Introduction to the Swiz Framework

Brian Kotek
[Team Swiz]

What is Swiz?

What is Swiz?

Brutally simple micro-architecture for Flex and ActionScript applications

Swiz in a nutshell

- Simple IoC for Flex
- Facilitates MVC Architecture
- Simple tools for common tasks
  - Remote method invocation
  - Event handling
- Utilities for advanced development

What Swiz is not

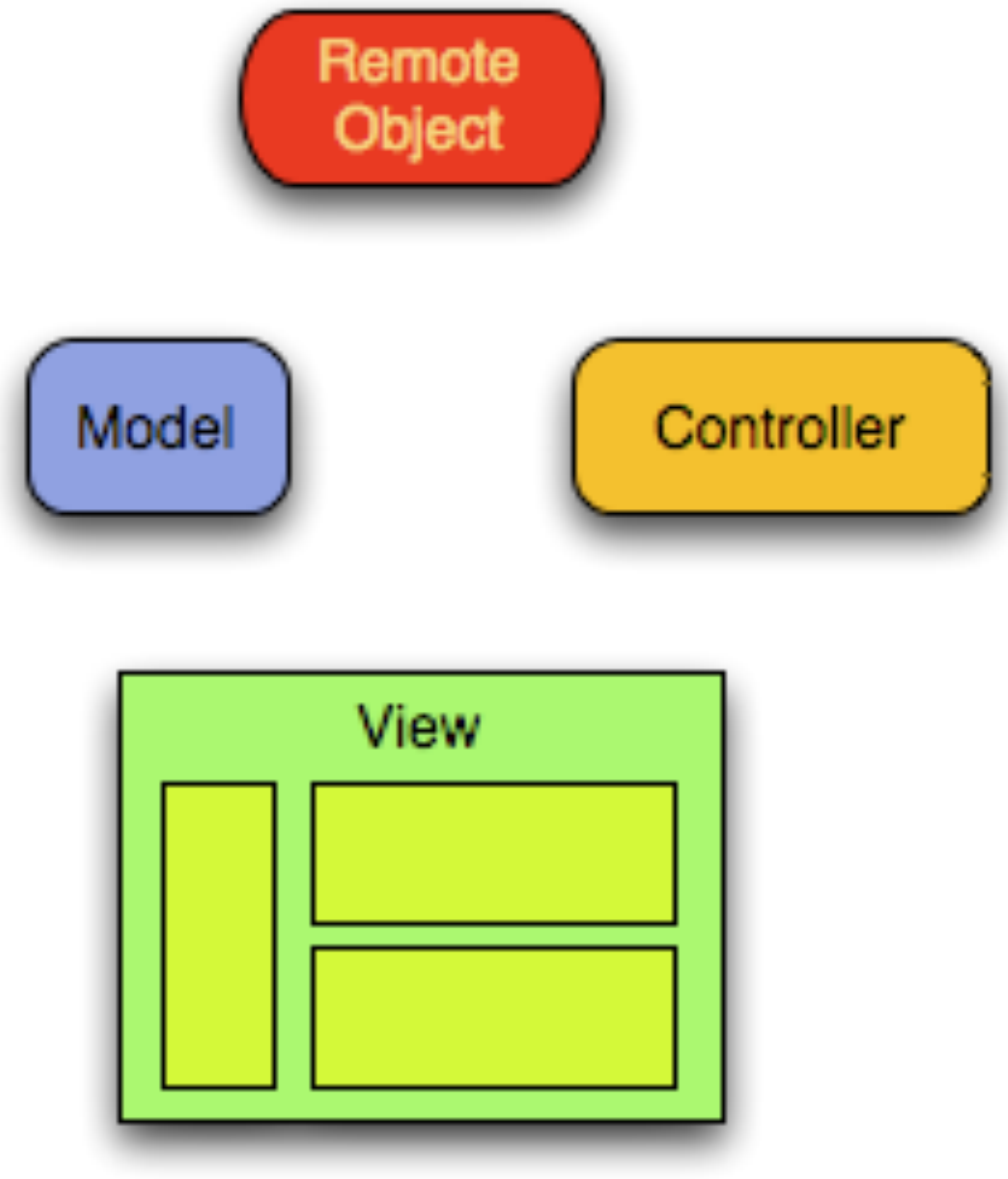
- Excessive JEE patterns
- Boilerplate code
- Verbose XML configuration
- Overly prescriptive

Inversion of Control

Copyright © 2007 Earth Patrol Media

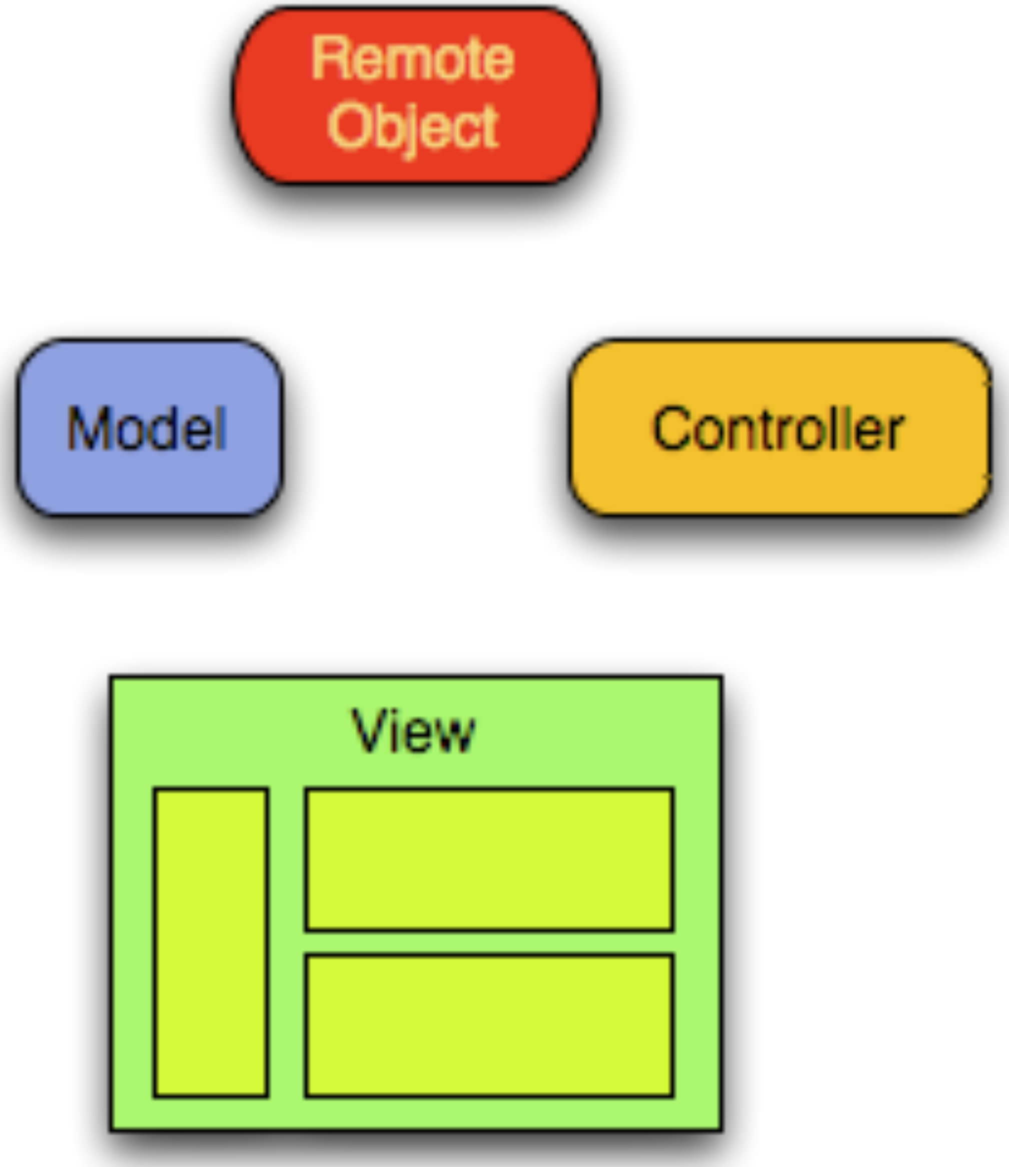
Swiz 101

- Flex Applications require:
  - Remote Services
  - Data
  - Logic
  - Views



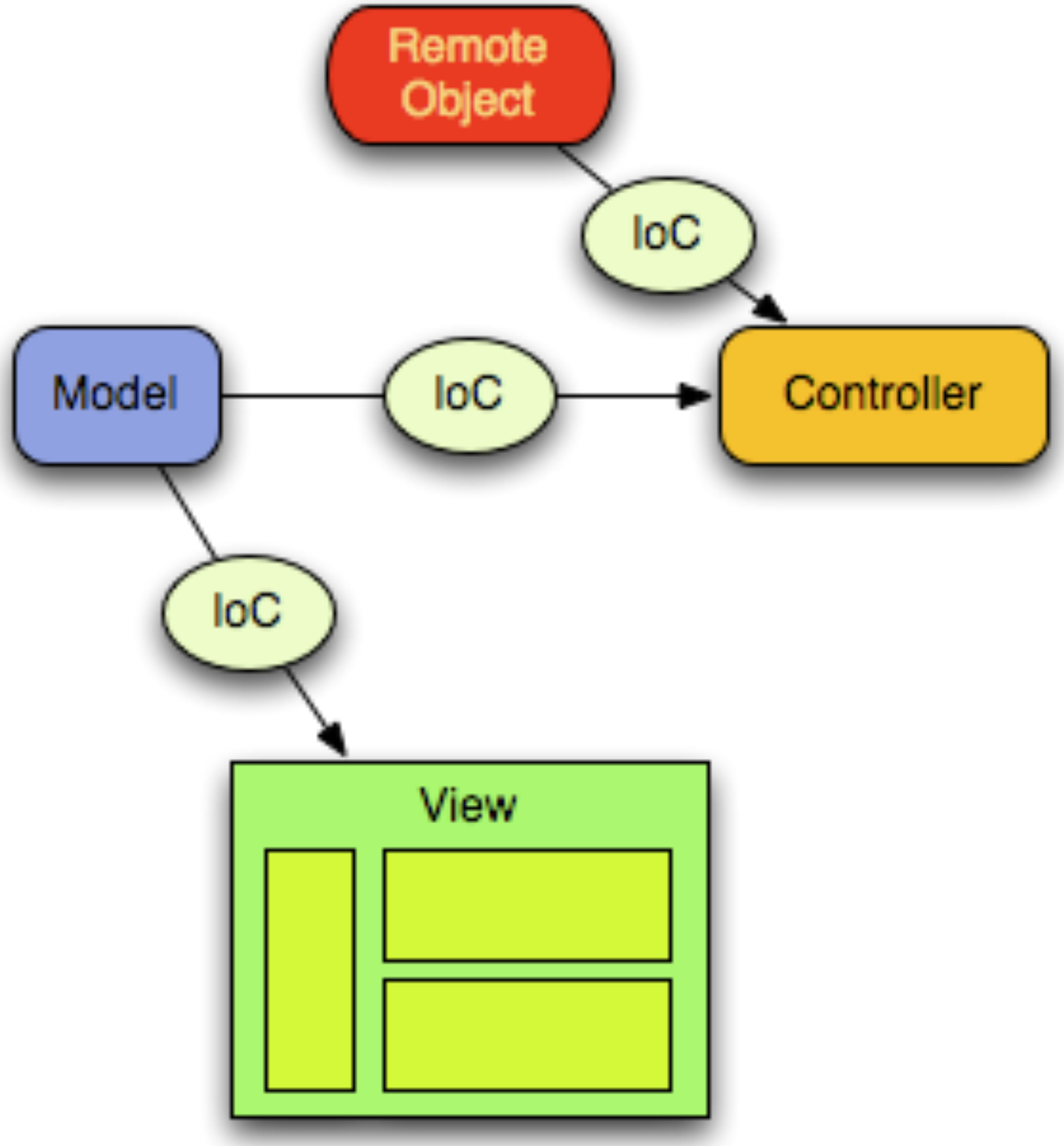
Swiz 101

- Application Components need each other
  - Wire ourselves
  - Use 'Service Locators'
  - Use verbose XML



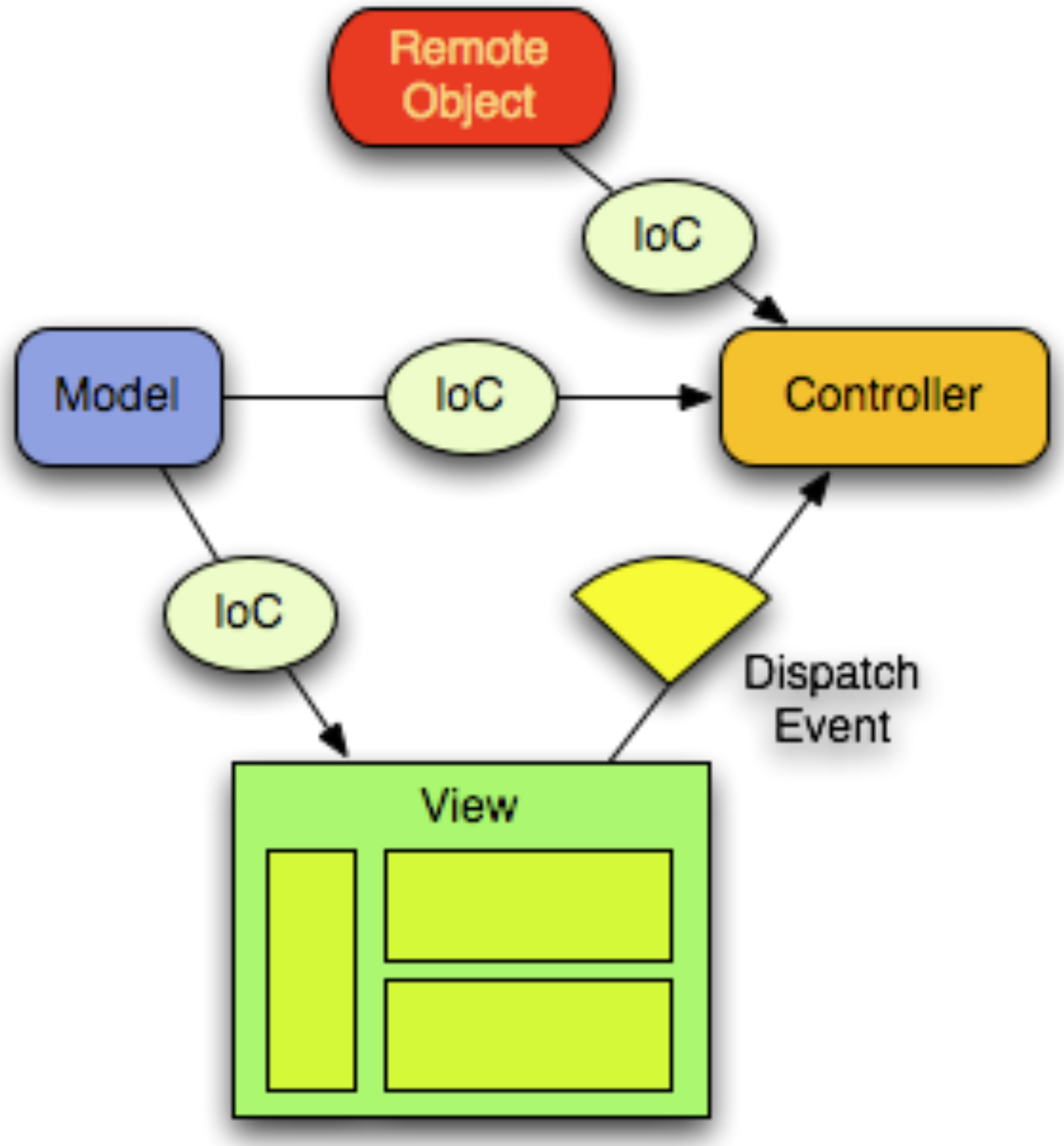
Swiz 101

- Application Components need each other
  - Inversion of Control
  - Annotations



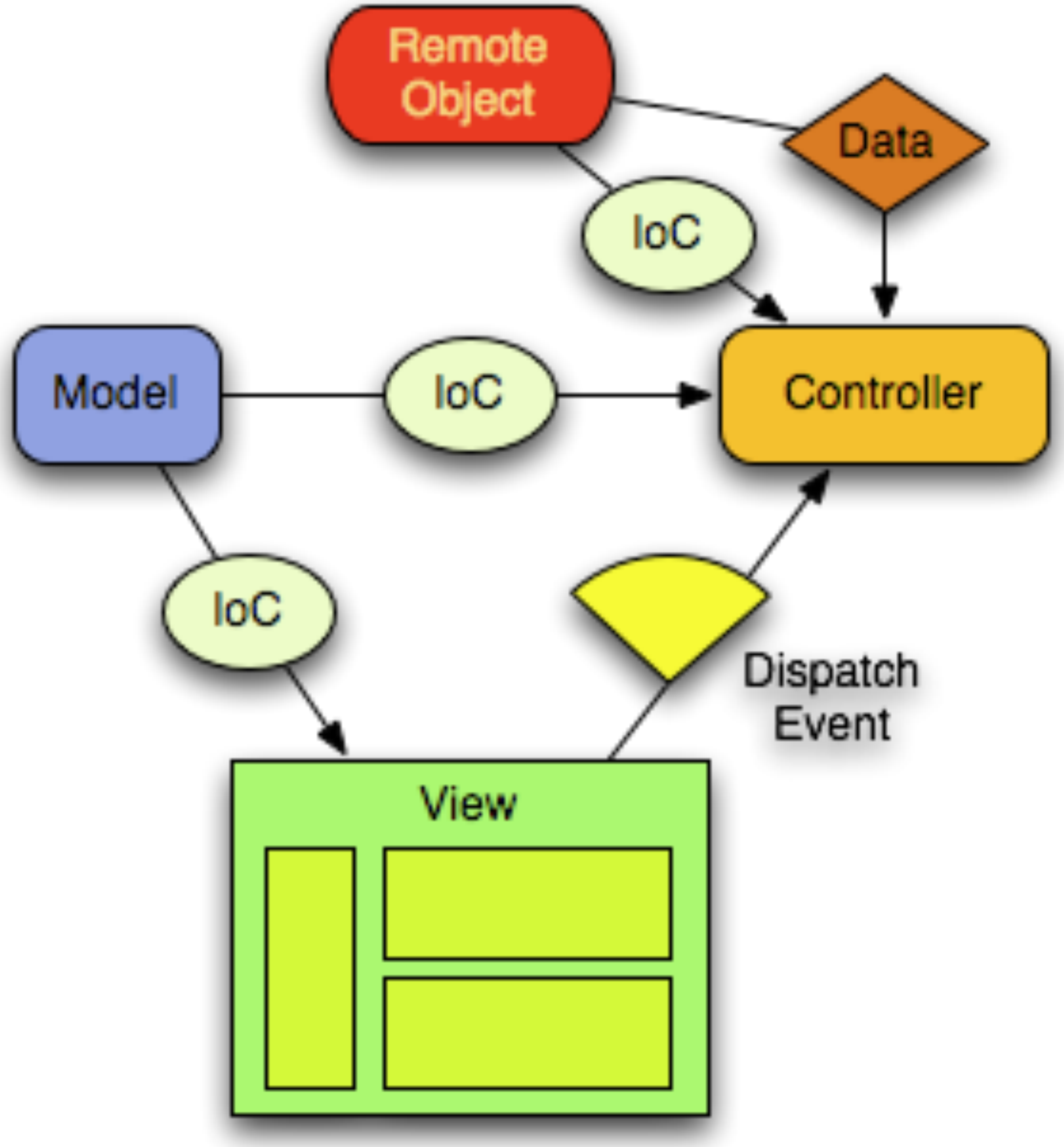
Swiz 101

- Views communicate with components
  - Standard Flex Events
  - Facilitates MVC Paradigm
  - Uses Swiz's Dynamic Mediators



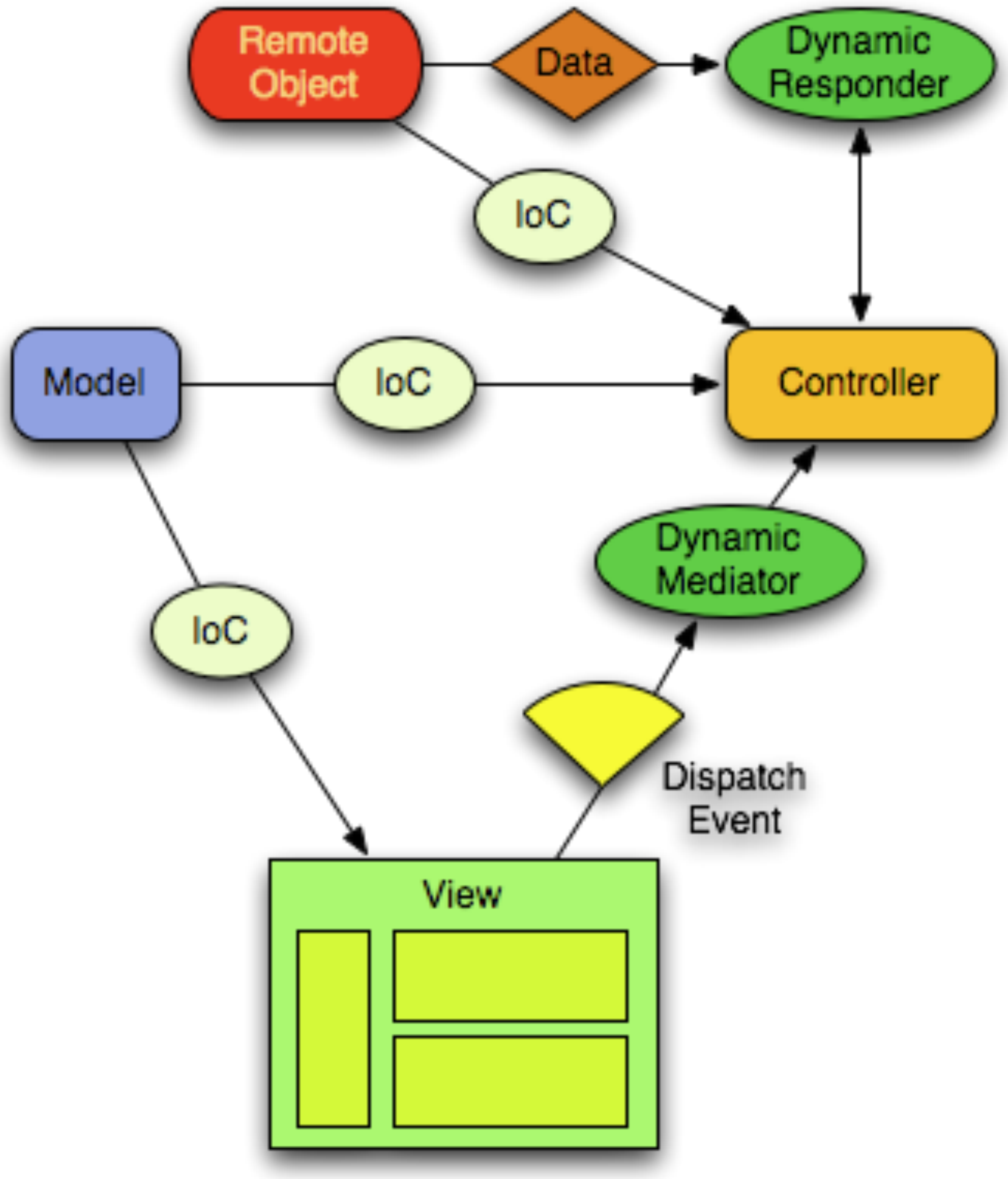
Swiz 101

- Applications need Data
  - Async Tokens
  - Responders
  - State
  - Uses Swiz's Dynamic Responder



Swiz 101

- Swiz Features at a glance
  - IoC
    - for Dependency Injection
  - DynamicResponder
    - for remote data
  - DynamicMediator
    - for event handling
  - And so much more...



Defining Beans

- Define Beans in "BeanProviders"
- Beans are defined in plain old MXML
- Swiz calls objects "Beans" because it only cares about their properties.

Swiz's IoC Factory

- When Swiz loads beans, it searches for [Inject] metadata
- When objects are created, Swiz does its dependency injection magic
- Swiz adds event listeners to listen for views being added to the application
- Cleans up when views are removed

Expressing Dependencies

- Dependencies are not defined in MXML
- Use [Inject] in AS blocks / objects
- Similar to Spring configuration

```
[Inject]

public var userController:UserController
```

Expressing Dependencies

- Typically you can simply inject by type
- Can specify bean ID if necessary
- Works with interfaces
- Works with inheritance

Remote Services

- Swiz provides help for working with Remote Services
- Dynamic Responders
- Dynamic Commands

Dynamic Responders

- Bind result and fault handlers transparently
- Done using executeServiceCall()
- Can pass through additional data to maintain state over asynchronous calls

Dynamic Commands

- Created using createCommand()
- Typically used with CommandChain
- Handles multiple events as a single unit
- Can abort or proceed if a command fails
- Can run in series or in parallel
- Works with asynchronous server calls, or internal Flex event chains

Event Handling

- Swiz provides easy access to an event dispatcher in beans:

```
[Dispatcher]
public var dispatcher:IEventDispatcher
```

- Allows different parts of an application to work together, whether they are DisplayObjects or not!

Event Mediation

- Helps greatly to decouple views and controllers
- Enables very simple event handling
- Done using [Mediate] annotation

```
[Mediate(event=”type”)]
public function doSomething()
```

Event Mediation

- Mediates standard Flex events
- No special Event classes or Swiz-specific Events are needed
- Handles events dispatched from display list too

Changes in Swiz 1.0

- Moved to GitHub
- No static methods any more
- No central dispatcher (use [Dispatcher])
- [Autowire] deprecated (use [Inject])
- Small changes to configuration/setup

Changes in Swiz 1.0

- Module support
- AIR windows support
- Additional metadata: [PostConstruct], [PreDestroy]
- Custom metadata processors (might just be THE killer feature of Swiz)

Swiz is Almost Invisible

- Extremely unobtrusive
- No prescriptive code or approaches forced on you
- Nearly everything is done with metadata
- Virtually no explicit coupling to the framework

A Swiz Application

Let’s see some code!

Roadmap

- 1.0 RC is imminent
- Documentation is a priority and is being built up now
- Bug tracker will be made public
- Sample apps at GitHub (w/ more coming)
- Already discussing great post-1.0 stuff

Wrap It Up!

Questions? Comments?

Thanks!

www.swizframework.org

github.com/swiz/swiz-framework/

www.briankotek.com/blog