container to manage component

Constructor Dependency Injection

- Dependencies passed into component's constructor
- public function Foo(bar:IBar)
  {
  this.bar = bar;
  }
  public function doSomething()
  {
  // instance level access to bar
  }
- Requires container to create instance

Setter Dependency Injection

- Dependencies set on component through setter methods
- public function set bar(bar:IBar)
  {
  this.bar = bar;
  }
  public function doSomething()
  {
  // instance level access to bar
  }
- Need to know what to set

Which is right for you?

- Lookup requires repeated code, interface implementation, and/or subclassing
- Lookup in general is more complex
- Injection lets you easily manually create instances
- Injection is passive
- Setter Injection better when a component can provide defaults
- Setter Injection allows changes to dependencies of an existing component

IoC and Flex

- MXML acts like a configuration language to define and set your dependencies
- Bindings provide the glue and setters provide the injection point
- Now for a lame example

That’s All Folks

- Much of the material comes from “Pro Spring” by Rob Harrop and Jan Machacek
- http://martinfowler.com/bliki/InversionOfControl.html