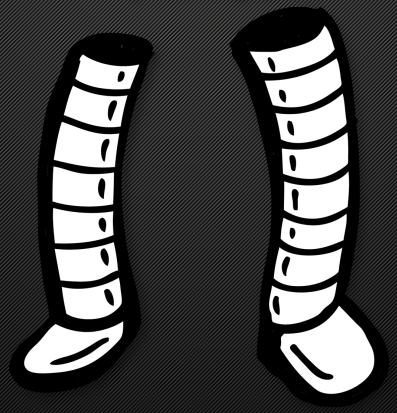
[robotlegs]



for Actionscript 3

About Me

Paul Robertson

Robotlegs Enthusiast

on the web

http://probertson.com

@probertson

email

paul@probertson.com

Who Started Robotlegs?

Shaun Smith
Robotlegs Inventor

find him
http://shaun.boyblack.co.za
@darscan

Not to mention...

Till Schneidereit
SwiftSuspenders Maestro

find him
http://tillschneidereit.de/

@tschneidereit

And...

Joel Hooks

Robotlegs Enthusiast

on the web

http://www.joelhooks.com

@jhooks

email

joelhooks@gmail.com

Contributors

Robert Penner @robPenner

Jonnie Hallman @DestroyToday

Sean Hess @seanhess

Craig Wickesser @codecraig

and so many others have contributed to the community
through support, ideas, and examples...

What is Robotlegs?

a pure Actionscript 3 micro-architecture

What is Robotlegs?

a pure Actionscript 3 micro-architecture

robotlegs has no Flex framework dependencies.

Why Robotlegs?

flash, flex, and air

What is Robotlegs?

a mechanism for wiring objects together

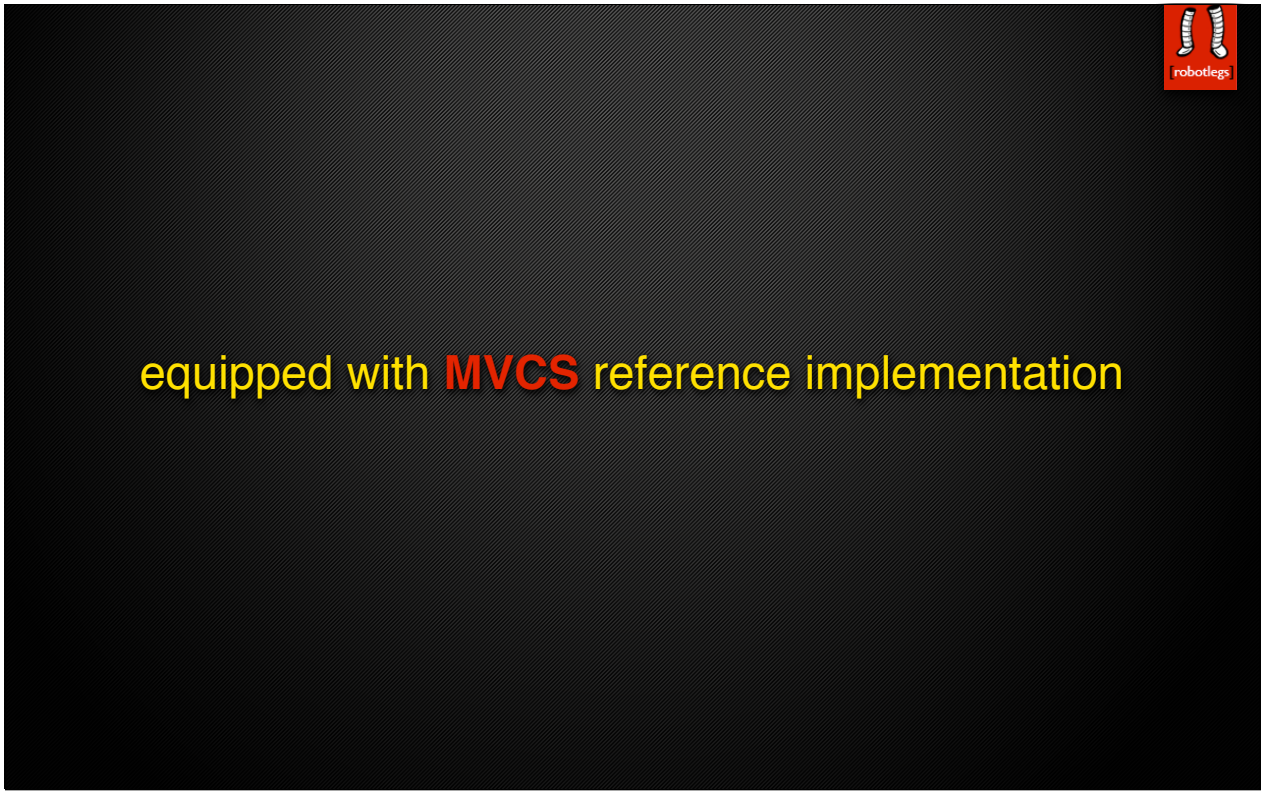
What is Robotlegs?

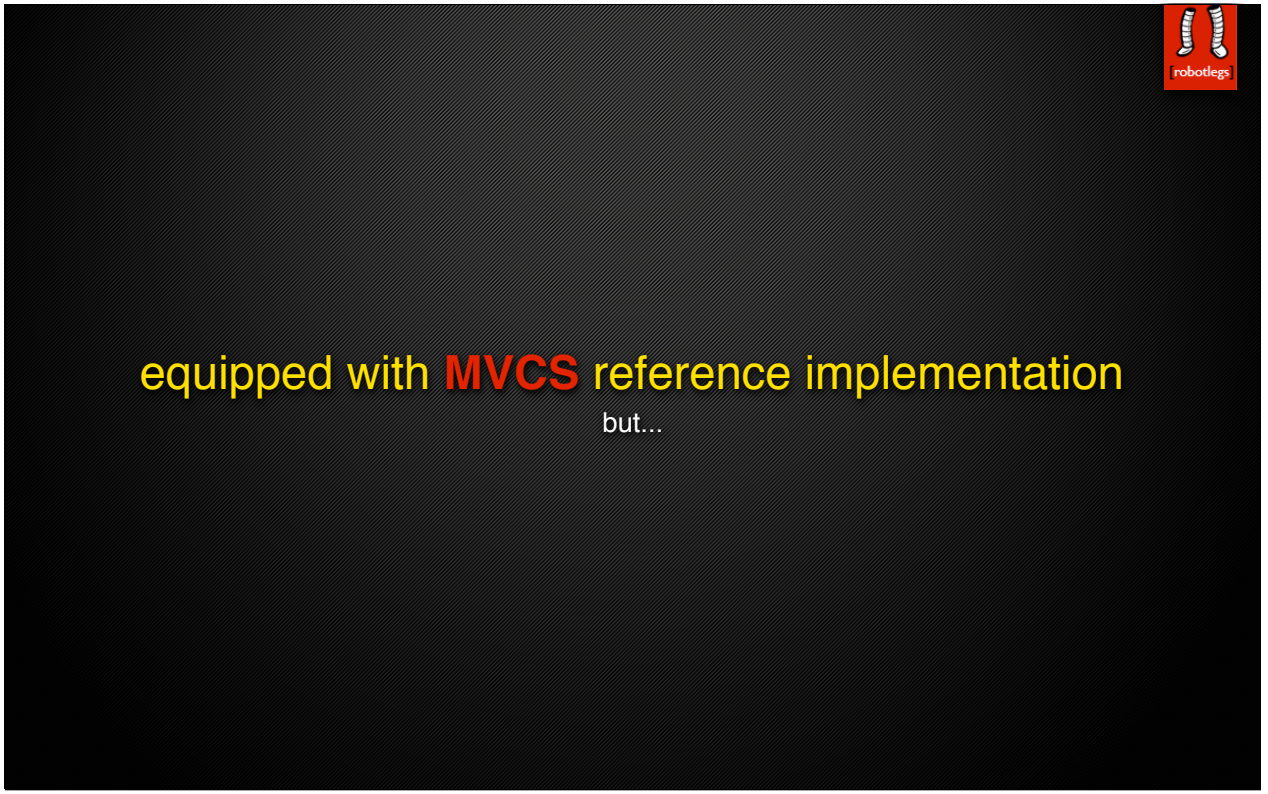
a mechanism for wiring objects together

focused solely on this limited scope

Why Robotlegs?

lightweight


[robotlegs]
equipped with MVCS reference implementation


robotlegs
equipped with MVCS reference implementation
but...

Why Robotlegs?

whatever you need it to be

Why Robotlegs?

whatever you need it to be

highly extensible to support YOUR workflow and style

Why Robotlegs?

full modular support

Why Robotlegs?

supports your workflow

Why Robotlegs?

peer reviewed

Why Robotlegs?

well documented

Why Robotlegs?

transparent development

Why Robotlegs?

stable

Why Robotlegs?

welcoming community

Why Robotlegs?

knowledge.robotlegs.org

supported through Tender

What does Robotlegs do for you?

removes framework pain

What does Robotlegs do for you?

allows you to focus on your app

What does Robotlegs do for you?

allows you to focus on your app

not the framework

What does Robotlegs do for you?

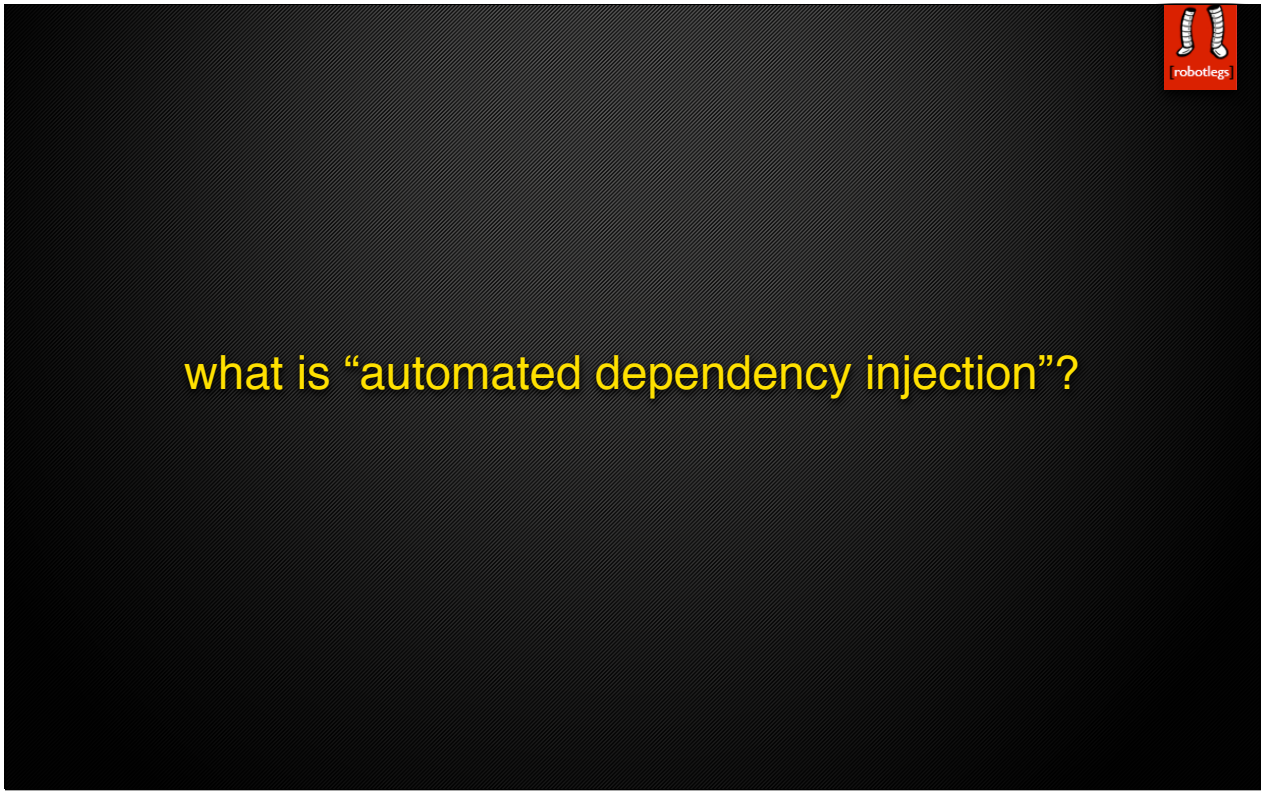
fights carpel tunnel

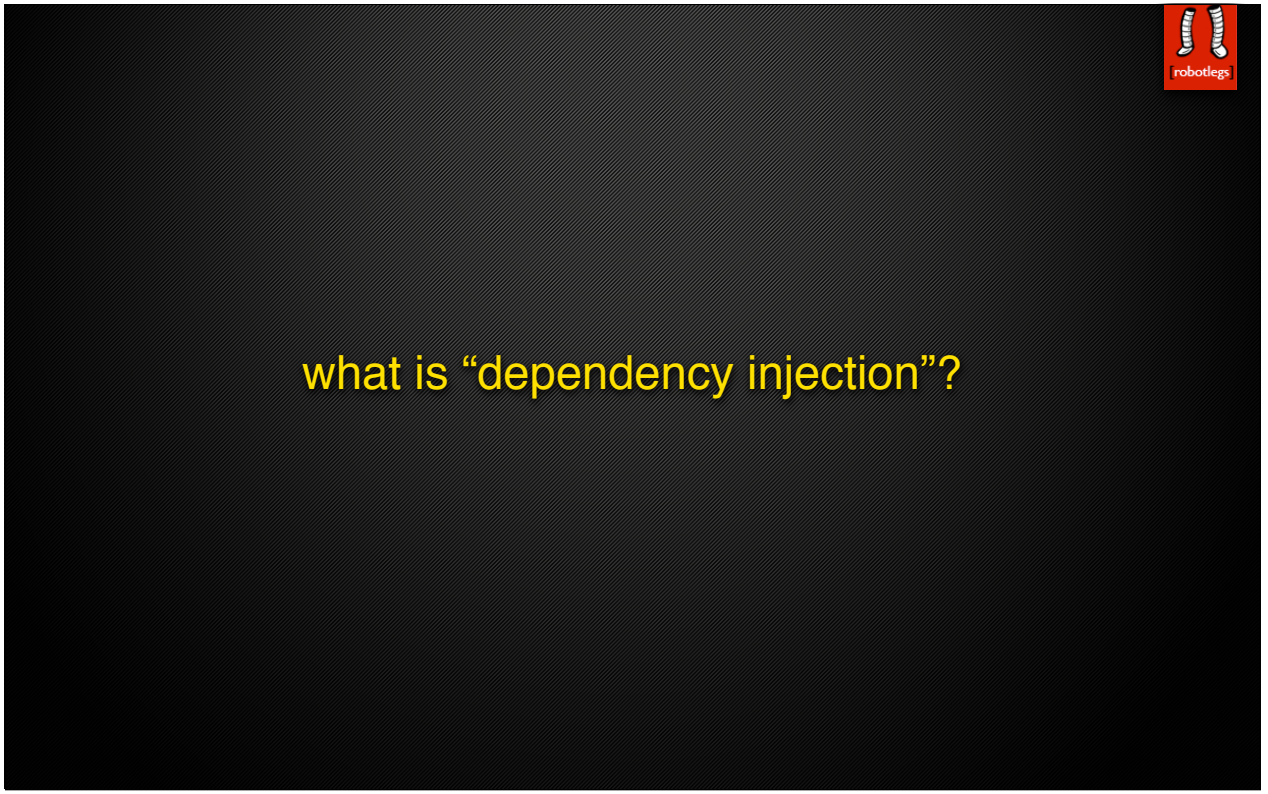
What does Robotlegs do for you?

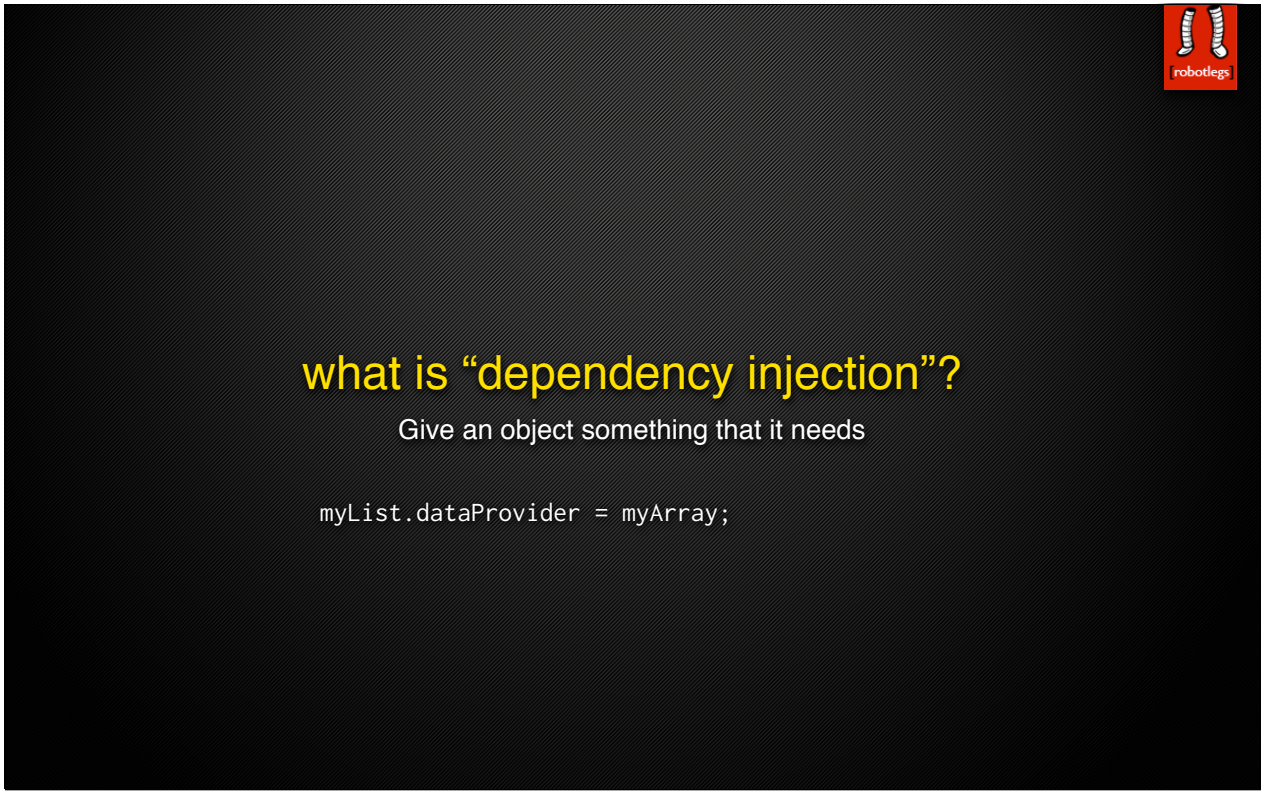
lets you move fast

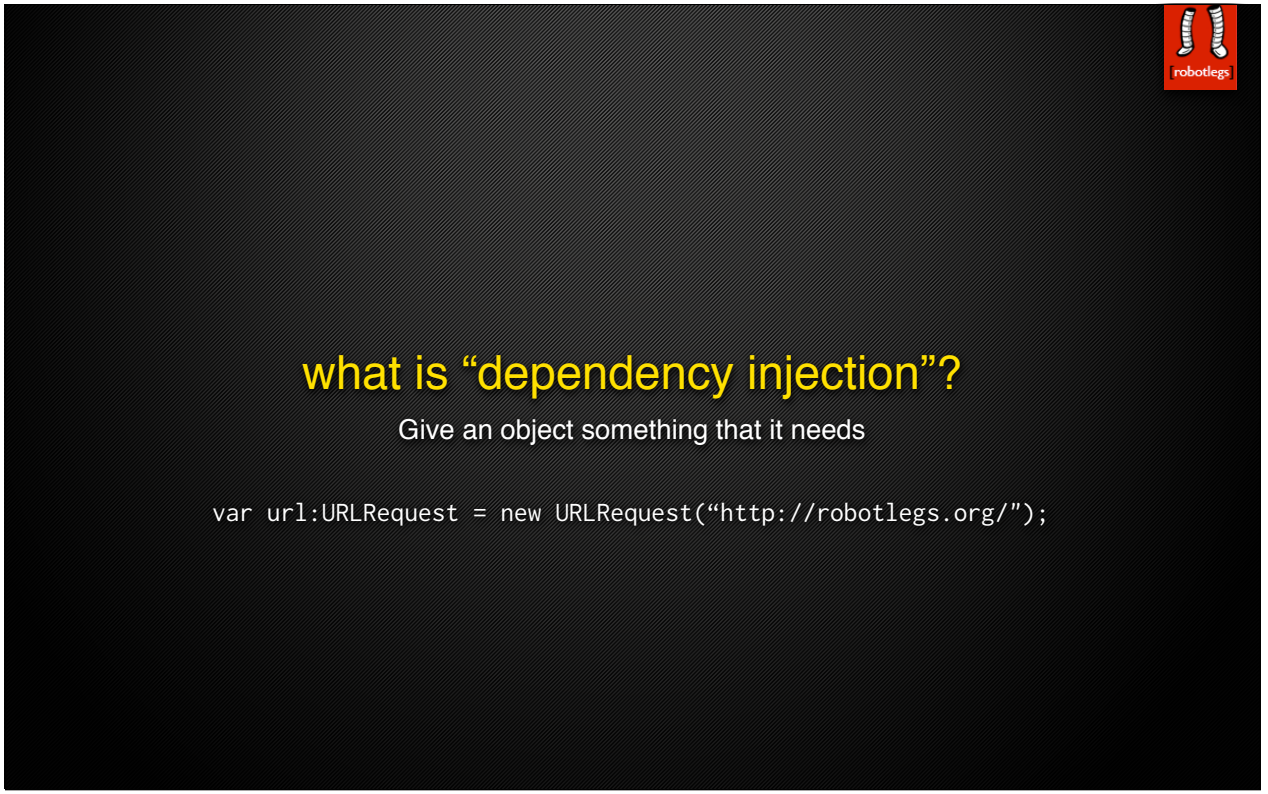
What does Robotlegs do for you?

automated dependency injection


[robotlegs]
what is “automated dependency injection”?


robotlegs
what is “dependency injection”?


robotlegs
what is “dependency injection”?
Give an object something that it needs
myList.dataProvider = myArray;


[robotlegs]
what is “dependency injection”?
Give an object something that it needs
var url:URLRequest = new URLRequest(“http://robotlegs.org/”);

what is “dependency injection”?

Give an object something that it needs

```
var url:URLRequest = new URLRequest(“http://robotlegs.org/”);

var loader:URLLoader = new URLLoader();
loader.load(url);
```

What does Robotlegs do for you?

automated dependency injection

[Inject]

What does Robotlegs do for you?

automated dependency injection

Have you ever written code like this*?

```
// in Main.as:
private var _widgetData:WidgetData = new WidgetData();
component1.widgetData = _widgetData;

// in Component1.as
public function set widgetData(value:WidgetData):void
{
    _widgetData = value;
    component2.widgetData = _widgetData;
    component3.widgetData = _widgetData;
}

// in Component2.as, Component3.as
// etc.
```

*adapted from http://probertson.com/projects/run-air-sqlite-query-testing-tool/

What does Robotlegs do for you?

automated dependency injection

Wouldn't you rather write this?

```
// in MainMediator.as:
[Inject]
public var widgetData:WidgetData;

// in Component1.as:
[Inject]
public var widgetData:WidgetData;

// in Component2.as, Component3.as, etc.:
[Inject]
public var widgetData:WidgetData;
```

What does Robotlegs do for you?

automated dependency injection
promotes clean code

What does Robotlegs do for you?

automated dependency injection
promotes clean code

clean code is easier to test

What does Robotlegs do for you?

automated dependency injection
promotes clean code

clean code is easier to refactor

What does Robotlegs do for you?

automated dependency injection
promotes clean code

clean code is easier to understand

What does Robotlegs do for you?

even if you don't test your code,
you should WRITE testable code

What does Robotlegs do for you?

even if you don't test your code,
you should WRITE testable code

Robotlegs wants to help

What does Robotlegs do for you?

objects communicate via native events

What does Robotlegs do for you?

objects communicate via native events

custom events with strongly typed properties

Collaborative Development

github makes it easy!

github.com/robotlegs

Collaborative Development

comments, criticism, and ideas welcome

Collaborative Development

fork the framework

Collaborative Development

create examples and utilities

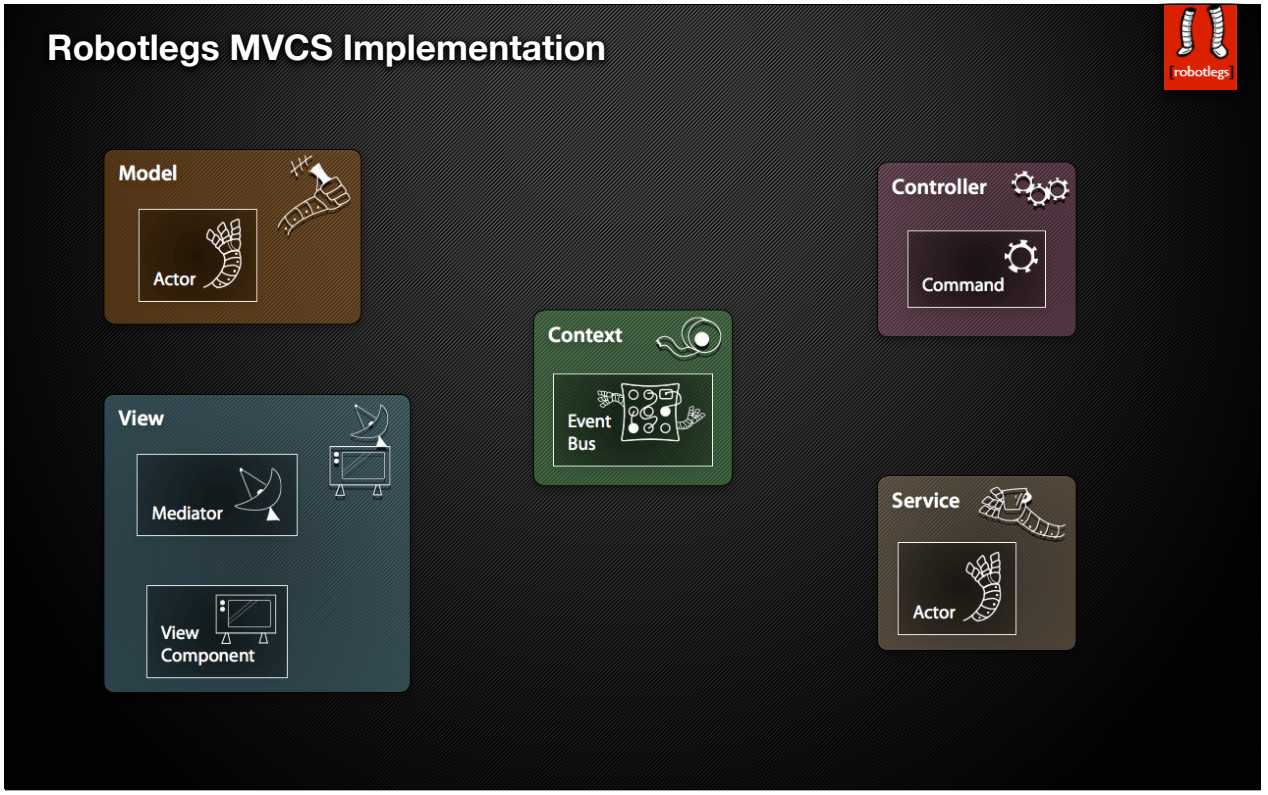
Collaborative Development

create an alternative implementation

Where to Start?

MVCS reference implementation

inspired by PureMVC


Robotlegs MVCS Implementation
[robotlegs]
Model
Actor
Controller
Command
Context
Event
Bus
View
Mediator
View
Component
Service
Actor

MVCS Implementation

MVCS is not the framework

MVCS Implementation

what does that mean?

MVCS Implementation

it is an implementation of Robotlegs

MVCS Implementation

it is an implementation of Robotlegs

a place to start

MVCS Implementation

it is an implementation of Robotlegs

a place to start

get a feel for the possibilities

MVCS Implementation

it is an implementation of Robotlegs

a place to start

get a feel for the possibilities

don’t let it wall you in

Robotlegs in Action

File reader

Context

Context initializes the framework

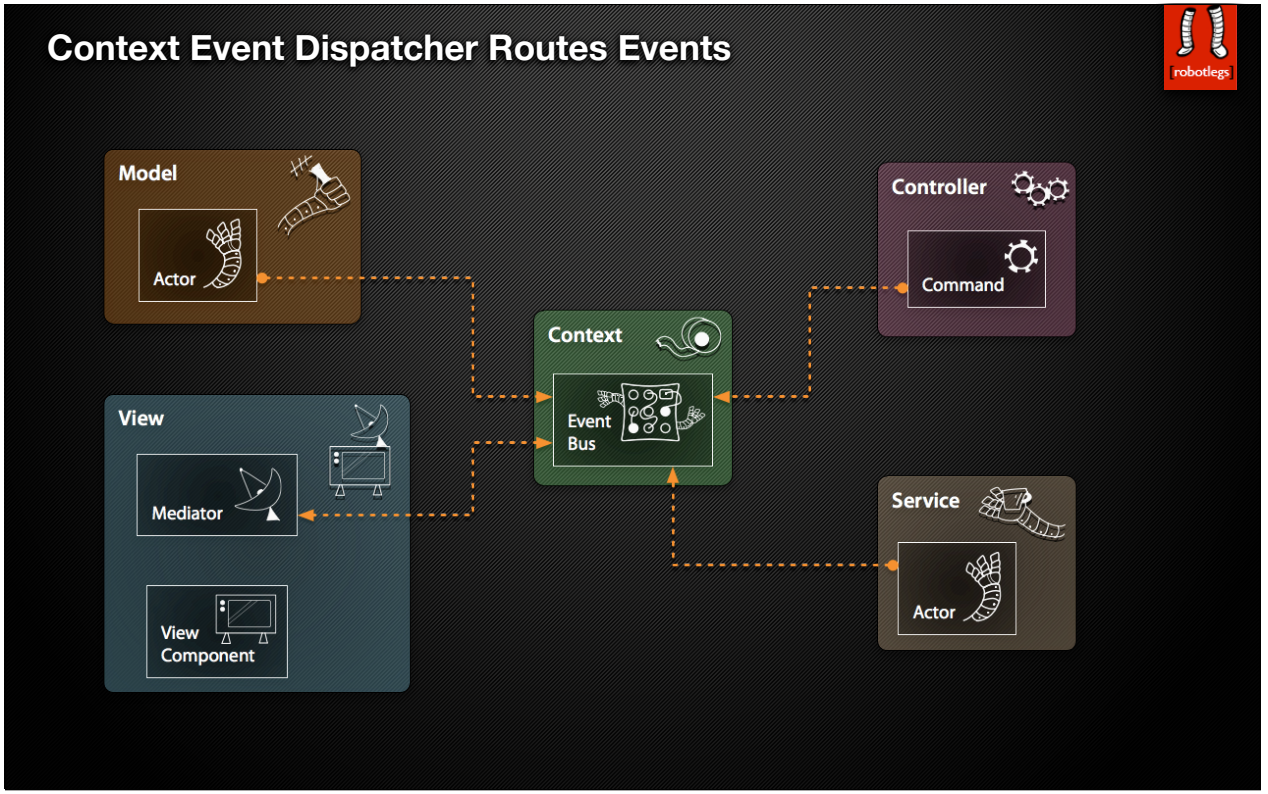
Context

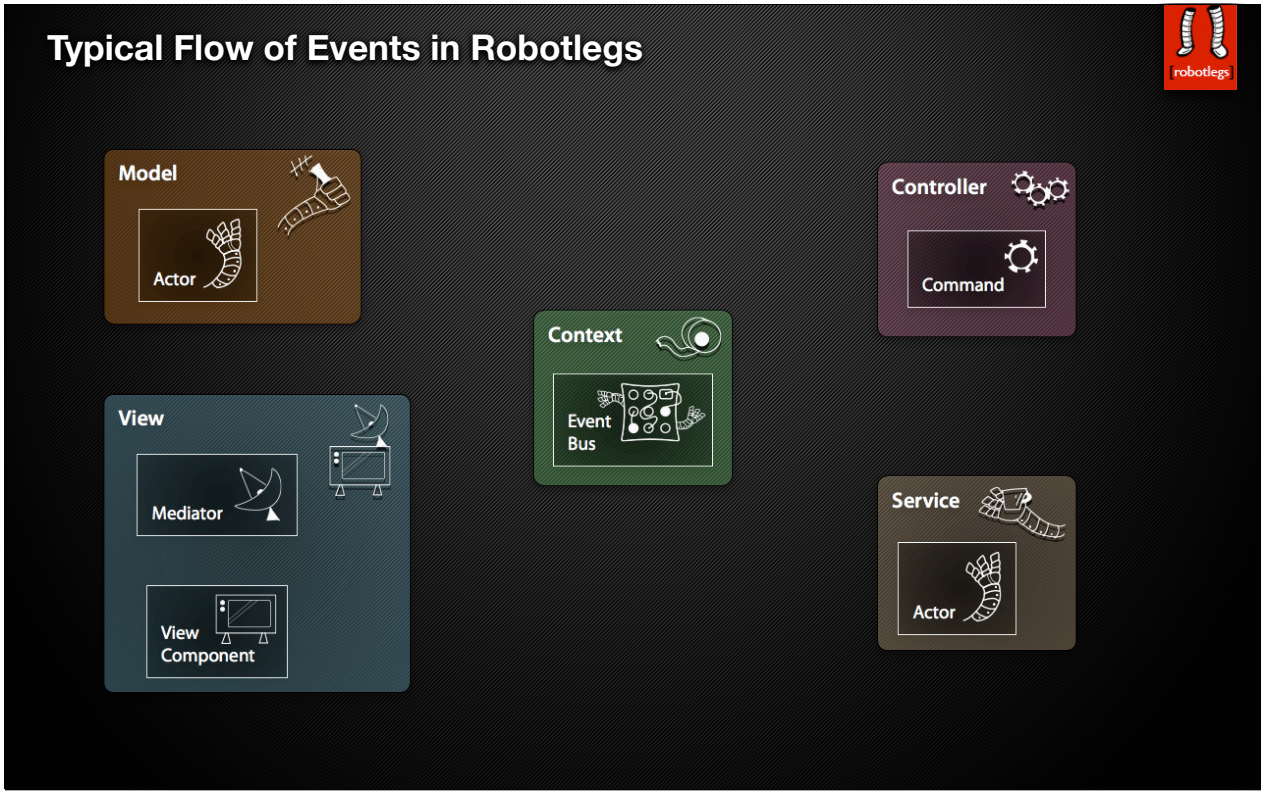

```
package simple
{
    import org.robotlegs.mvcs.Context;

    public class SimpleContext extends Context
    {
        override public function startup():void
        {
        }
    }
}
```

```
<fx:Declarations>
    <simple:SimpleContext contextView="{this}"/>
</fx:Declarations>
```

Context

Context provides an event bus

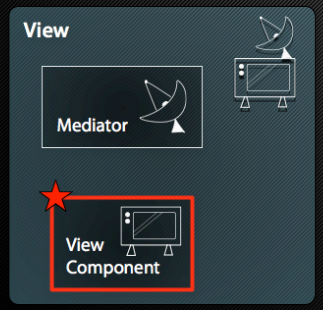

Context Event Dispatcher Routes Events
robotlegs
Model
Actor
Controller
Command
Context
Event
Bus
View
Mediator
View
Component
Service
Actor


Typical Flow of Events in Robotlegs
robotlegs
Model
Actor
Controller
Command
Context
Event
Bus
View
Mediator
View
Component
Service
Actor

User Performs an Action (button click)

1. Create event class
2. Add to button:

```
click="dispatchEvent(new SimpleAppEvent(SimpleAppEvent.CHOOSE_FILE));"
```



View Component Dispatches Event to Mediator

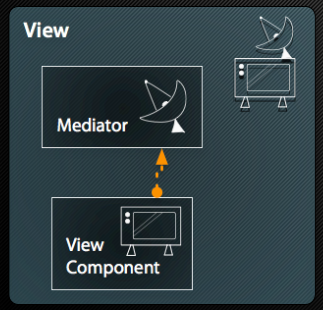
1. Create mediator

2. Create view <-> mediator mapping in context

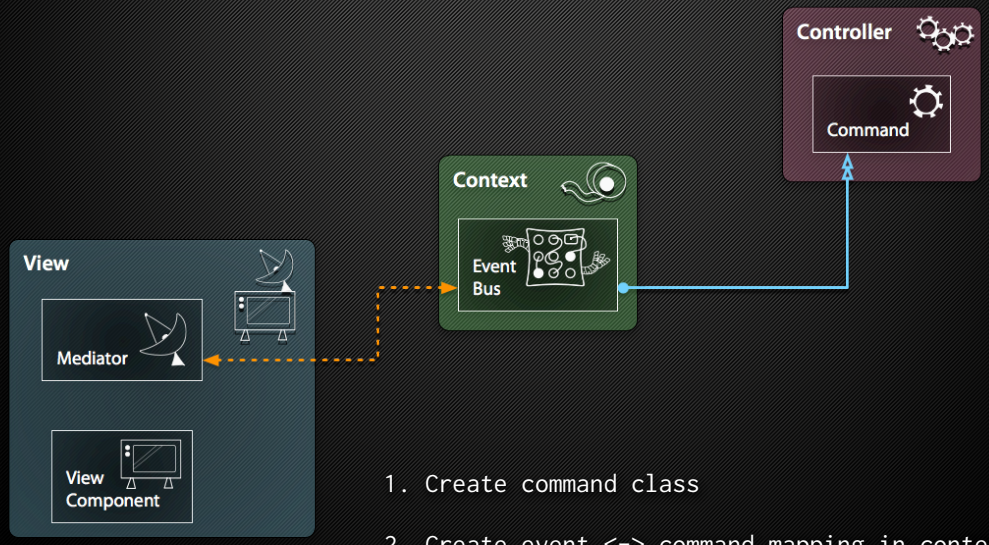
```
mediatorMap.mapView(ButtonContainer, ButtonContainerMediator);
```

3. Register for view event in mediator

```
addViewListener(SimpleAppEvent.CHOOSE_FILE, chooseFileHandler, SimpleAppEvent);
```



Mediator Dispatches Event that Triggers Command



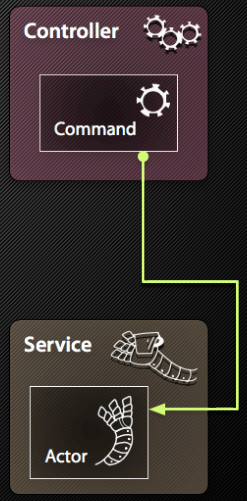
1. Create command class

2. Create event <-> command mapping in context

```
commandMap.mapEvent(SimpleAppEvent.CHOOSE_FILE, ChooseFileCommand, SimpleAppEvent);
```

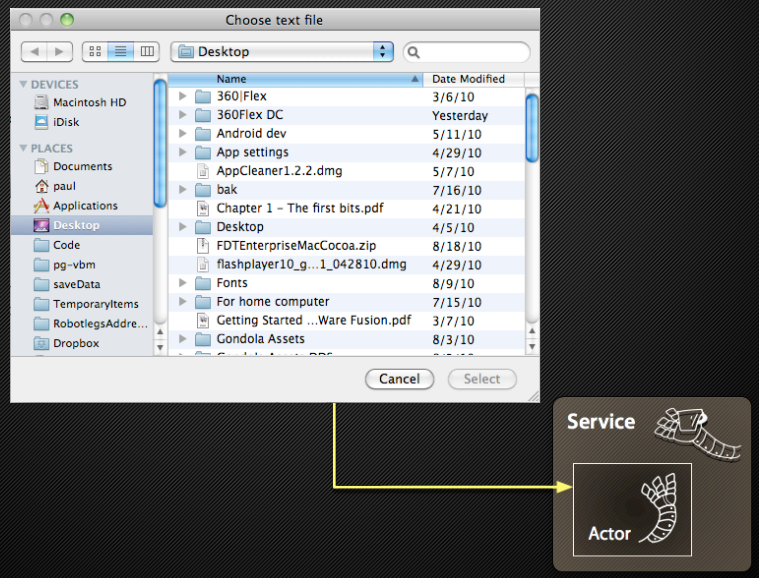
Command calls a Method on a Service

```
1. Create service interface
function promptToChooseFile():void;

2. Inject service into command
[Inject]
public var fileService:IFileService;

3. Call service in execute() method
fileService.promptToChooseFile();
```

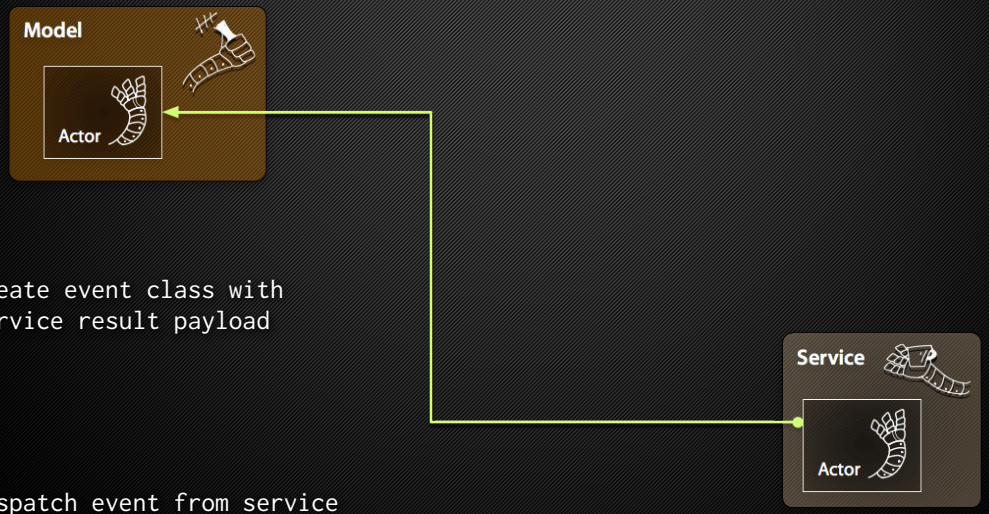



```
Oh yeah:
4. Create service implementation

5. Create interface <-> service mapping in context
injector.mapSingletonOf(IFileService, FileService);
```

Service Makes External call and Parses Data



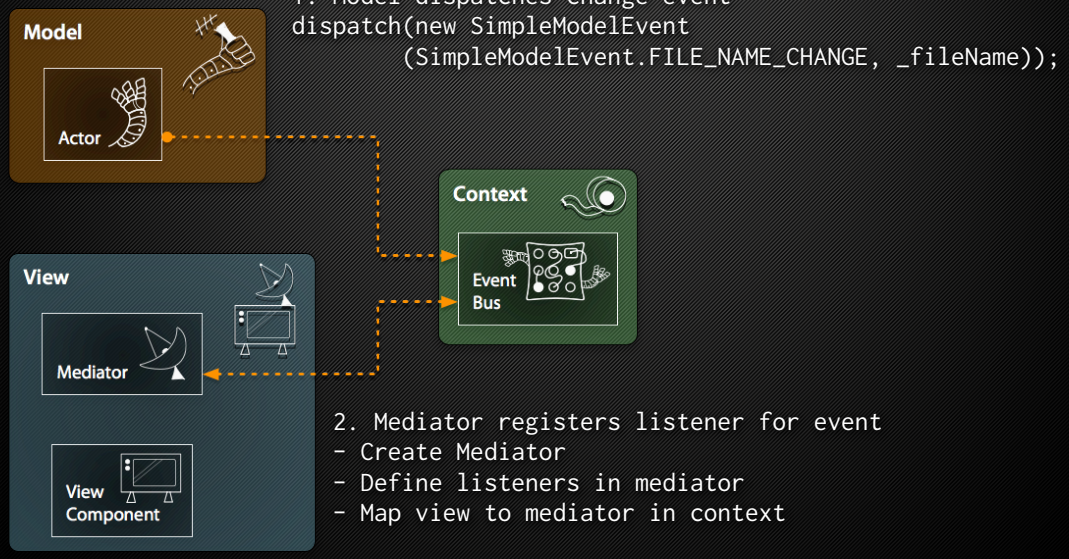
Service Updates the Model (directly or via Command)



1. Create event class with
   service result payload

```
2. Dispatch event from service
dispatch(new FileResultEvent(FileResultEvent.FILE_RESULT, _file.name, _file.nativePath));

3. Repeat steps for creating a command
```

Model dispatches Event that Mediator listens for

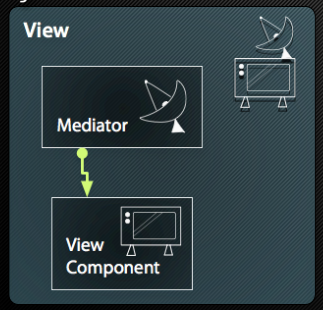



```
1. Model dispatches change event
dispatch(new SimpleModelEvent
        (SimpleModelEvent.FILE_NAME_CHANGE, _fileName));
```

```
2. Mediator registers listener for event
- Create Mediator
- Define listeners in mediator
- Map view to mediator in context
```

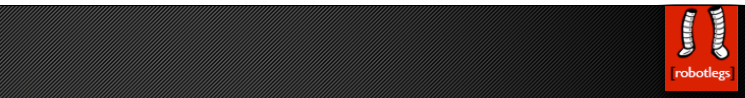
Mediator Updates View with Current Data


```
1. Create public api on view
public function setFileName(name:String):void
{
    fileName.text = name;
}

public function setFilePath(path:String):void
{
    filePath.text = path;
}
```

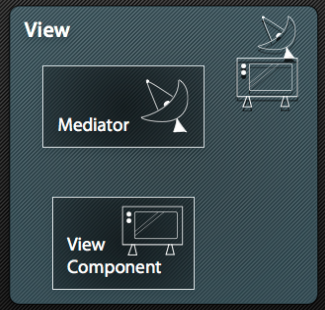



```
2. Inject view into mediator

[Inject]
public var view:TextContainer;


3. Call api from mediator

view.setFileName(event.value);
view.setFilePath(event.value);
```

View



the View is represented by your
view components and their Mediators



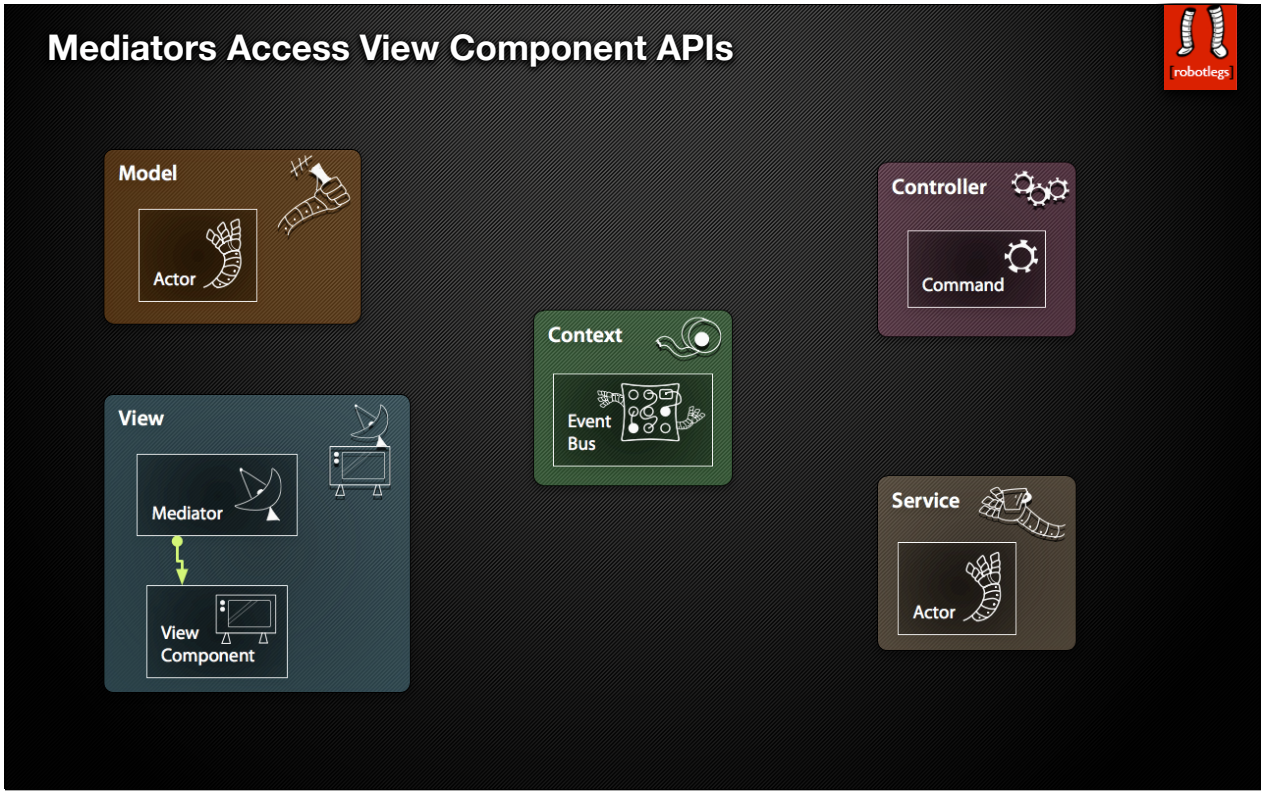
View

Mediators provide API for view components

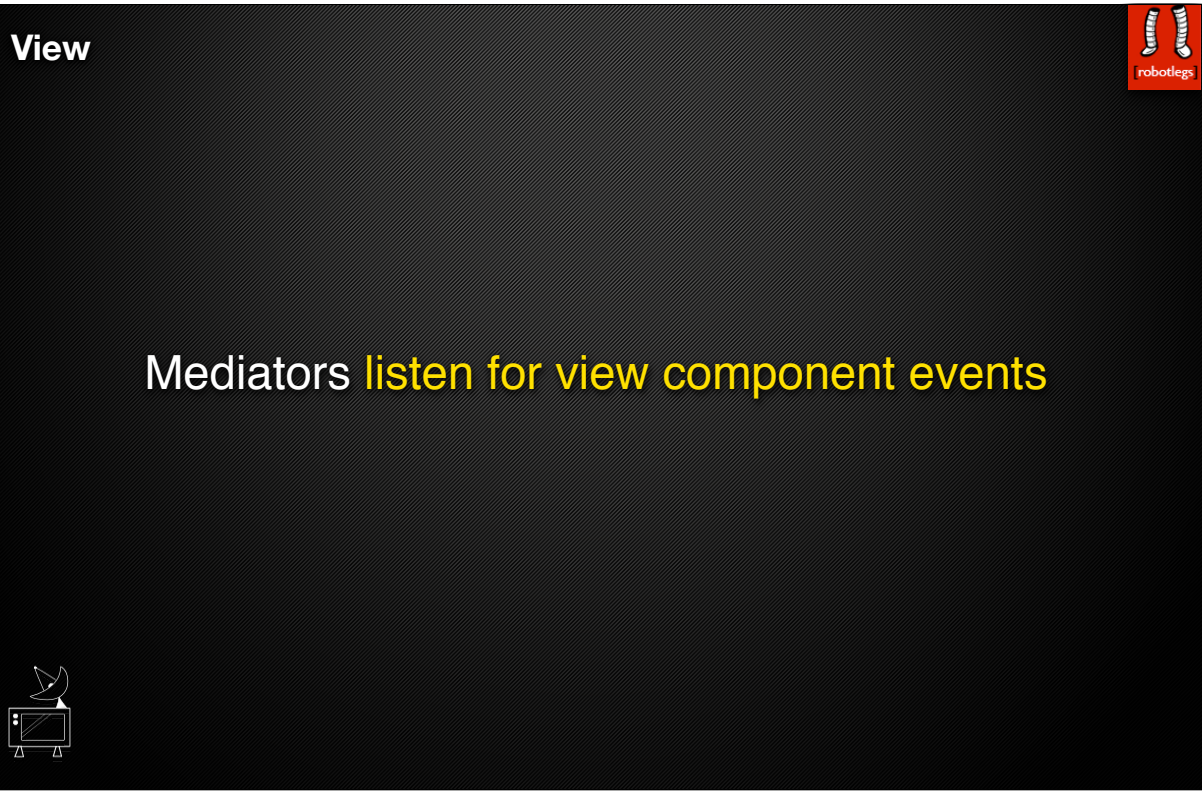
View

Mediators provide API for view components

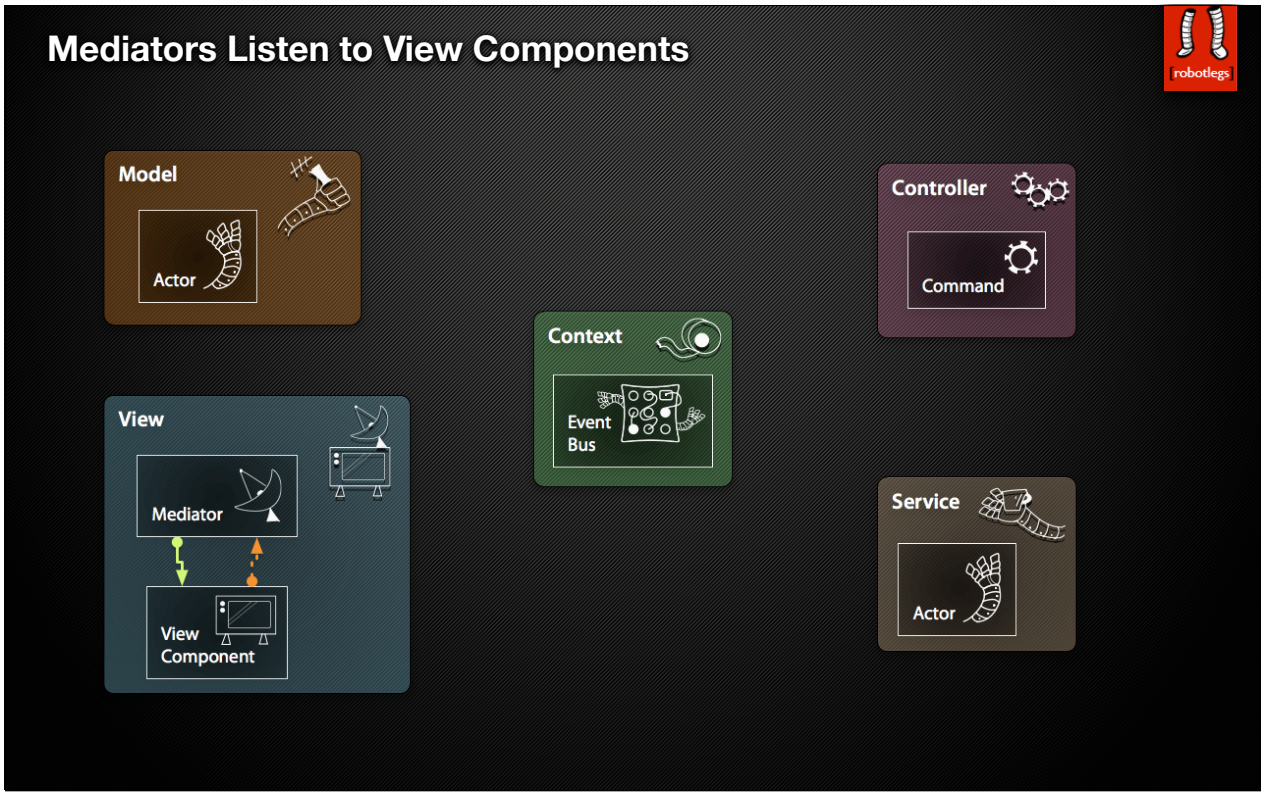
to keep the framework out


Mediators Access View Component APIs
robotlegs
Model
Actor
Controller
Command
Context
Event
Bus
View
Mediator
View
Component
Service
Actor

View

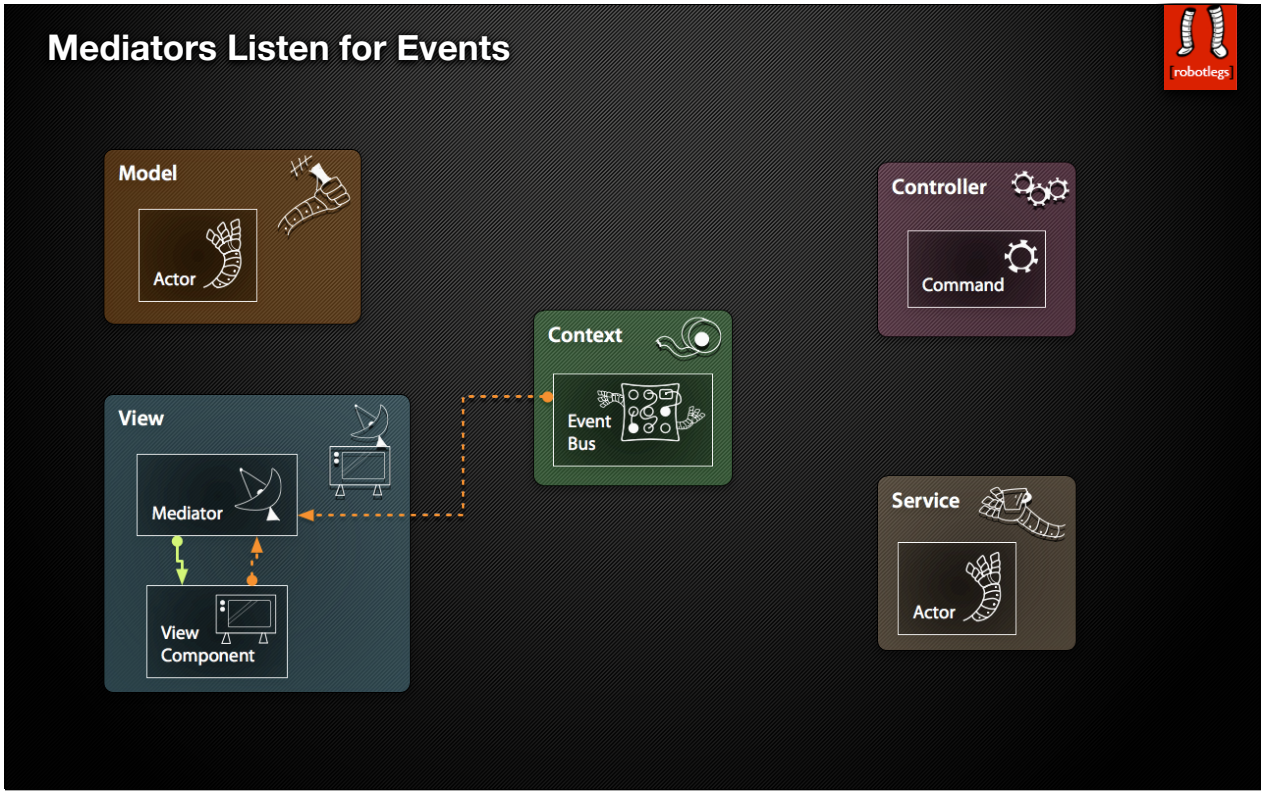


Mediators Listen to View Components



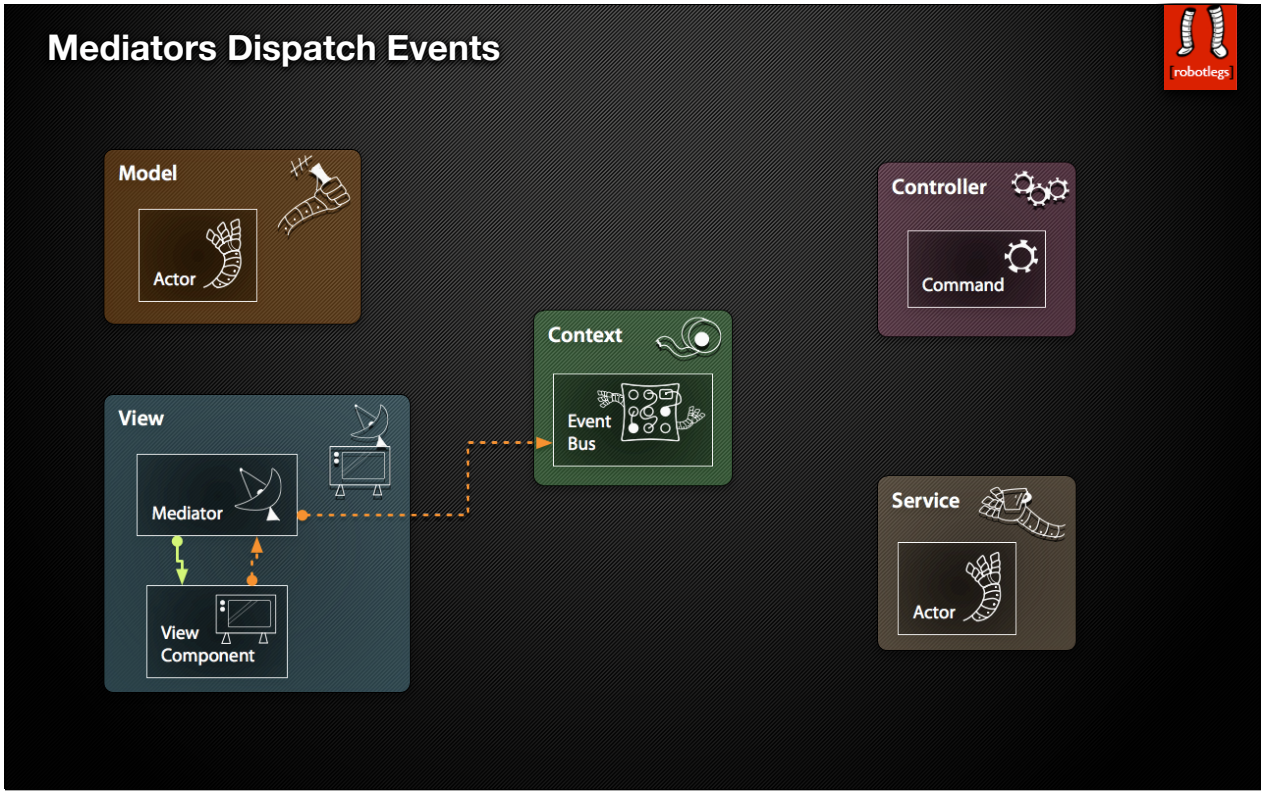
View

Mediators listen for framework events


Mediators Listen for Events
robotlegs
Model
Actor
Controller
Command
Context
Event
Bus
View
Mediator
View
Component
Service
Actor

View

Mediators dispatch framework events


Mediators Dispatch Events
robotlegs
Model
Actor
Controller
Command
Context
Event
Bus
View
Mediator
View
Component
Service
Actor

View

view components are not coupled to their Mediators

View

view components are not coupled to their Mediators

or any other framework class

View

view components are not coupled to their Mediators

or any other framework class
period.

View

Mediators are coupled to their view components

View

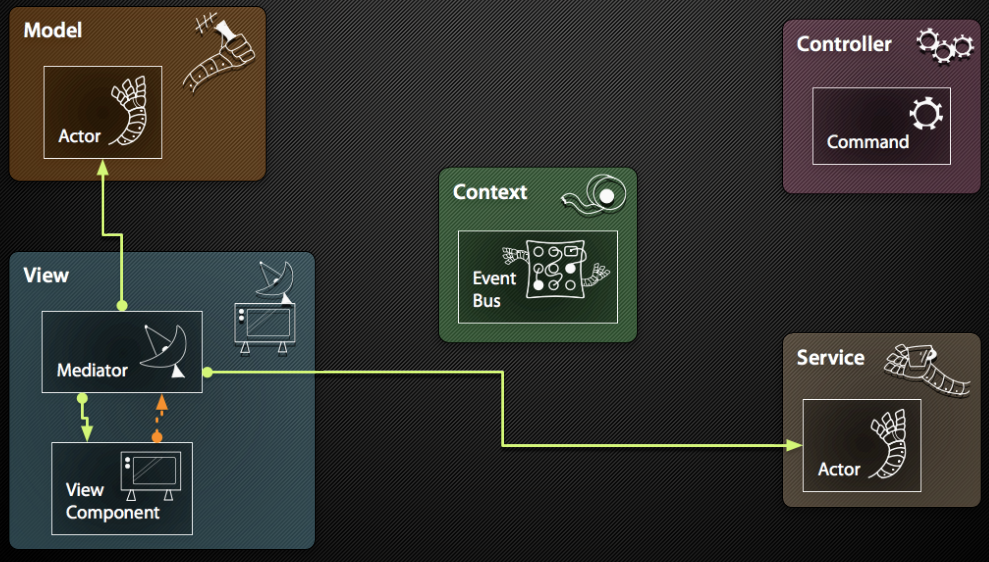
Mediators can access
Service and Model classes directly

View

Mediators can access
Service and Model classes directly

but this will couple the Mediator to the Actor

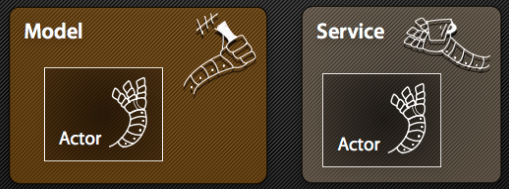
Accessing Models and Services from Mediators



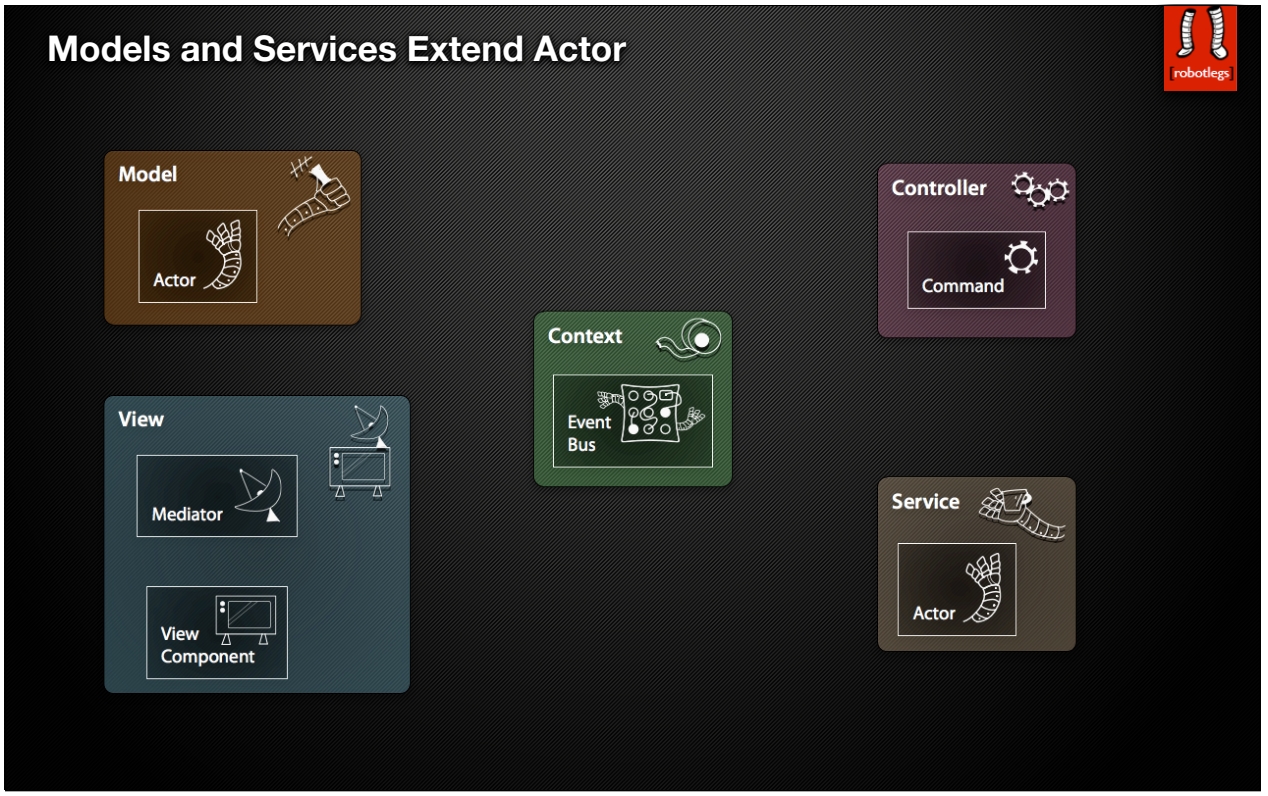
(tread carefully)

Actor

Actor is the base class for
Model and Service classes



Models and Services Extend Actor



Actor

eventDispatcher is injected
into Actor

Actor

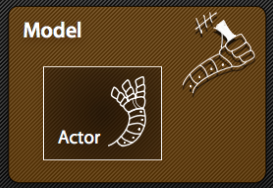
Actor provides a *dispatch(event)* method

Actor

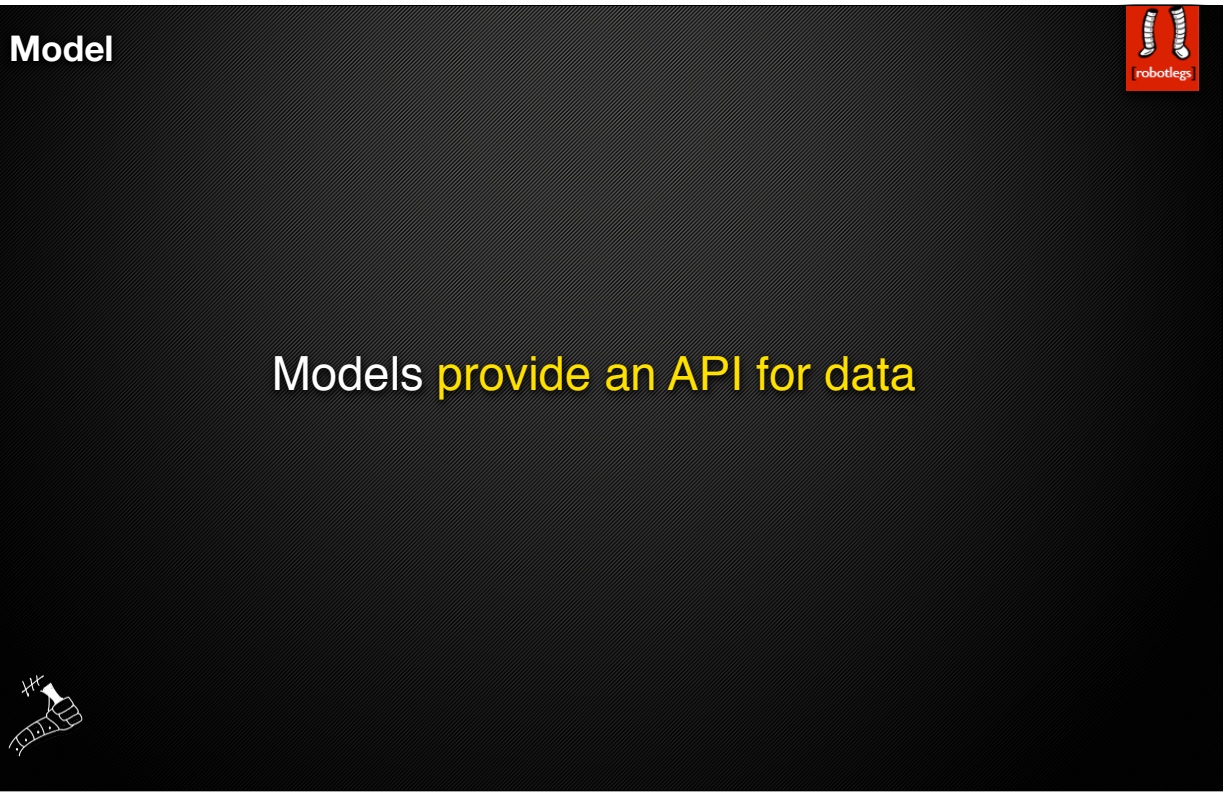
Actor is for your convenience

Model

Models extend Actor



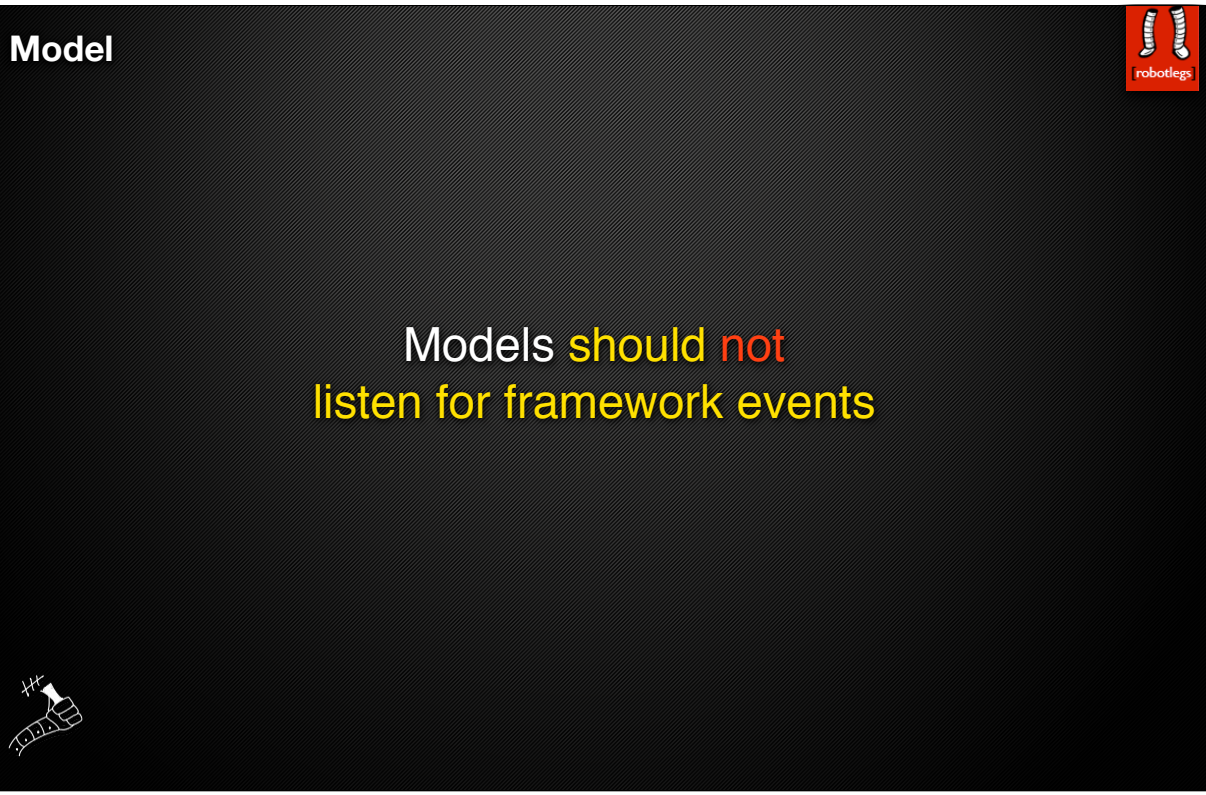
Model



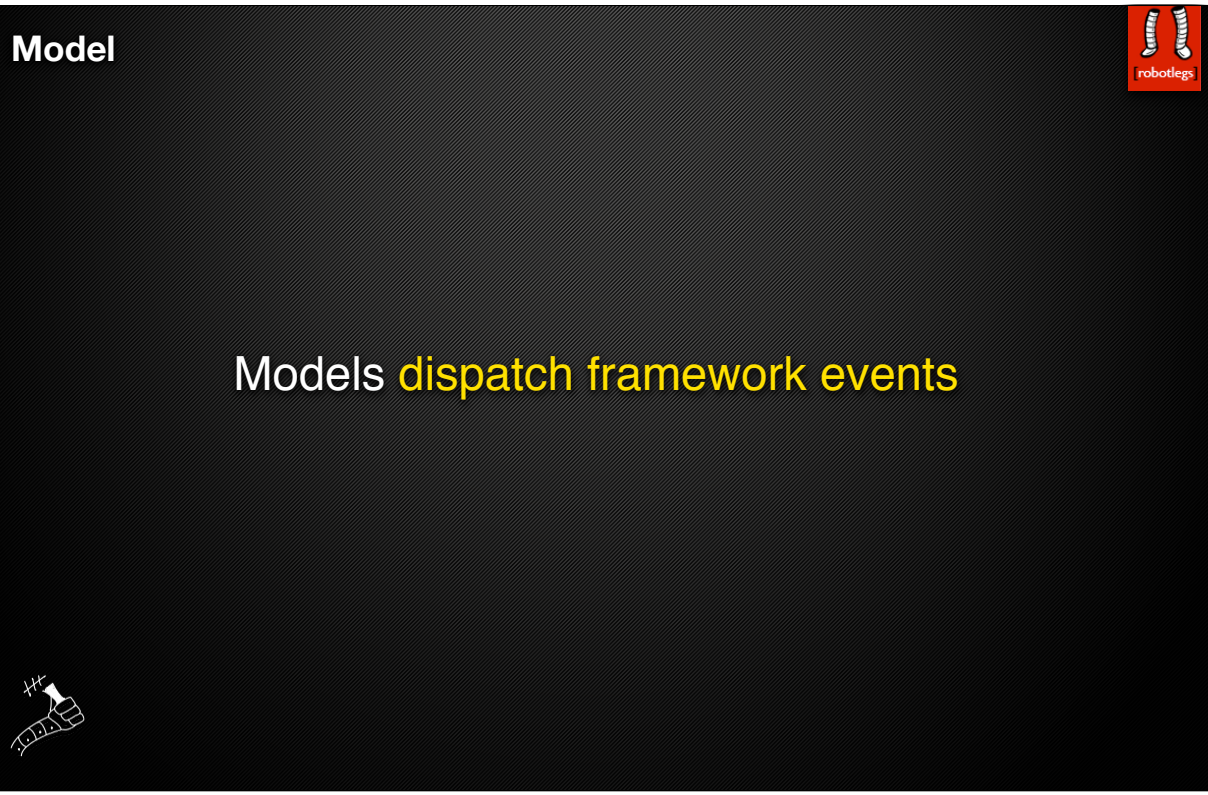
Model

Models sit between application data
and other actors

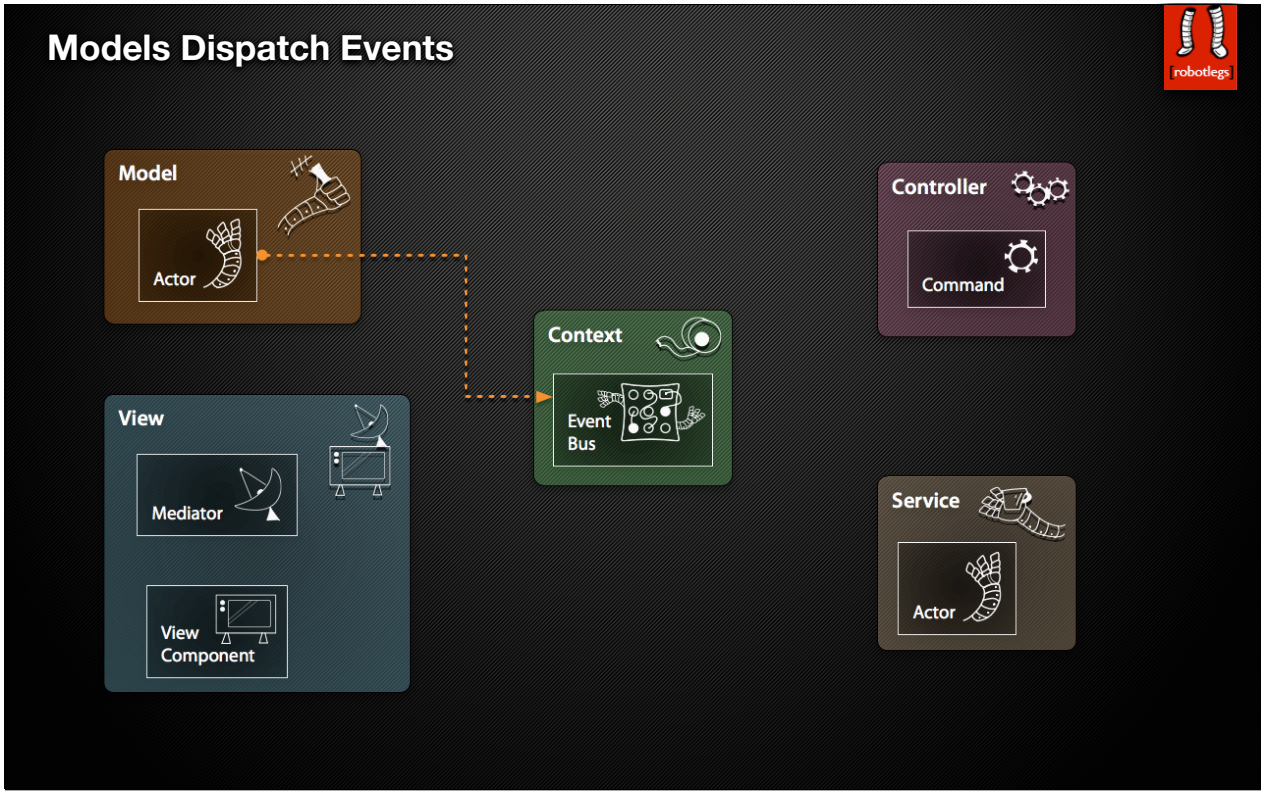
Model



Model



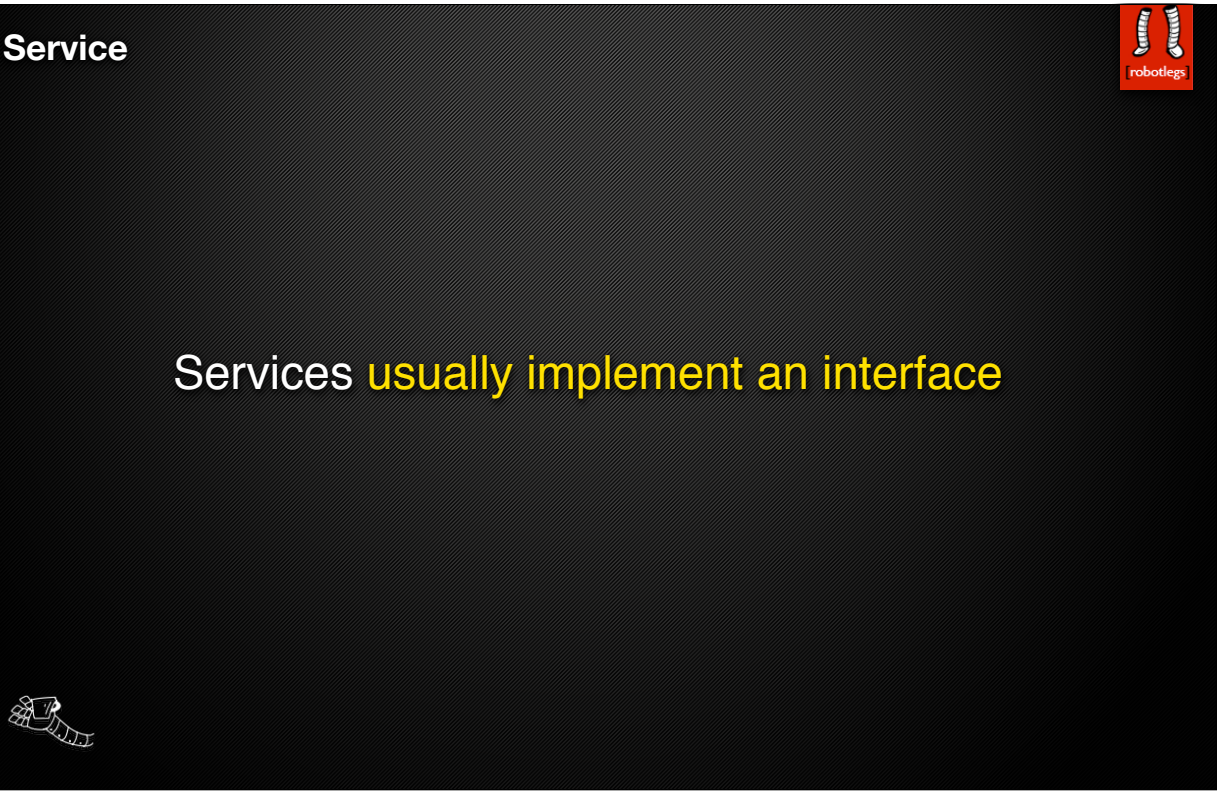
Models dispatch framework events


Models Dispatch Events
robotlegs
Model
Actor
Context
Event Bus
Controller
Command
View
Mediator
View Component
Service
Actor

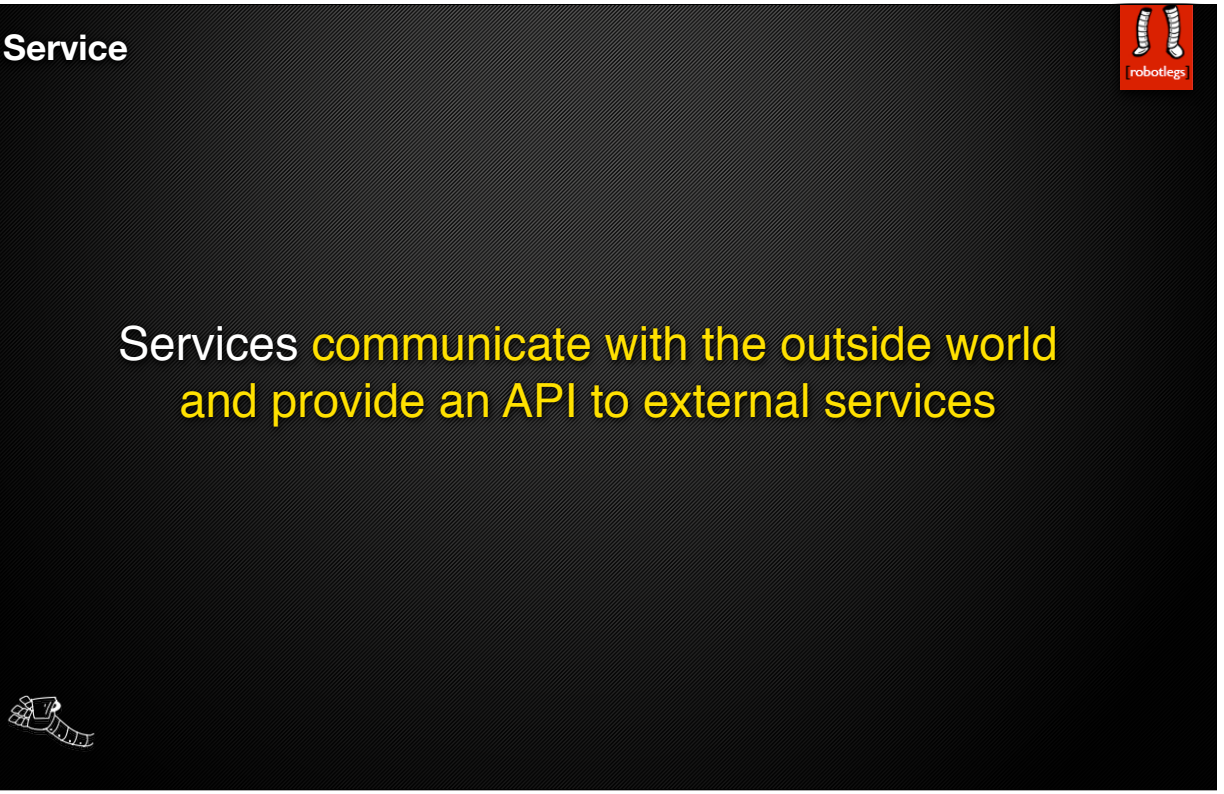
Service

Services extend Actor

Service

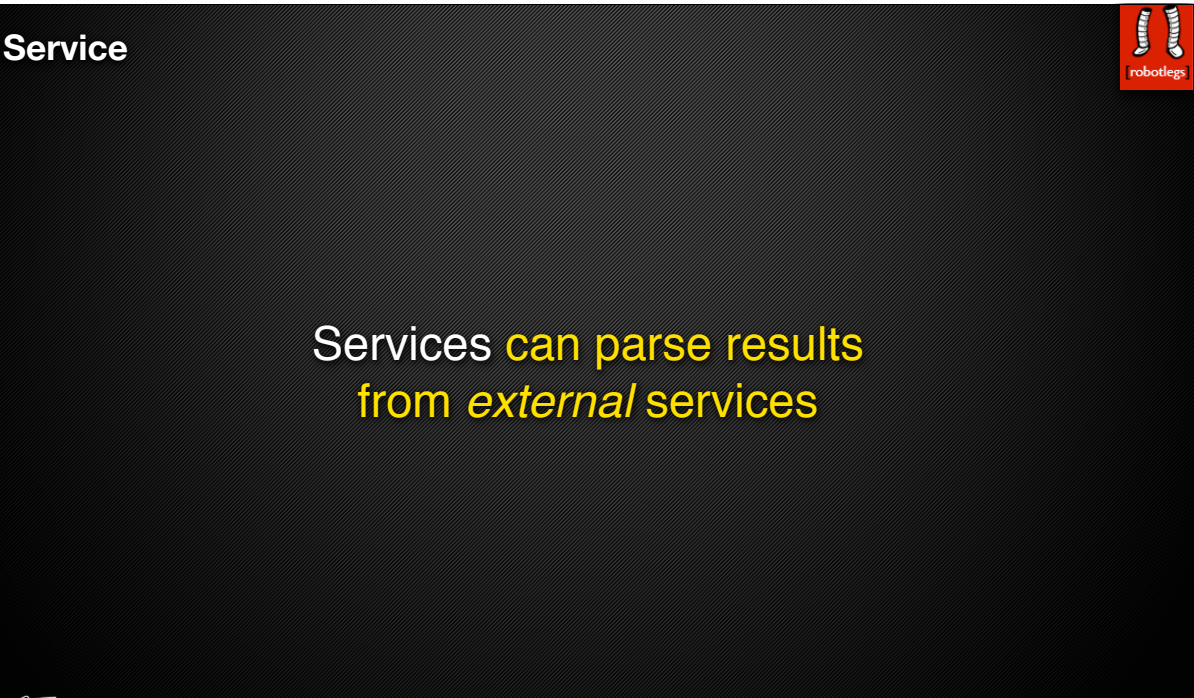


Service



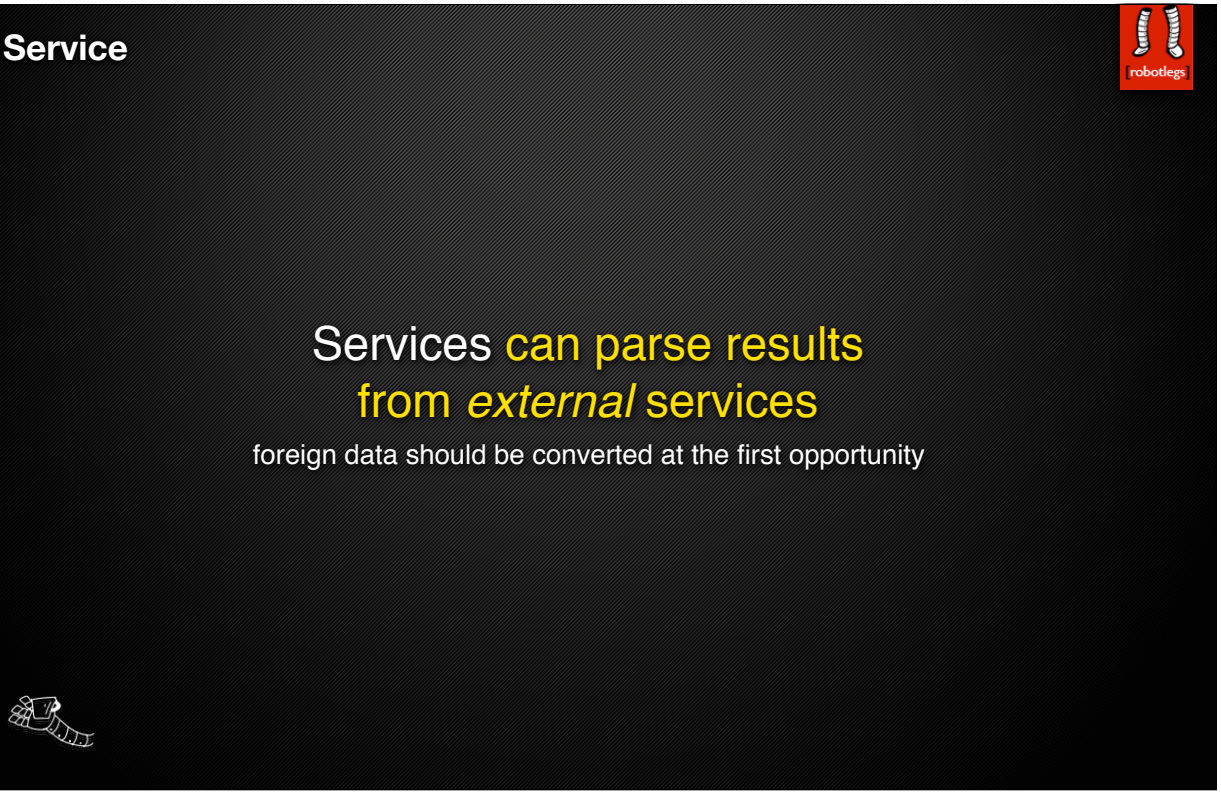
Services communicate with the outside world
and provide an API to external services

Service



Services can parse results
from *external* services

Service



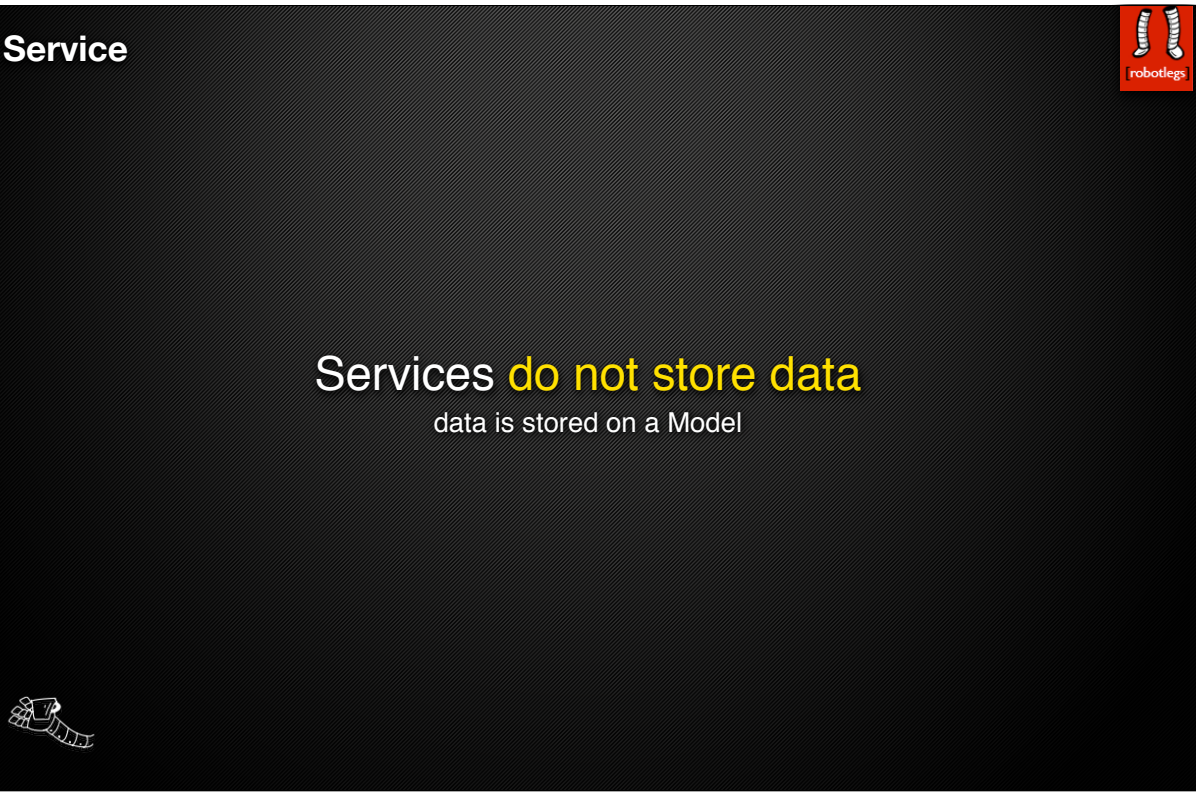
Services can parse results
from *external* services

foreign data should be converted at the first opportunity

Service

Services do not store data

Service



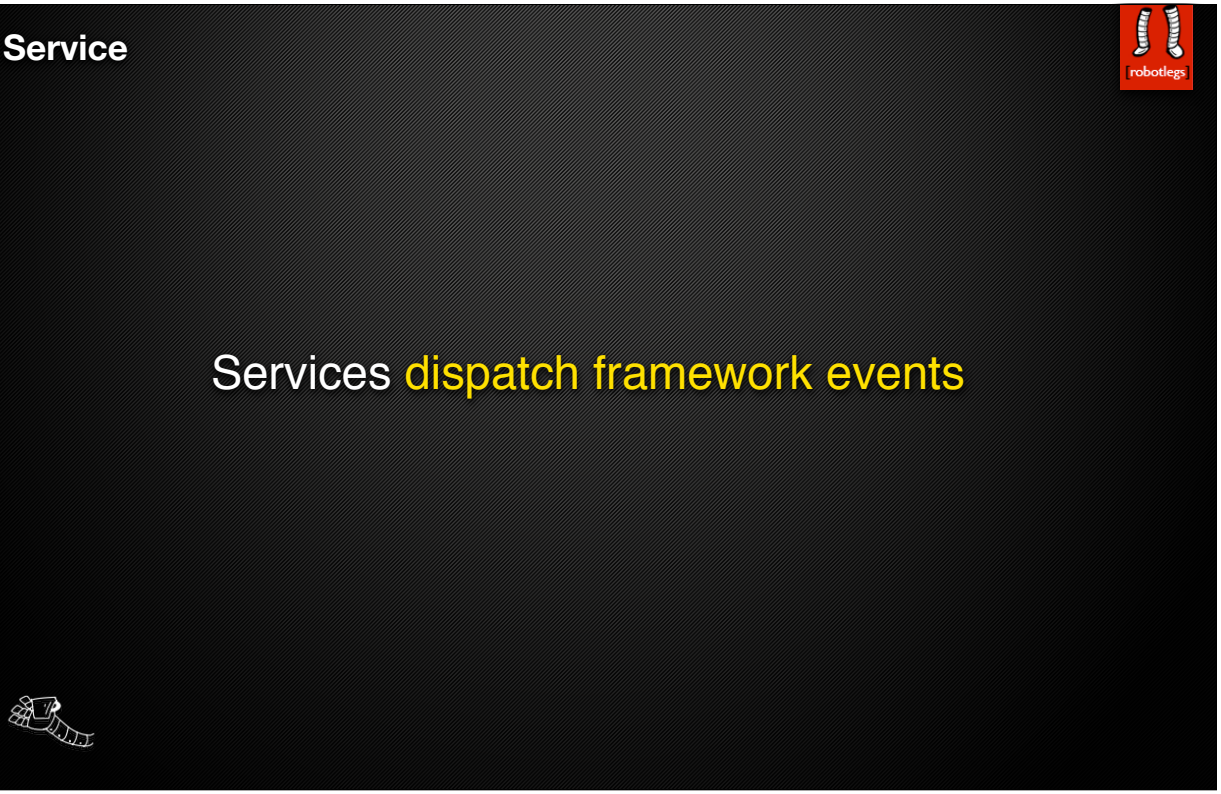
Services do not store data

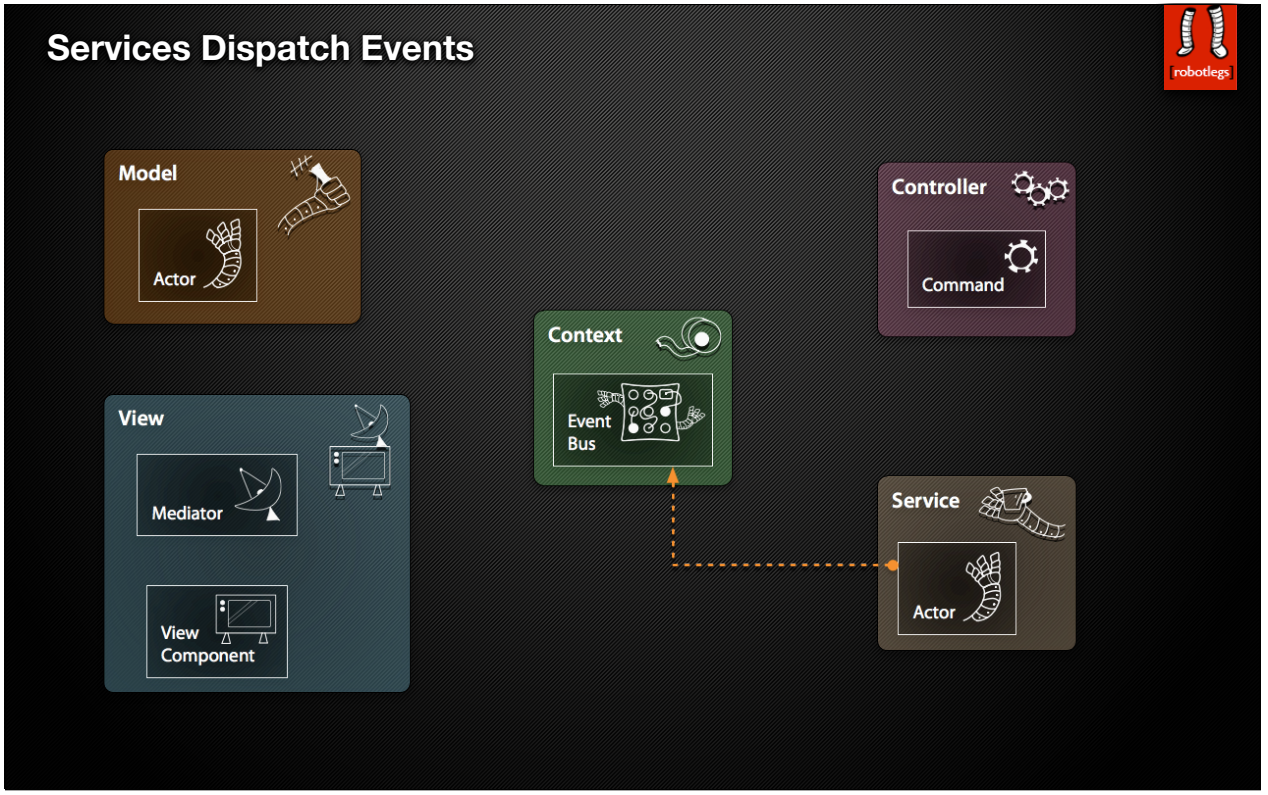
data is stored on a Model

Service

Services do not
receive framework events

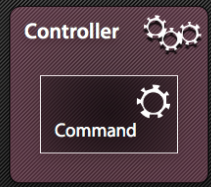
Service




Services Dispatch Events
robotlegs
Model
Actor
Controller
Command
Context
Event
Bus
View
Mediator
View
Component
Service
Actor

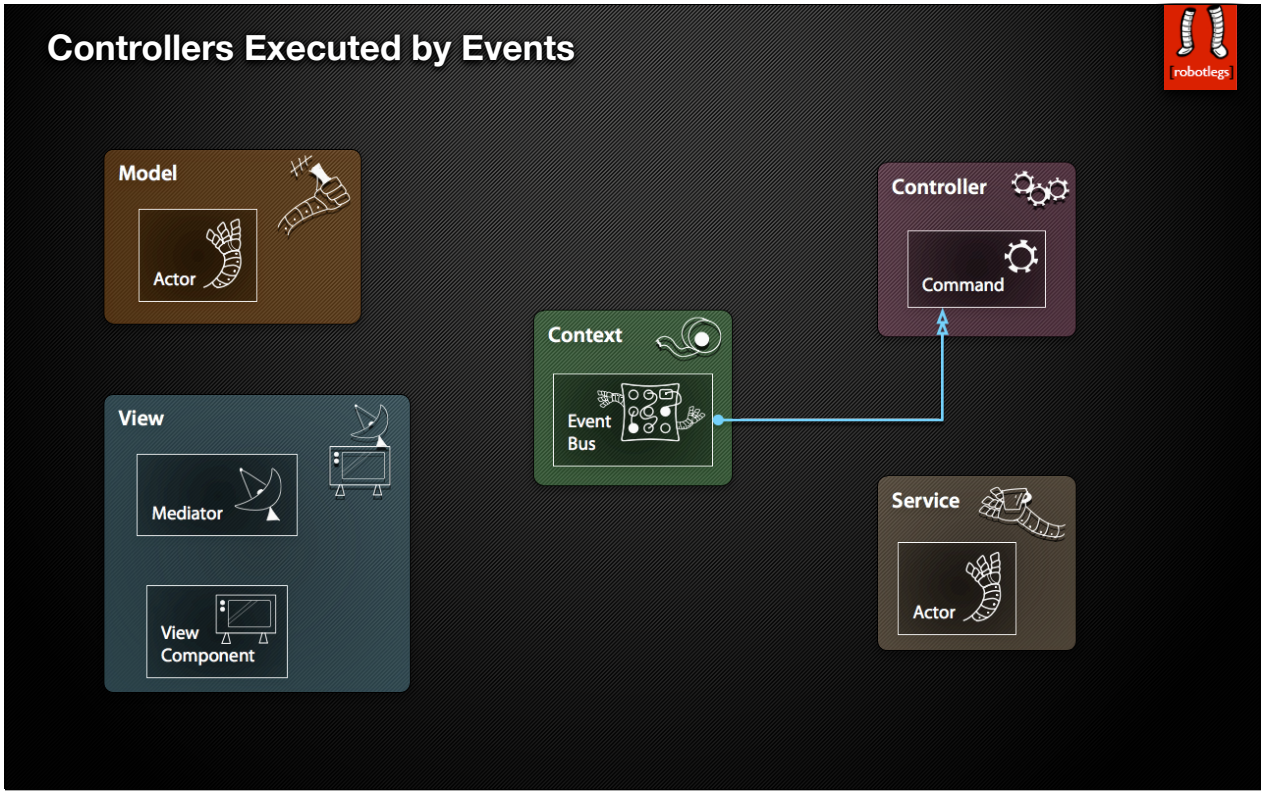
Controller

represented by the Command class



Controller

Commands are executed in response to framework events


Controllers Executed by Events
robotlegs
Model
Actor
View
Mediator
View Component
Context
Event Bus
Controller
Command
Service
Actor

Controller

Commands are stateless

Controller

Commands are stateless

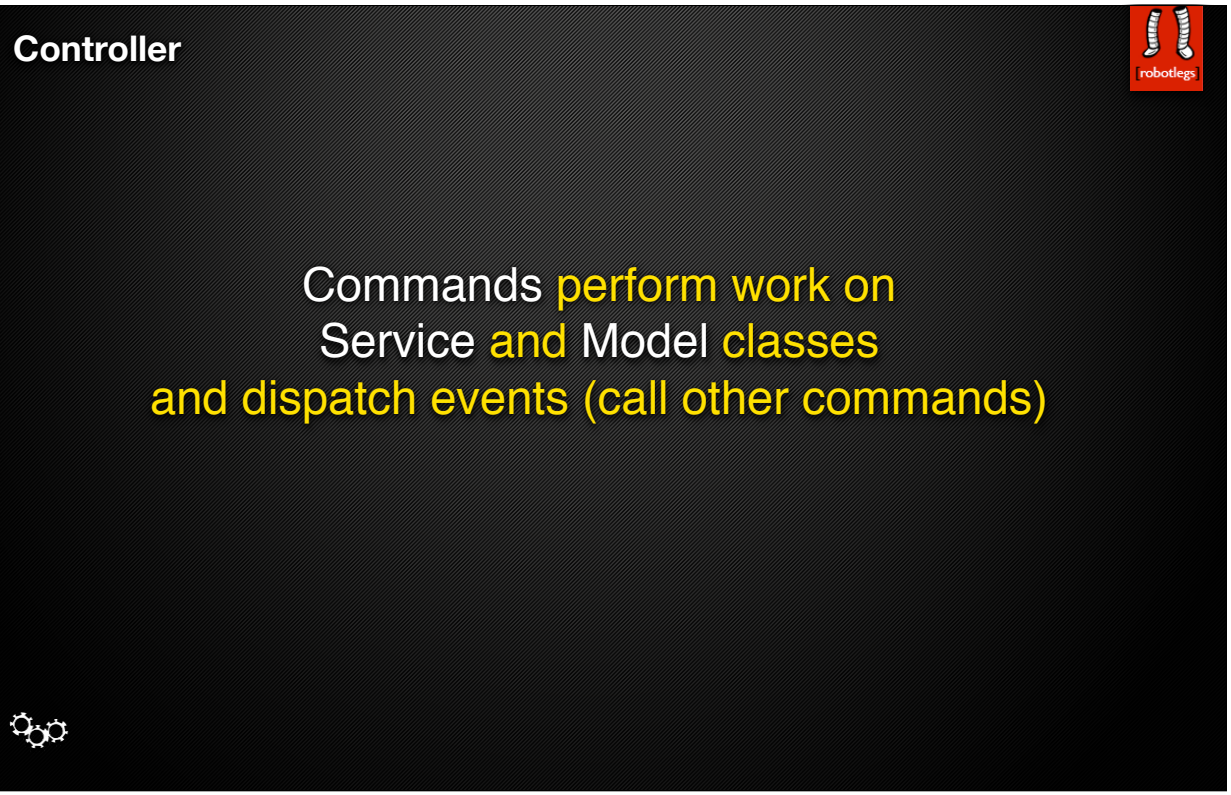
they execute and die

Controller

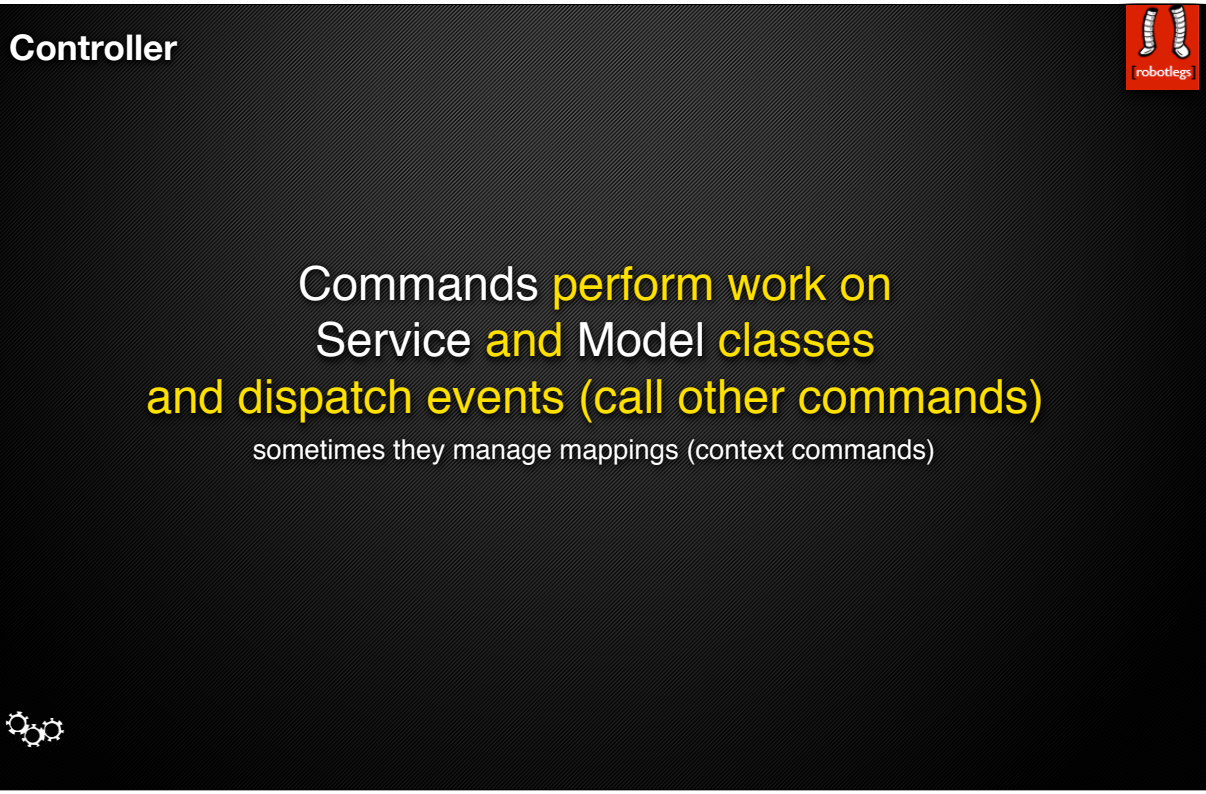
Commands are stateless

they execute and die
performing a single unit of work

Controller


[robotlegs]
Commands perform work on
Service and Model classes
and dispatch events (call other commands)

Controller

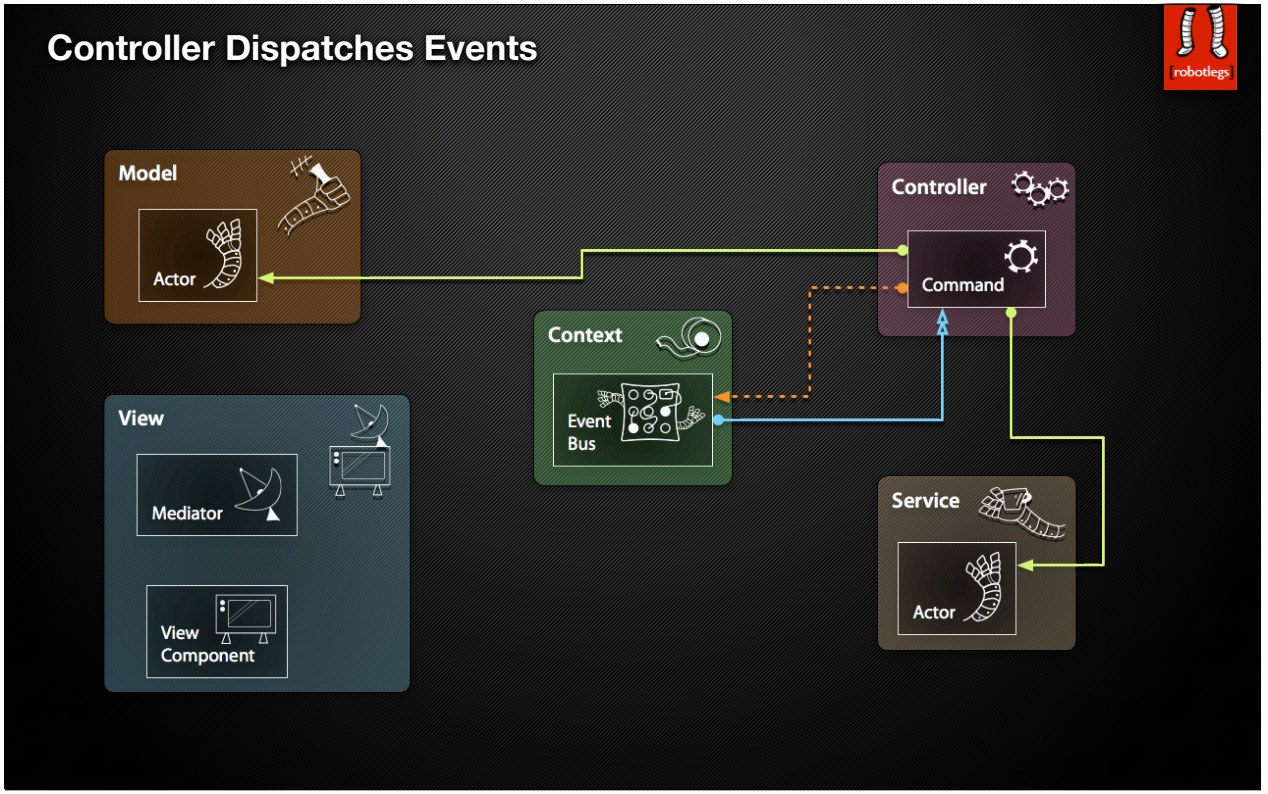

[robotlegs]
Commands perform work on
Service and Model classes
and dispatch events (call other commands)
sometimes they manage mappings (context commands)

Controller

Commands receive data from the events that trigger them

Controller

Commands dispatch framework events


Controller Dispatches Events
robotlegs
Model
Actor
Controller
Command
Context
Event
Bus
View
Mediator
View
Component
Service
Actor

Controller

Commands do not receive framework events

Controller

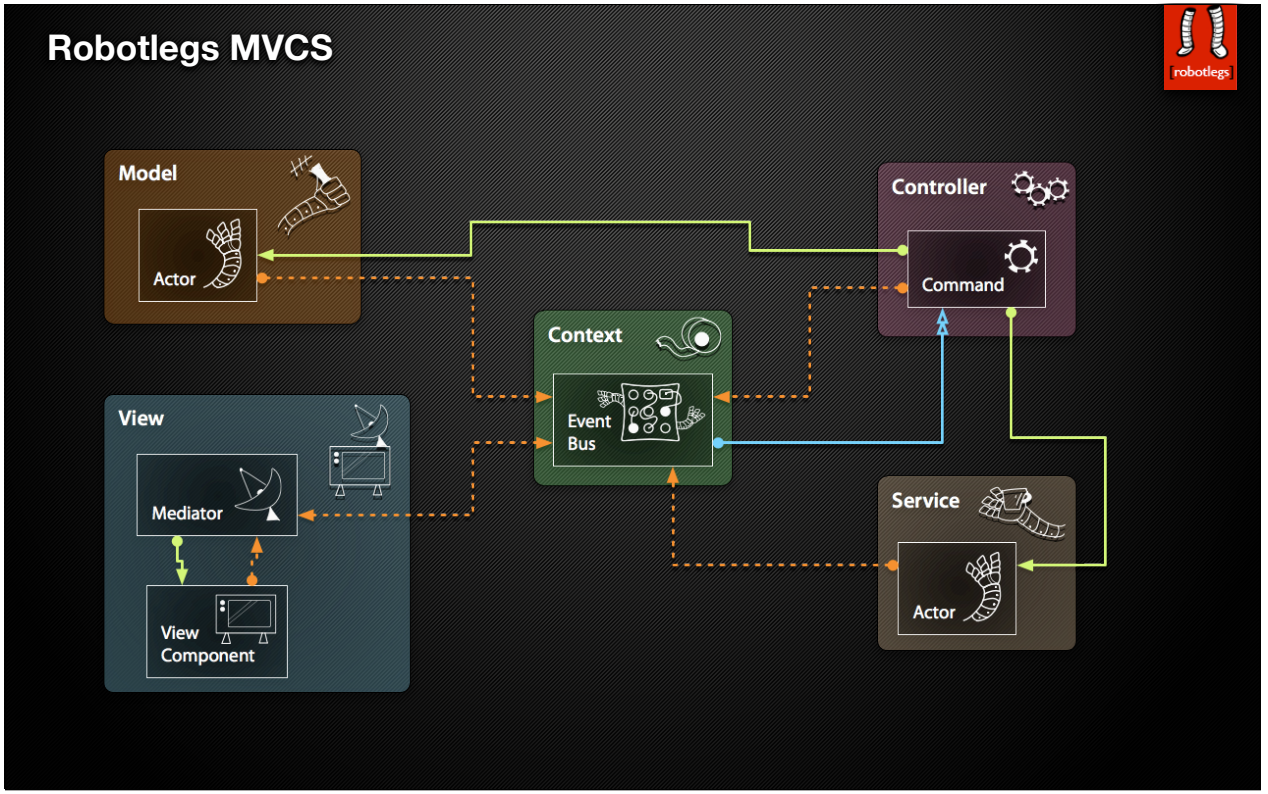
Commands do not receive framework events

outside of the event that triggers them

Controller

Commands do not receive framework events

outside of the event that triggers them
which is available for injection


Robotlegs MVCS
[robotlegs]
Model
Actor
Controller
Command
Context
Event
Bus
View
Mediator
View
Component
Service
Actor


www.robotlegs.org
download the framework
best practices documentation
project on github
FAQ and Knowledge Base
live examples


[robotlegs]
Questions?