

hexMachina

urban framework for hardcore developers

W H O A M I ?

francis Bourre / Docler Holding
author of [@pixlib](#)

pixLib adventures

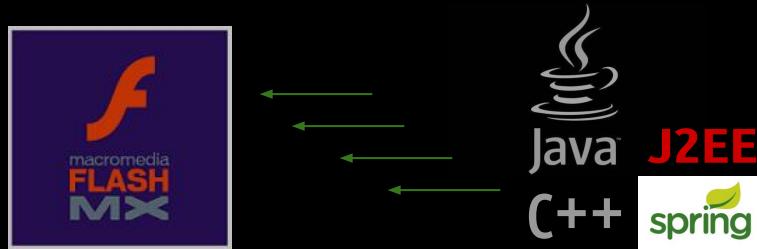
in the web apps jungles



Crédits : solkkete / DollarPhotoClub

pixLib adventures

in the web apps jungles



*“Let’s grab knowledge from
other Tech eco-systems”*

pixLib adventures

pixlib (as2) 2003

pixioc (as2) 2004

lowra (as3) 2007

palmer (as3) 2008

pixlib community version (as3) 2008

pixlib closed source version (as3) 2009

hexMachina (as3) 2015

Six Feet Under

2015

...



How can I target the browser now? with a statically typed language



TypeScript

No interface **@compile time**

A large, stylized red outline of the word "failure" is positioned diagonally across the slide. The letters are bold and have a slight drop shadow, set against a solid black background.



Works
with extra benefits

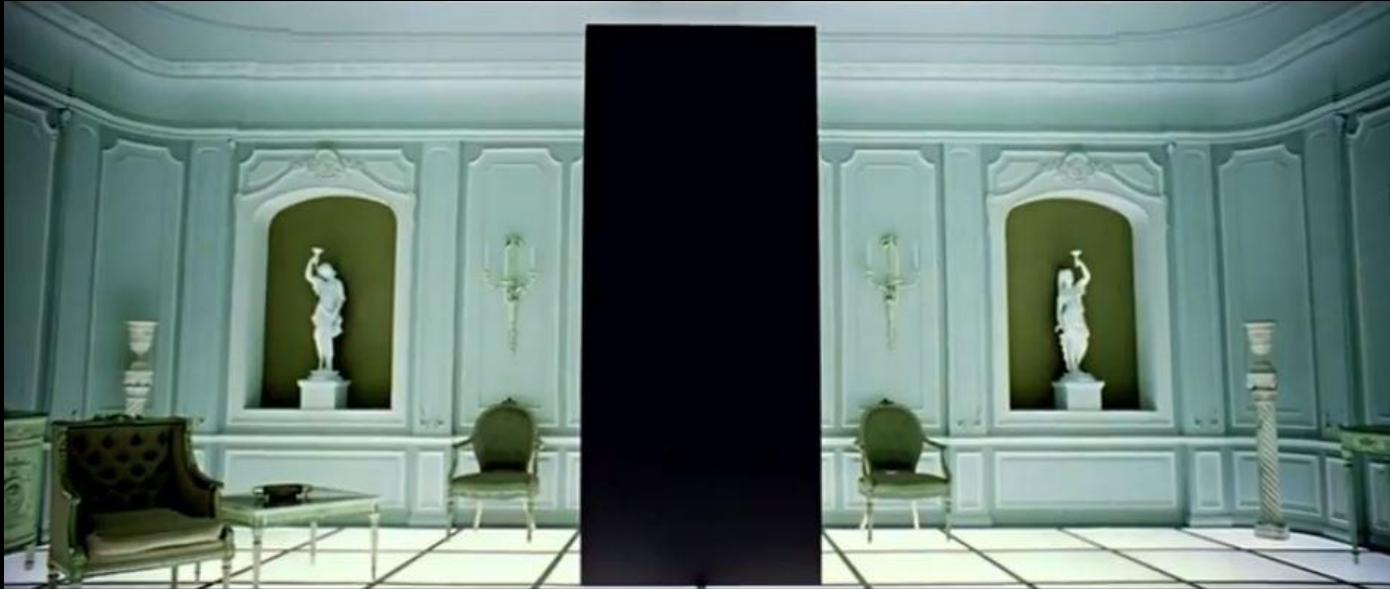
HAXE

hexMachina

Which kind of framework is it...

and what is the goal?

Monolith framework?



Monolith framework?



Many repositories available:

Core, Annotation, Inject, Service, MVC, IoC, State...

Golden Principles

for maintaining long lifecycle applications

Golden Principles

for maintaining long lifecycle applications

- secure collaboration

Golden Principles

for maintaining long lifecycle applications

- secure collaboration
- code reusability

Golden Principles

for maintaining long lifecycle applications

- secure collaboration
- code reusability
- low coupling

Secure collaboration

design by contract @compile time



INTERFACES

define contracts

- Formal specs of your components

INTERFACES

define contracts

- Formal specs of your components
- Hide the implementation

INTERFACES

define contracts

- Formal specs of your components
- Hide the implementation
- Easy team workflow

INTERFACES

define contracts

- Formal specs of your components
- Hide the implementation
- Easy team workflow
- And more...

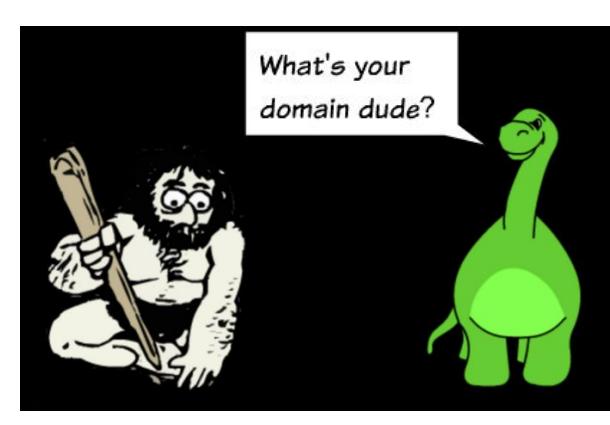
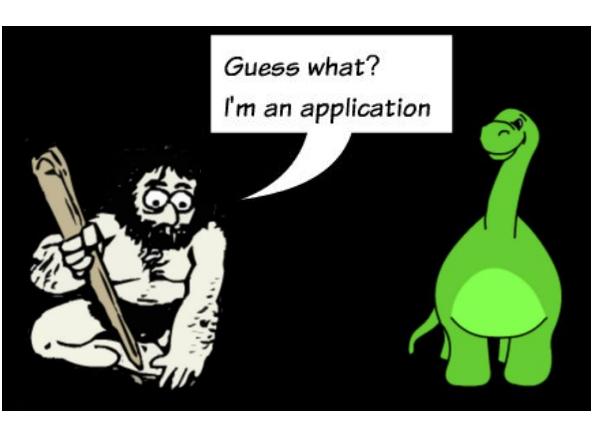
Code reusability

modular system and configurable components



MODULE

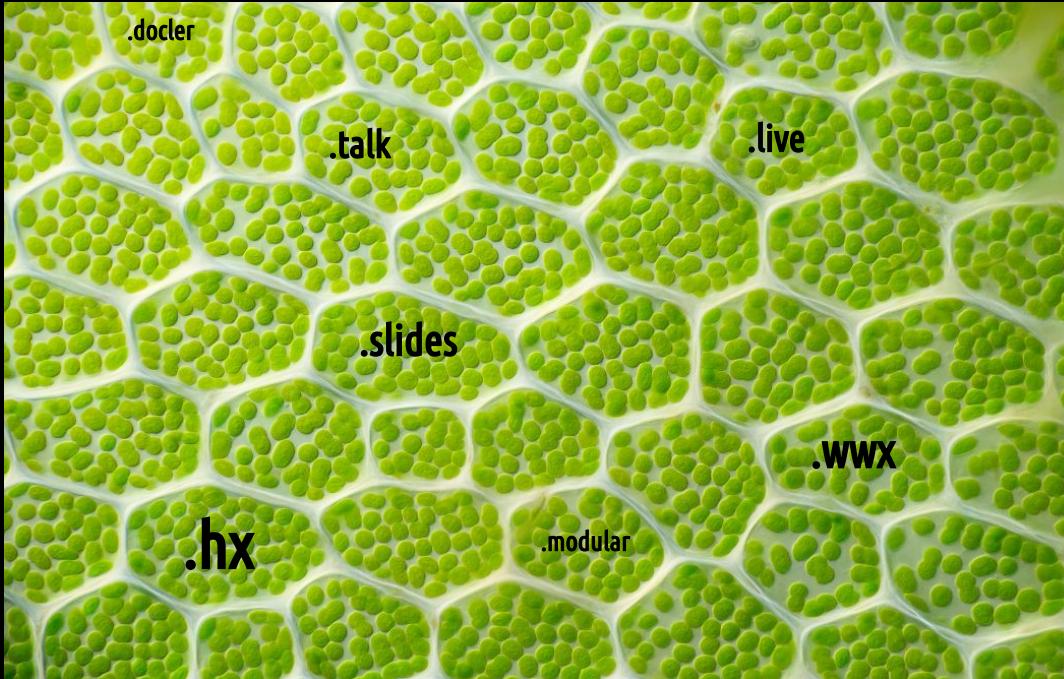
Every module is a micro-application



...And proud to be.

DOMAIN

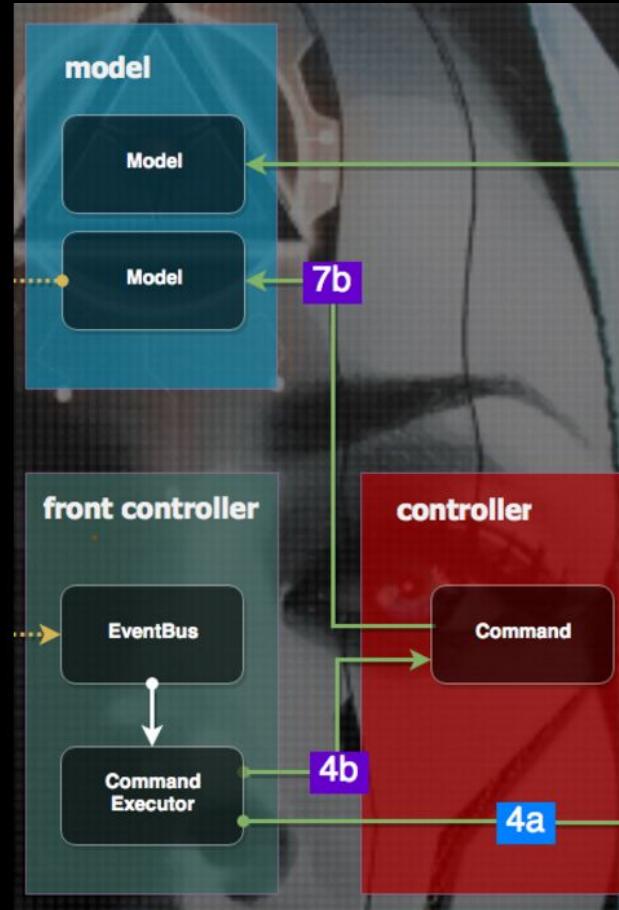
Every **module** belongs to an **unique domain**



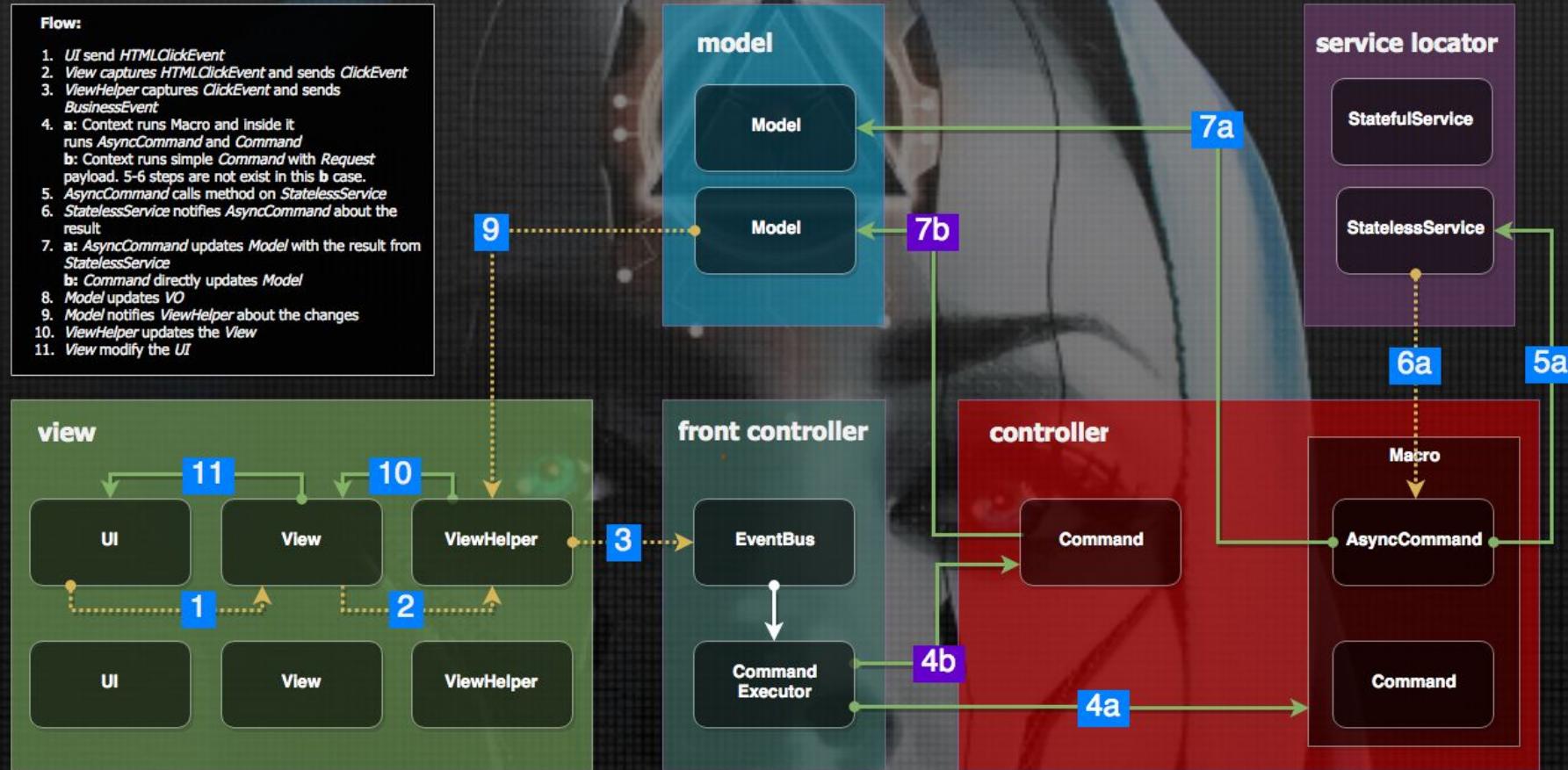
MODULE

micro-application

- unique **Domain**
- private **EventBus**
- **FrontController**
(and/or Controller/s)
- **Model/s**



hexMachina Module Flow Diagram



Low coupling

prefer runtime coupling - USE IoC

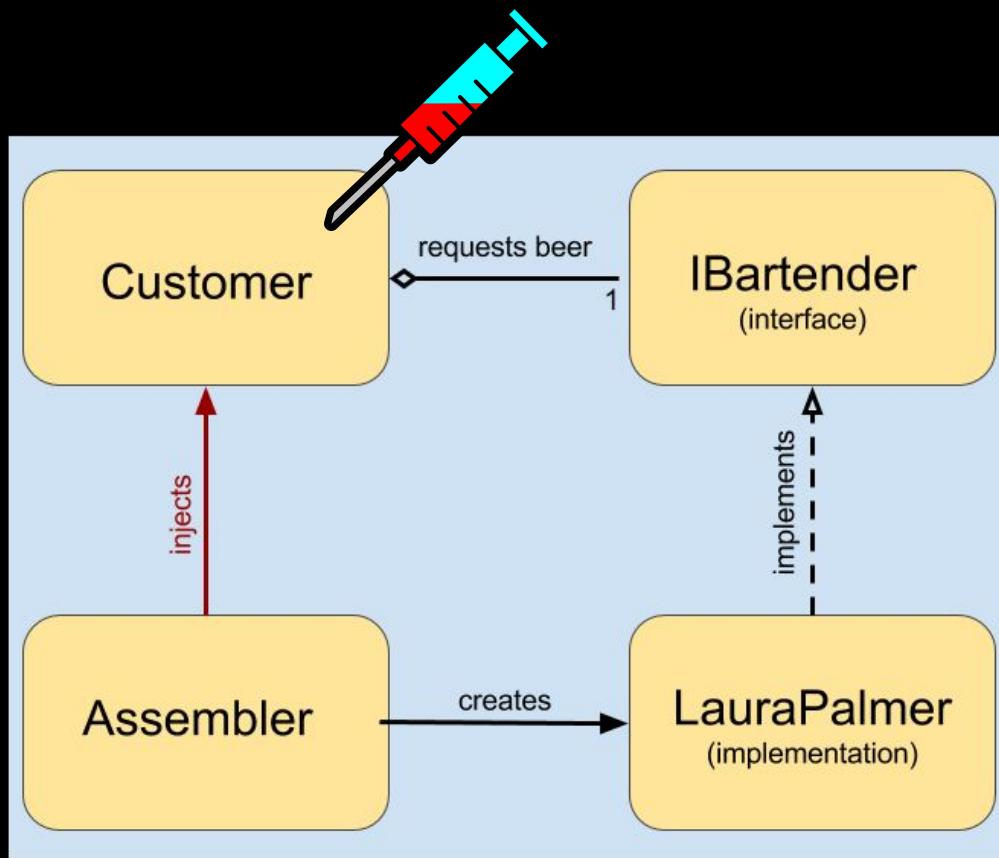


Hollywood principle

"Don't call us, we will call you"



Dependency Injection



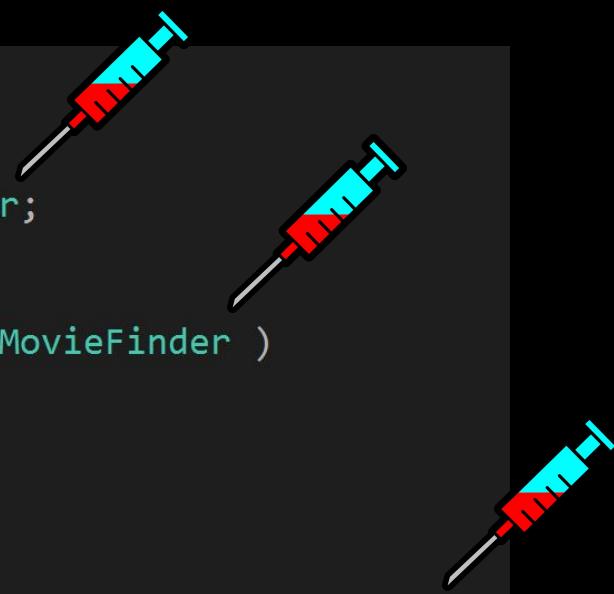
Automate coupling

Use automatic dependency injection



INJECTION with ANNOTATIONS

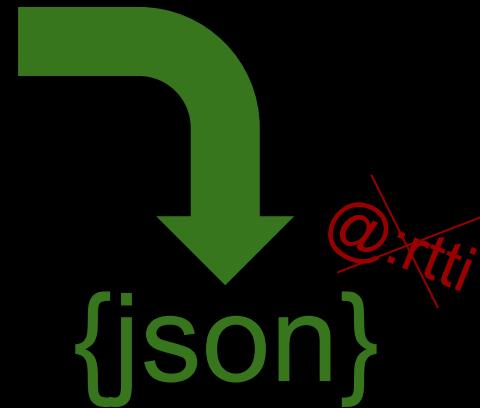
```
1 class MovieLister
2 {
3     @Inject
4     public var finder : IMovieFinder;
5
6     @Inject
7     public function new( finder : IMovieFinder )
8     {
9         //implementation
10    }
11
12    @Inject
13    public function setMovieFinder( finder : IMovieFinder )
14    {
15        //implementation
16    }
17 }
```



hexAnnotation

Do dirty reflection job @compile time

```
1 class LoadPhotosCommand
2     implements IInjectorContainer
3 {
4     @Inject
5     public var photosService : IGetPhotosService;
6
7     @Inject
8     public var galleryModel : IGalleryModel;
9 }
```



```
LoadPhotosCommand.__meta__ = { obj : { 'hex.di.IInjectorContainer' : ["\\"ctor\\"", {"args\":[], "isPre":false, "name\":"new"}, {"isPost":false, "order":0}, "props": [{"isOpt":false, "name": "photosService", "type": "IGetPhotosService"}, {"key": "\\""}, {"isOpt":false, "name": "galleryModel", "type": "IGalleryModel"}, {"key": "\\"}], "name": "LoadPhotosCommand", "methods": []} };
```

hexInject

- Works with annotations: Properties, methods and constructors

hexInject

- Works with annotations: Properties, methods and constructors
- Mapping by type and optionally by name

hexInject

- Works with annotations: Properties, methods and constructors
- Mapping by type and optionally by name
- Value, class instance or singleton can satisfy dependencies

hexInject

- Works with annotations: Properties, methods and constructors
- Mapping by type and optionally by name
- Value, class instance or singleton can satisfy dependencies
- Injections can be optional

hexInject

- Works with annotations: Properties, methods and constructors
- Mapping by type and optionally by name
- Value, class instance or singleton can satisfy dependencies
- Injections can be optional
- Injectors chaining

hexInject

- Works with annotations: Properties, methods and constructors
- Mapping by type and optionally by name
- Value, class instance or singleton can satisfy dependencies
- Injections can be optional
- Injectors chaining
- Event-driven and Exception handling

hexInject

- Works with annotations: Properties, methods and constructors
- Mapping by type and optionally by name
- Value, class instance or singleton can satisfy dependencies
- Injections can be optional
- Injectors chaining
- Event-driven and Exception handling
- Supports ordered @PostConstruct and @PreDestroy

Build configurable architecture

use Domain Specific Language

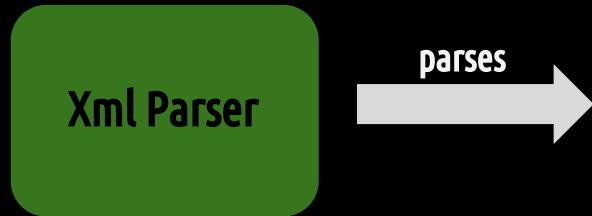


hexloC

Injections with DSL

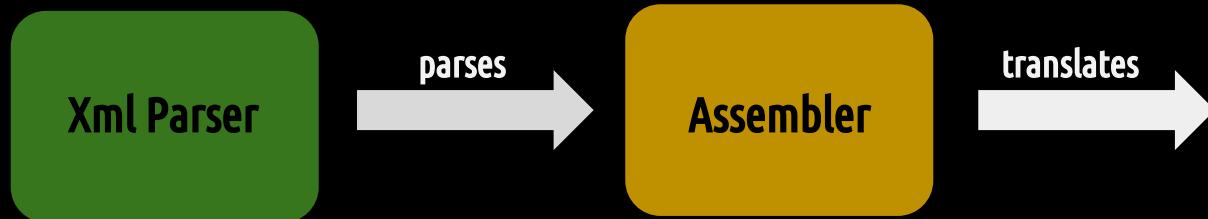
```
1 <root name="applicationContext">
2
3     <service id="movieFinder" type="services.MovieFinder"/>
4
5     <module id="movieLister" type="modules.MovieLister">
6         <argument ref="movieFinder">
7     </module>
8
9 </root>
```

DSL PARSING/ASSEMBLING FLOW



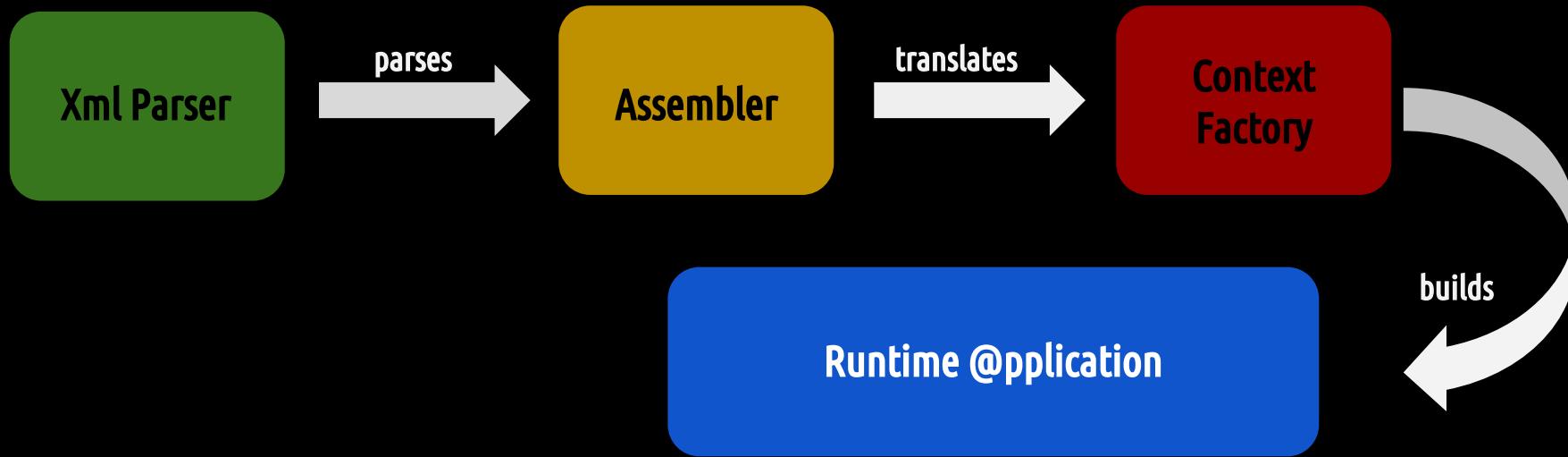
```
public function new()
{
    XmlReader.readXmlFile( "example/configuration/context.xml" );
}
```

DSL PARSING/ASSEMBLING FLOW



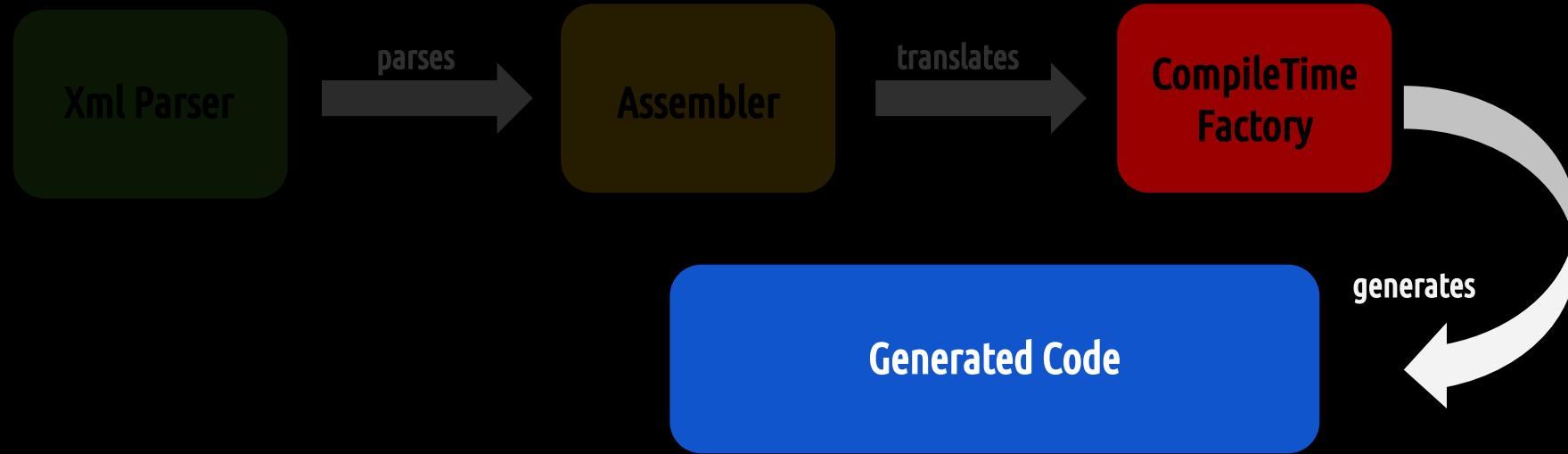
```
public function new()
{
    XmlReader.readXmlFile( "example/configuration/context.xml" );
}
```

DSL PARSING/ASSEMBLING FLOW



```
public function new()
{
    XmlReader.readXmlFile( "example/configuration/context.xml" );
}
```

DSL PARSING/COMPIILING FLOW



```
public function new()
{
    XmlCompiler.readXmlFile( "example/configuration/context.xml" );
}
```

HEXMACHINA DSL

hello world

```
1<root name="applicationContext">
2  <test id="s" value="hello world"/>
3</root>
```

HEXMACHINA DSL

String is default type

```
1<root name="applicationContext">
2  <test id="s" value="hello world"/>
3  <test id="i" type="Int" value="-3"/>
4</root>
```

HEXMACHINA DSL

simple module creation

```
1<root>
2    <!-- MODULES -->
3    <module id="gallery" type="example.module.gallery.GalleryModule">
4        <argument ref="flickerServiceLocator"/>
5    </module>
6</root>
```

SERVICES

service locator / services mapping

```
1<root>
2  <serviceLocator id="flickerServiceLocator" type="hex.config.stateful.ServiceLocator">
3    <item>
4      <key type="Class" value="example.module.gallery.service.IGetPhotosService"/>
5      <value type="Class" value="example.service.flickr.FlickrGetPhotos"/>
6    </item>
7  </serviceLocator>
8</root>
```

HEXMACHINA DSL

constructor injection by DSL

```
1<root>
2    <!-- MODULES -->
3    <module id="gallery" type="example.module.gallery.GalleryModule">
4        <argument ref="flickerServiceLocator"/>
5    </module>
6</root>
```



After compilation

JavaScript is generated

```
var example_FlickrExample = $hx_exports.FlickrExample = function() {
    console.log("XmlCompiler starts compilation..."); █
    var applicationAssembler = new hex_ioc_assembler_ApplicationAssembler(); █
    var applicationContext = applicationAssembler.getApplicationContext("flickr");
    var coreFactory = applicationContext.getCoreFactory(); █
    var flickerServiceLocator = new hex_config_stateful_ServiceLocator(); █
    flickerServiceLocator.addService(example_module_gallery_service_IGetPhotosService,example_service_flickr_FlickrGetPhotos,null); █
    coreFactory.register("flickerServiceLocator",flickerServiceLocator); █
    var gallery = new example_module_gallery_GalleryModule(flickerServiceLocator);
    coreFactory.register("gallery",gallery); █
    applicationAssembler; █
}; █
```

INCLUDE

recursive files inclusion

```
1<root name="flickr">
2  <include file="example/configuration/ModuleConfiguration.xml"/>
3  <include file="example/configuration/ServiceConfiguration.xml"/>
4</root>
```

DSL CHECKING @compile time

Fail early, fail loudly



DSL CHECKING @compile time

type not found

```
2  <root>
3      <!-- MODULES -->
4      <module id="gallery" type="example.module.gallery.GalleryModules">
5          <argument ref="flickerServiceLocator"/>
6      </module>
7
8  </root>
```

ModuleConfiguration.xml

! 4 characters 22-26 : XmlCompiler parsing error with 'module' node, 'example.module.gallery.GalleryModules' type not found.

DSL CHECKING @compile time

missing ID

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <root>
3   <!-- MODULES -->
4   <module type="example.module.gallery.GalleryModule">
5     <argument ref="flickerServiceLocator"/>
6   </module>
7 </root>
```

ModuleConfiguration.xml



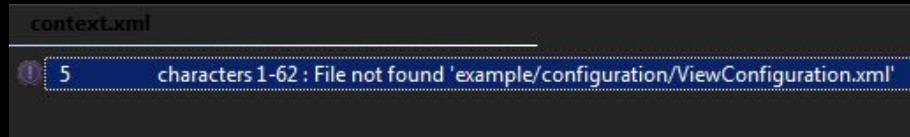
4

characters 2-8 : XmlCompiler parsing error with 'module' node, 'id' attribute not found.

DSL CHECKING @compile time

file missing

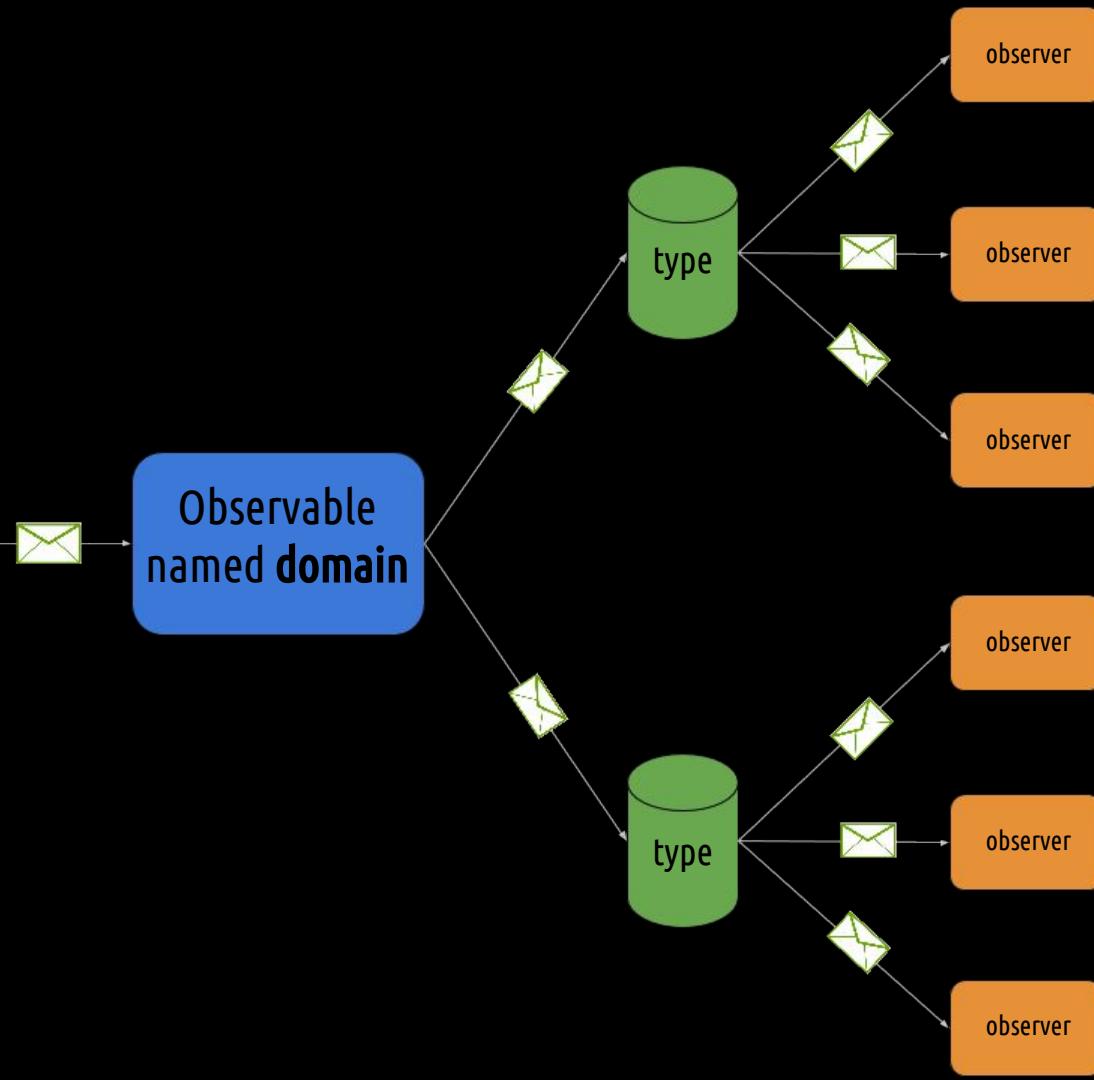
```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <root name="flickr">
3   <include file="example/configuration/ModuleConfiguration.xml"/>
4   <include file="example/configuration/ServiceConfiguration.xml"/>
5   <include file="example/configuration/ViewConfiguration.xml"/>
6 </root>
```



Low coupling

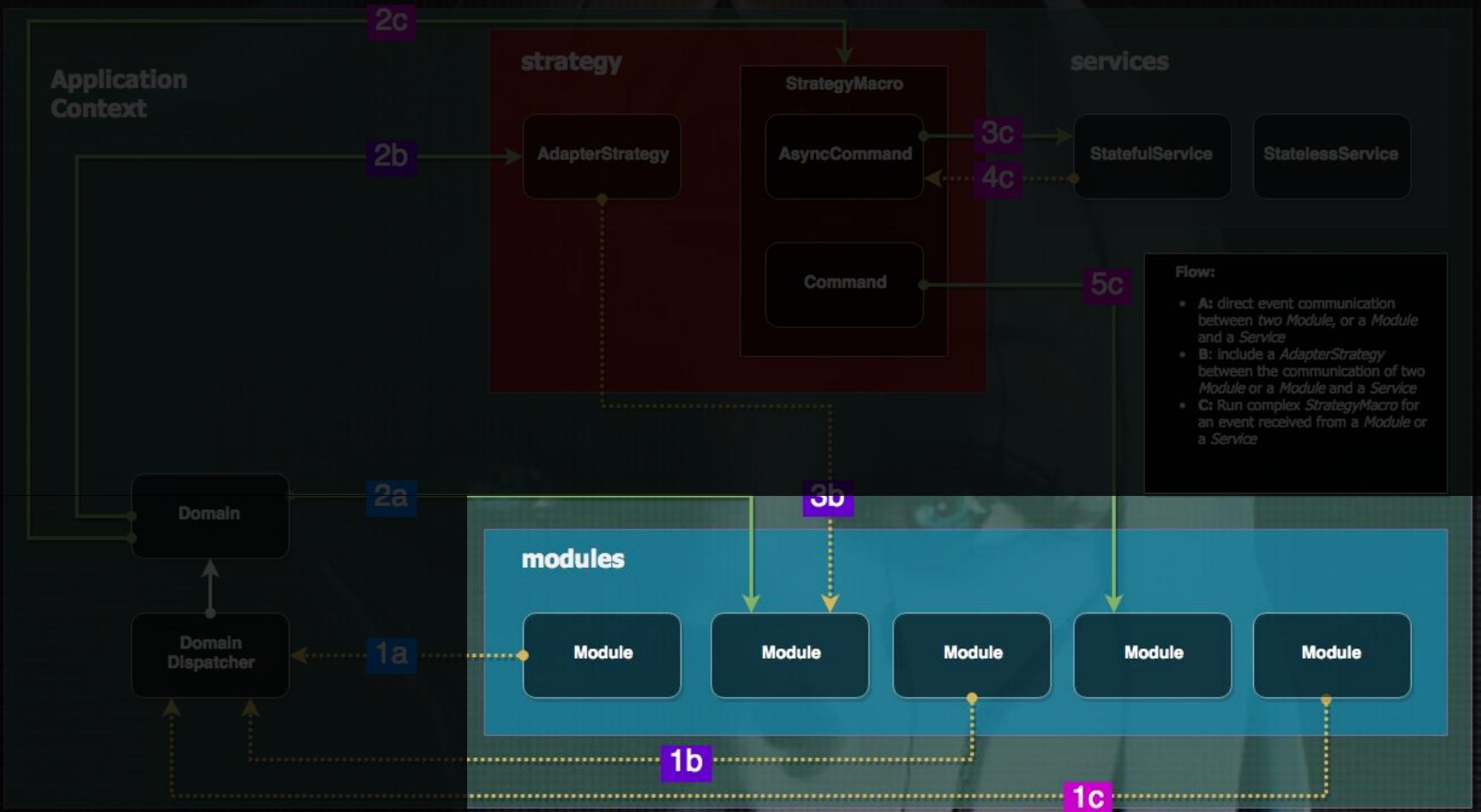
Use messaging



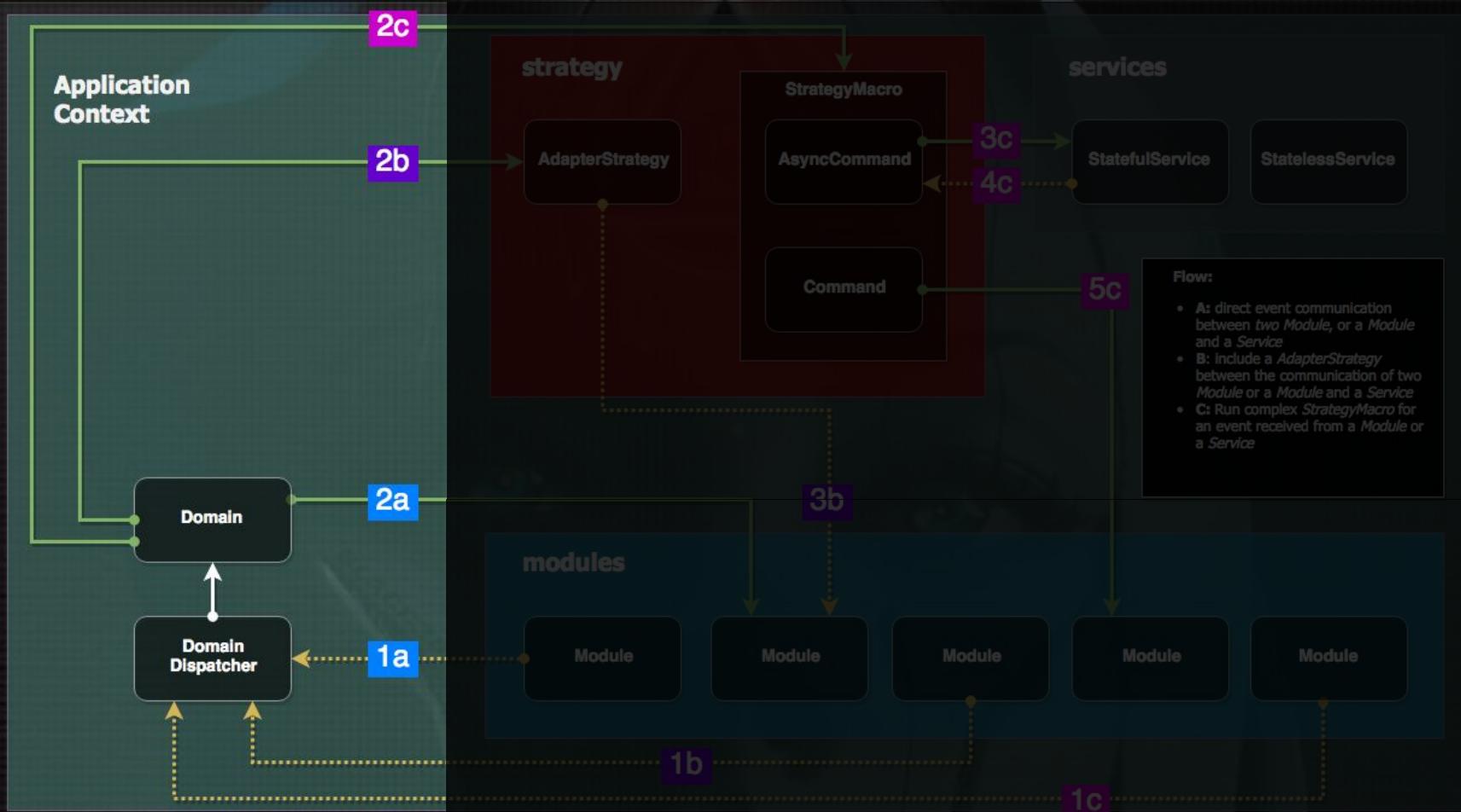


**Observer pattern
with domain messaging**

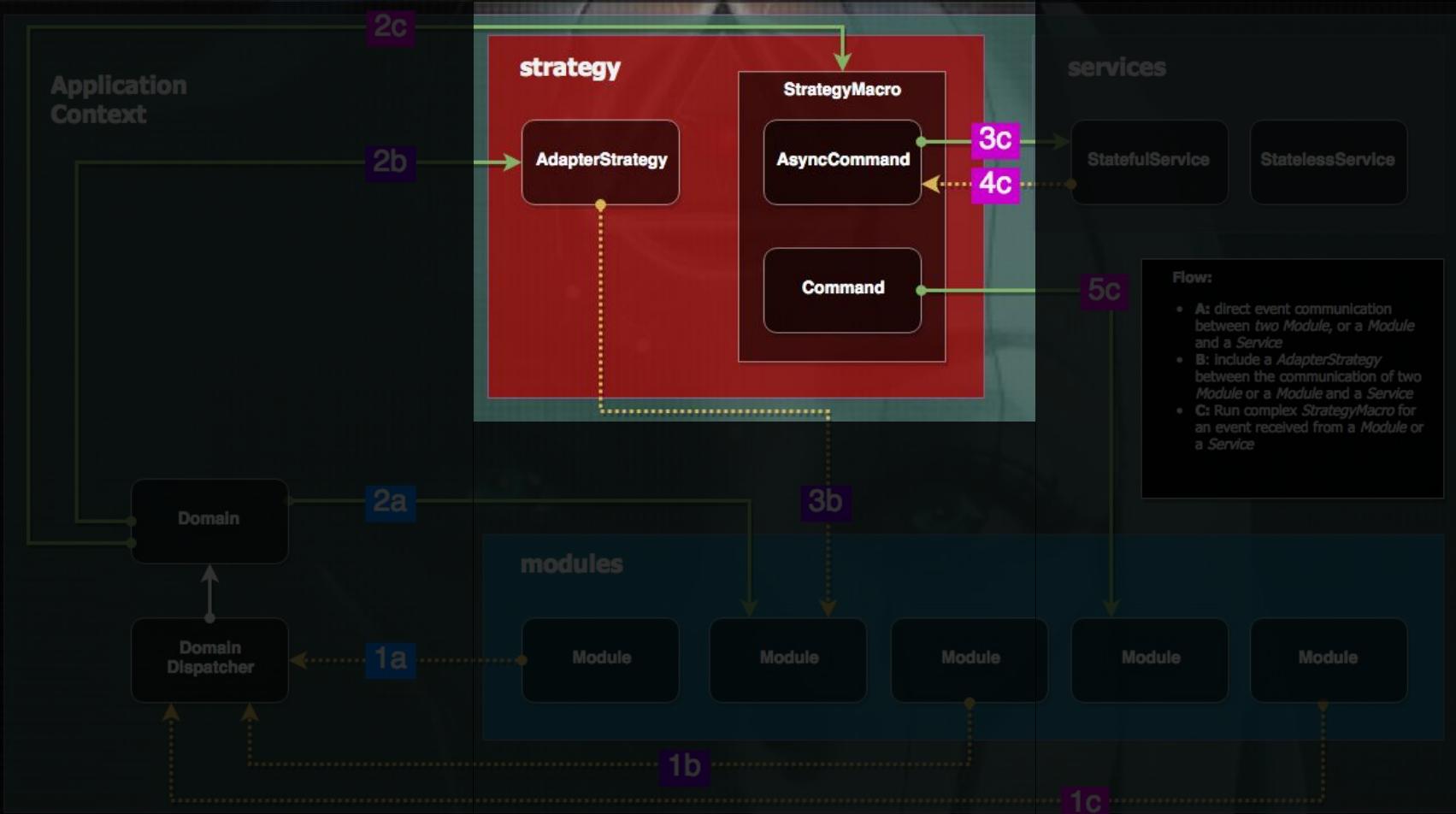
hexMachina ApplicationContext Flow Diagram



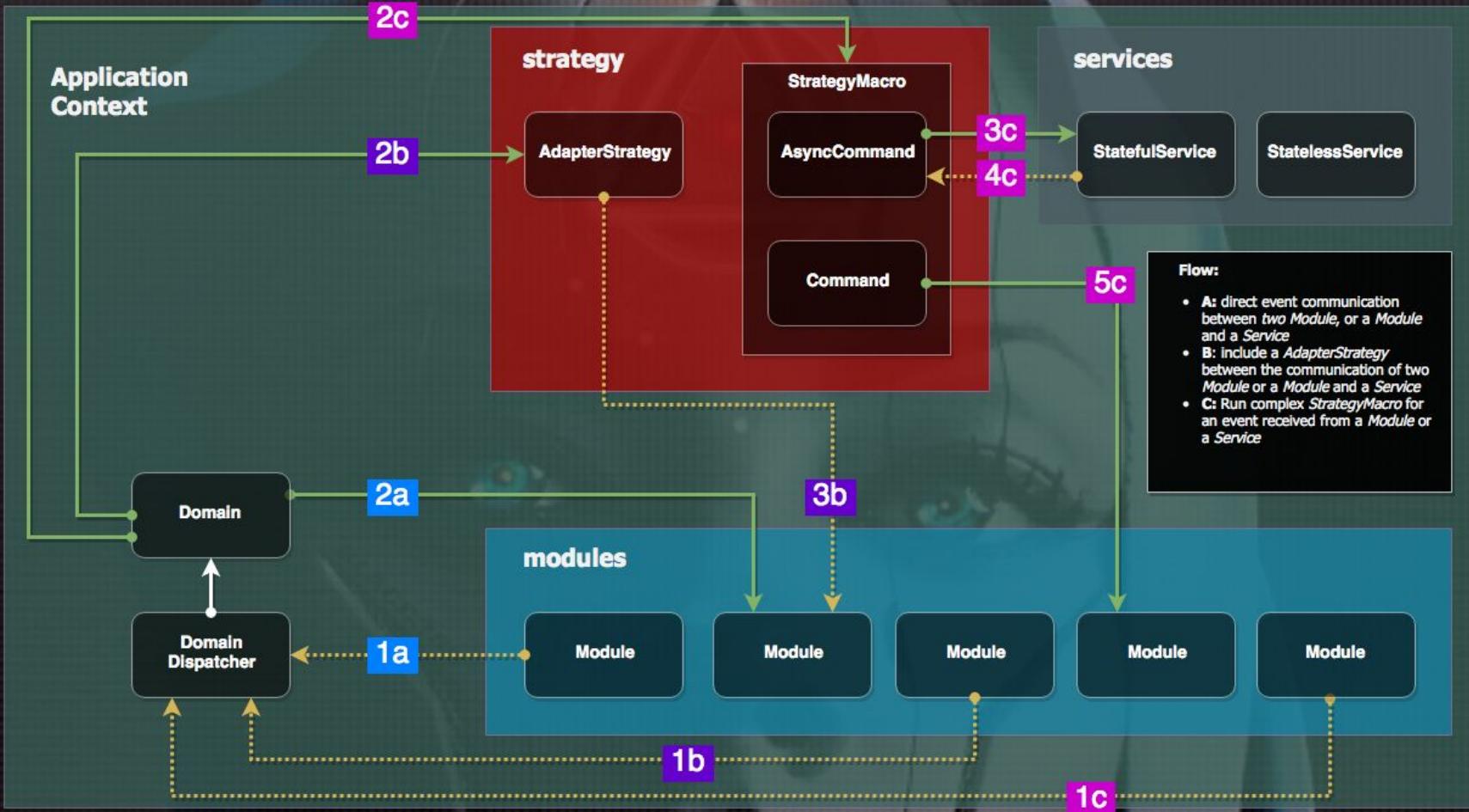
hexMachina ApplicationContext Flow Diagram



hexMachina ApplicationContext Flow Diagram



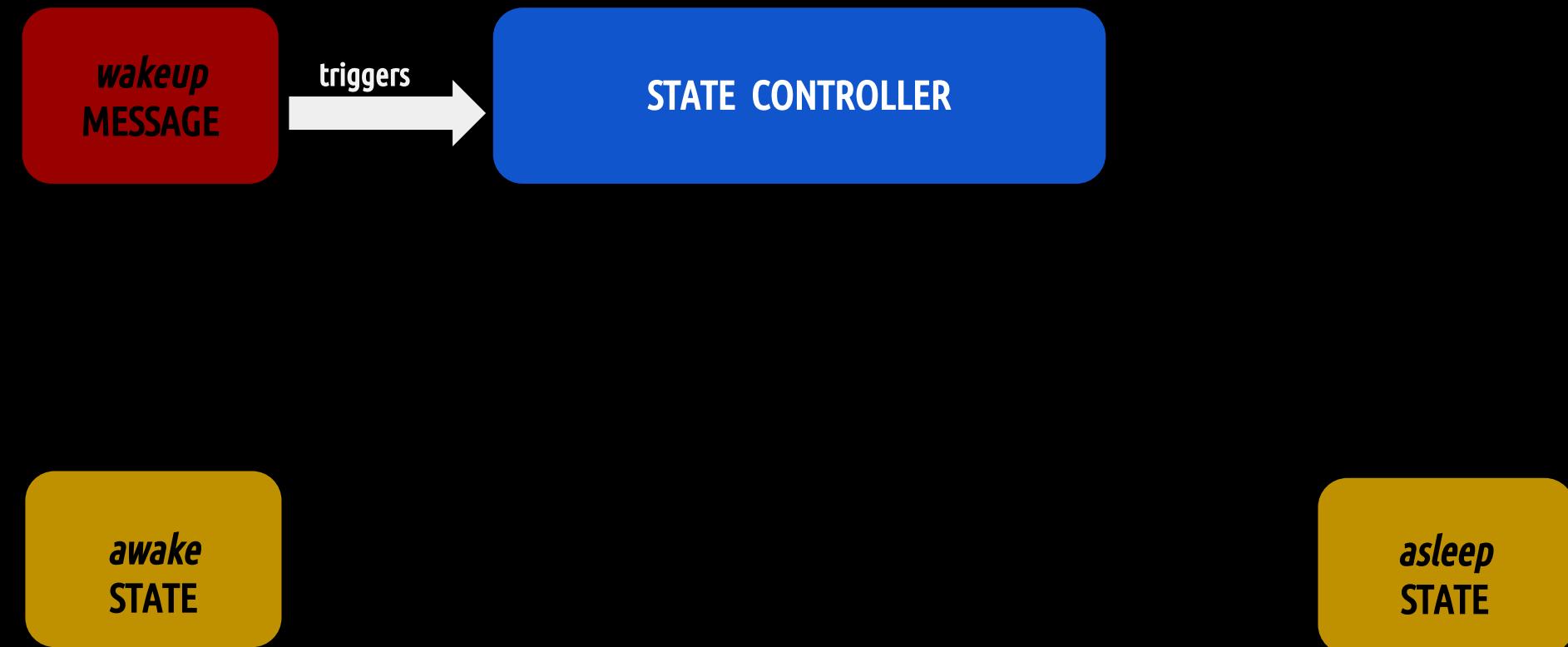
hexMachina ApplicationContext Flow Diagram



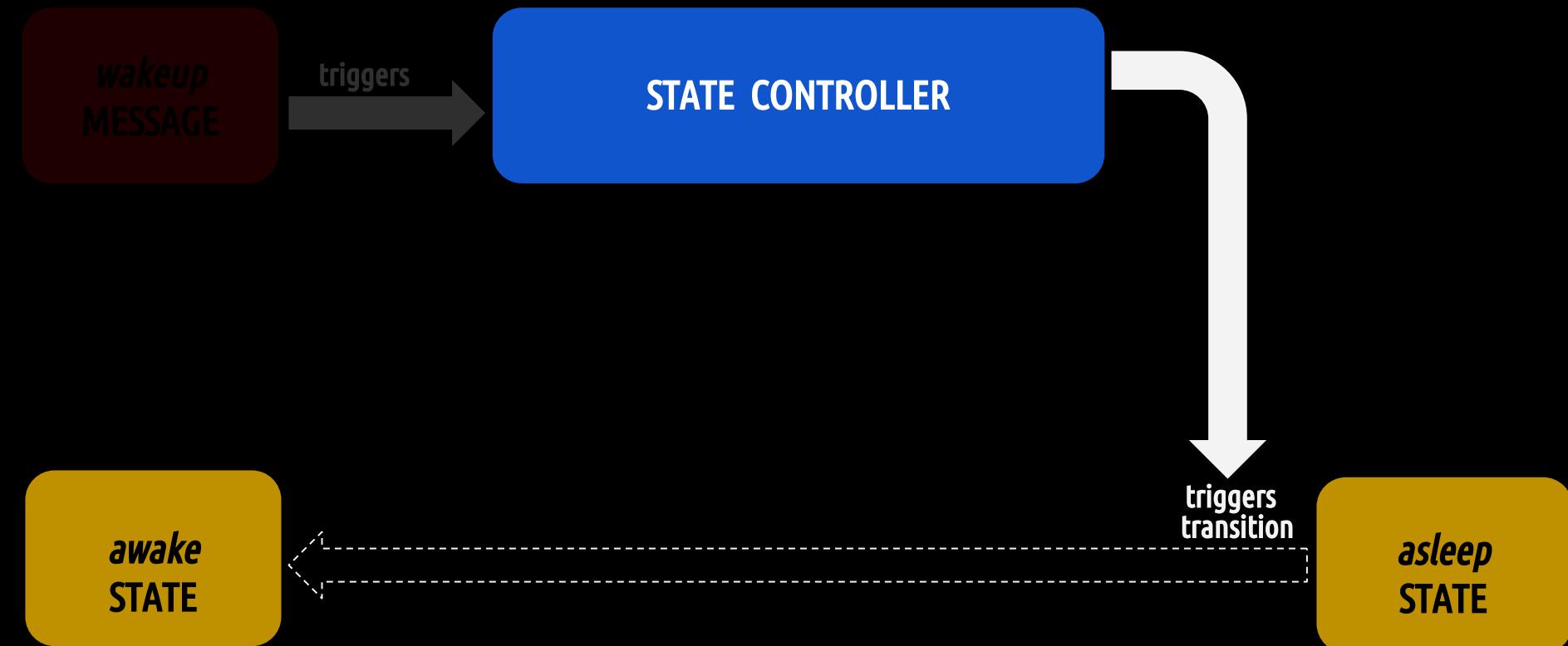
hexState

states flow configurable with DSL

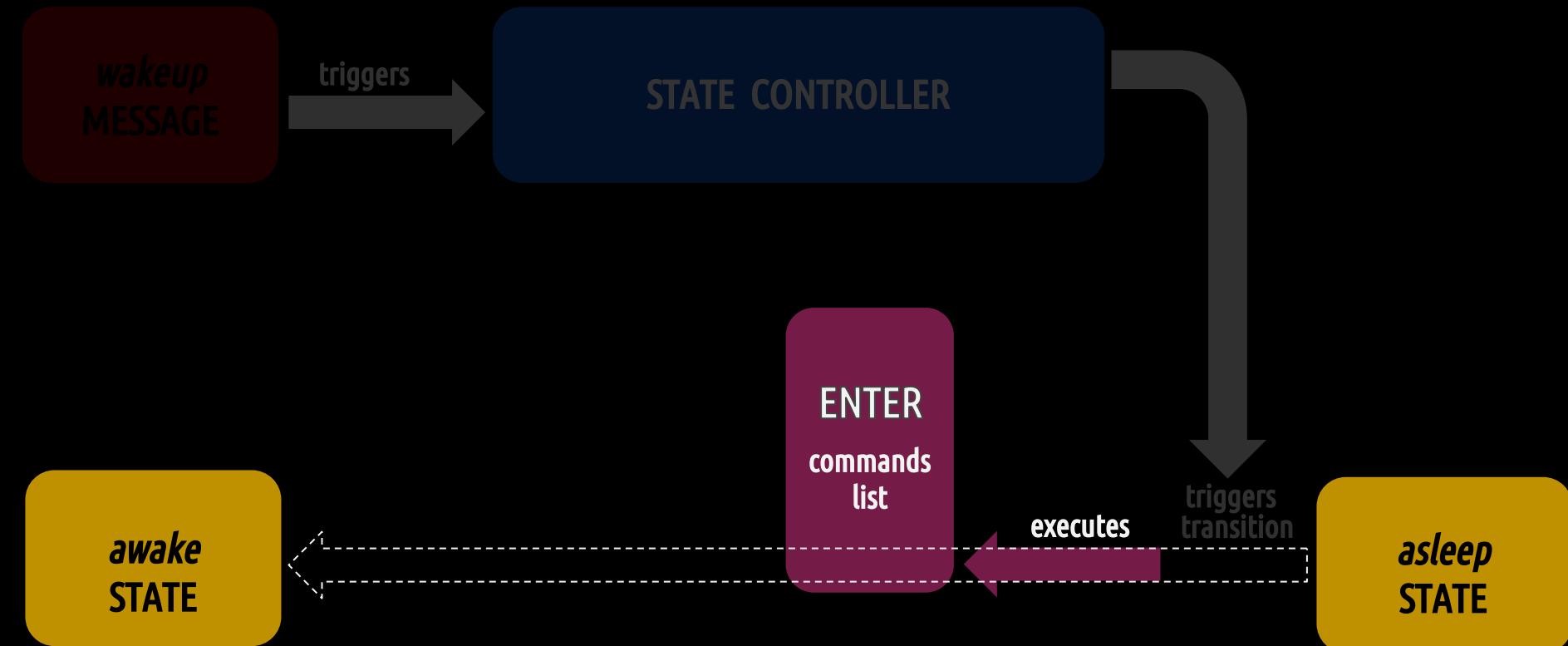
state machine



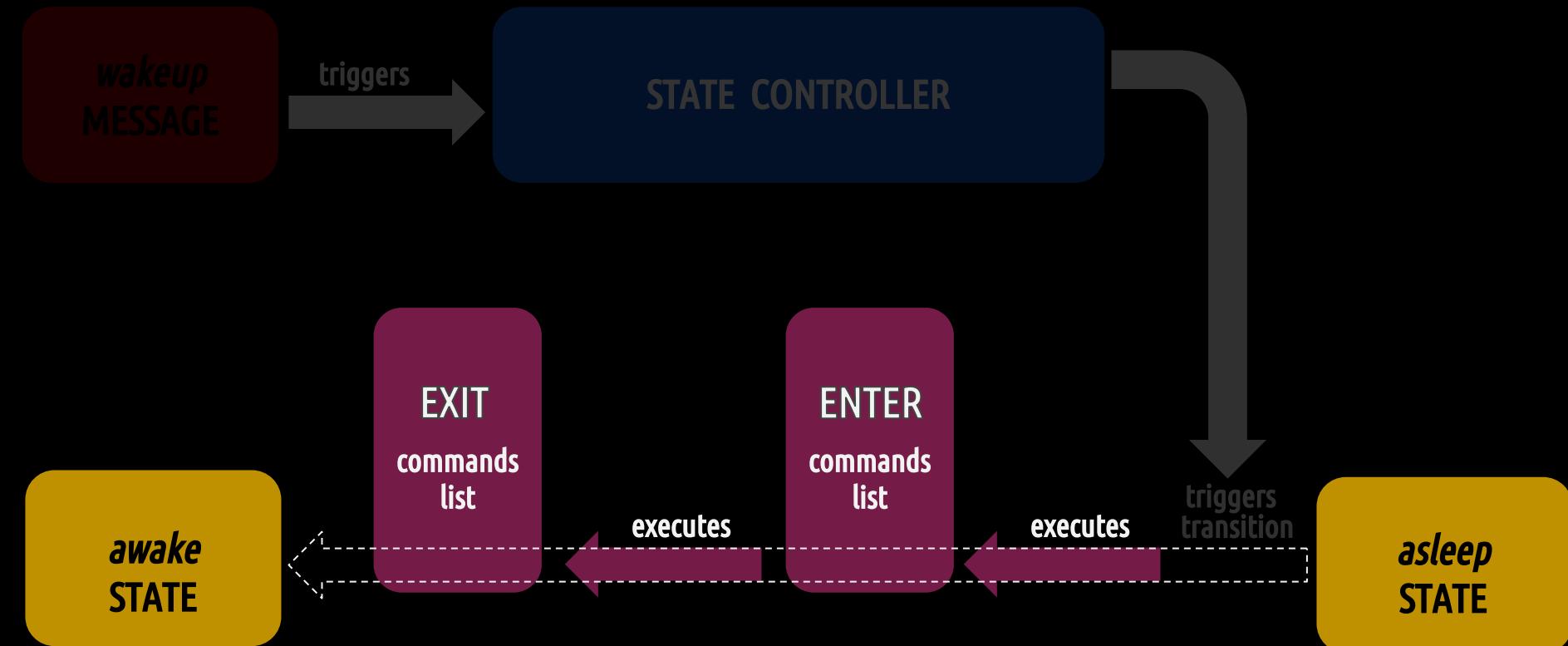
state machine



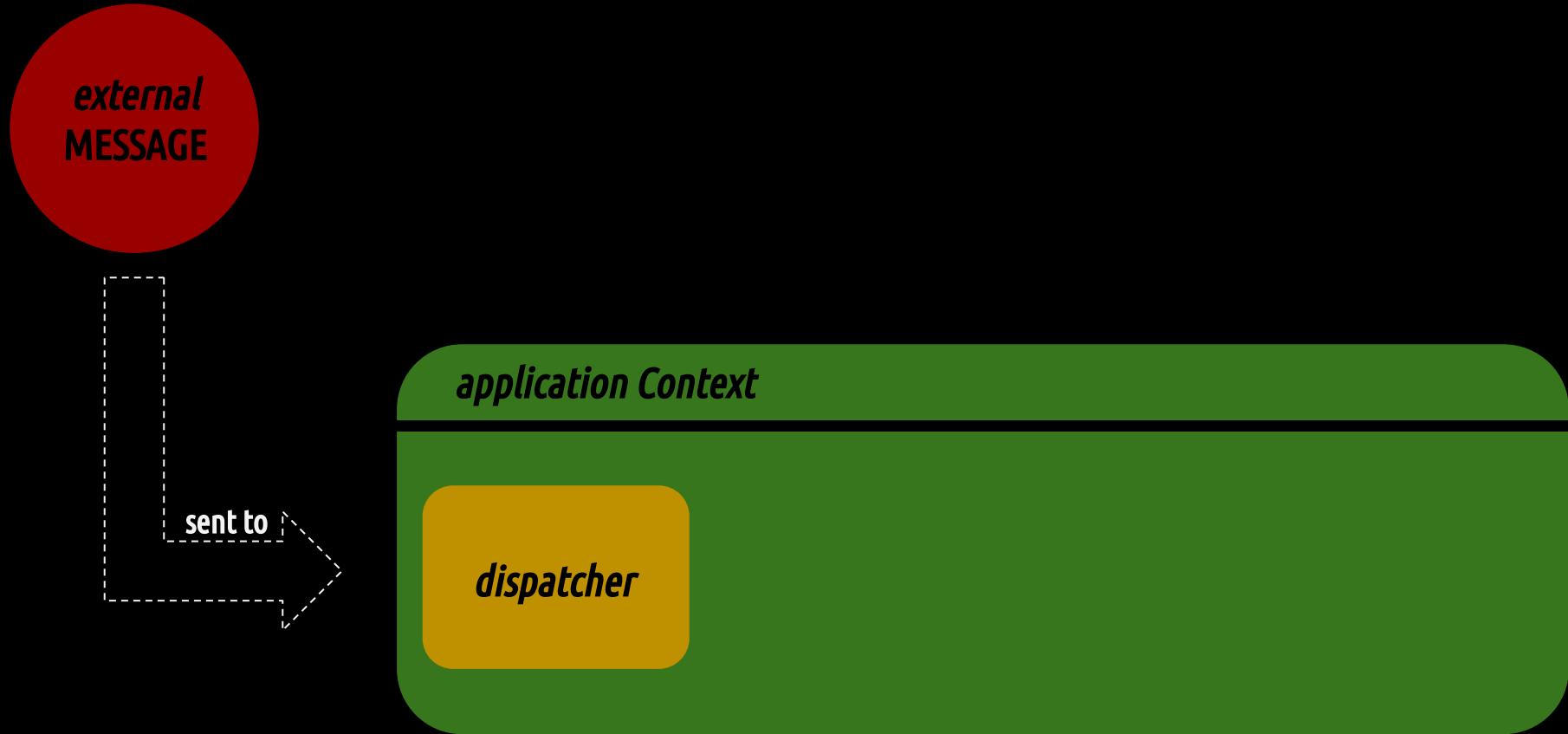
state machine



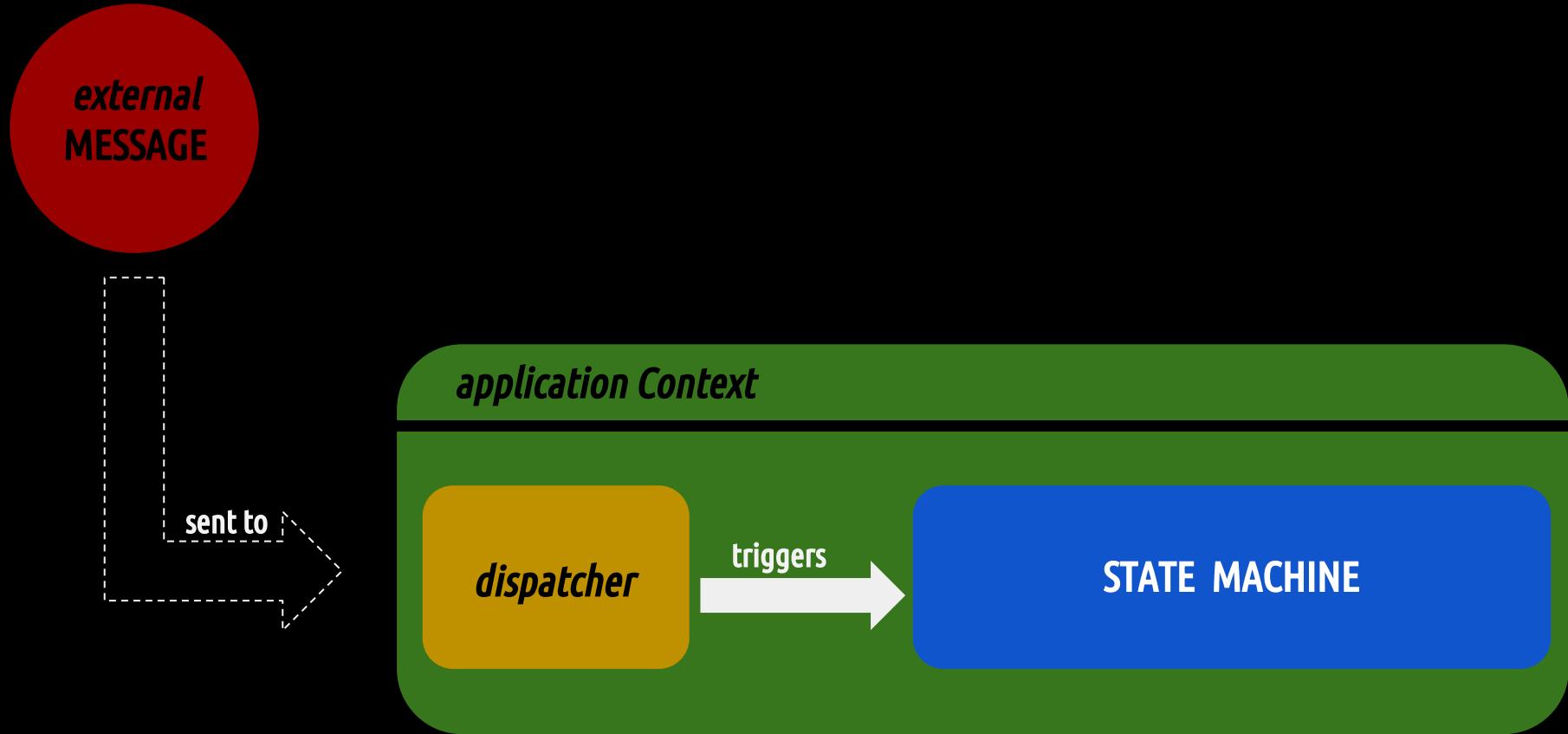
state machine



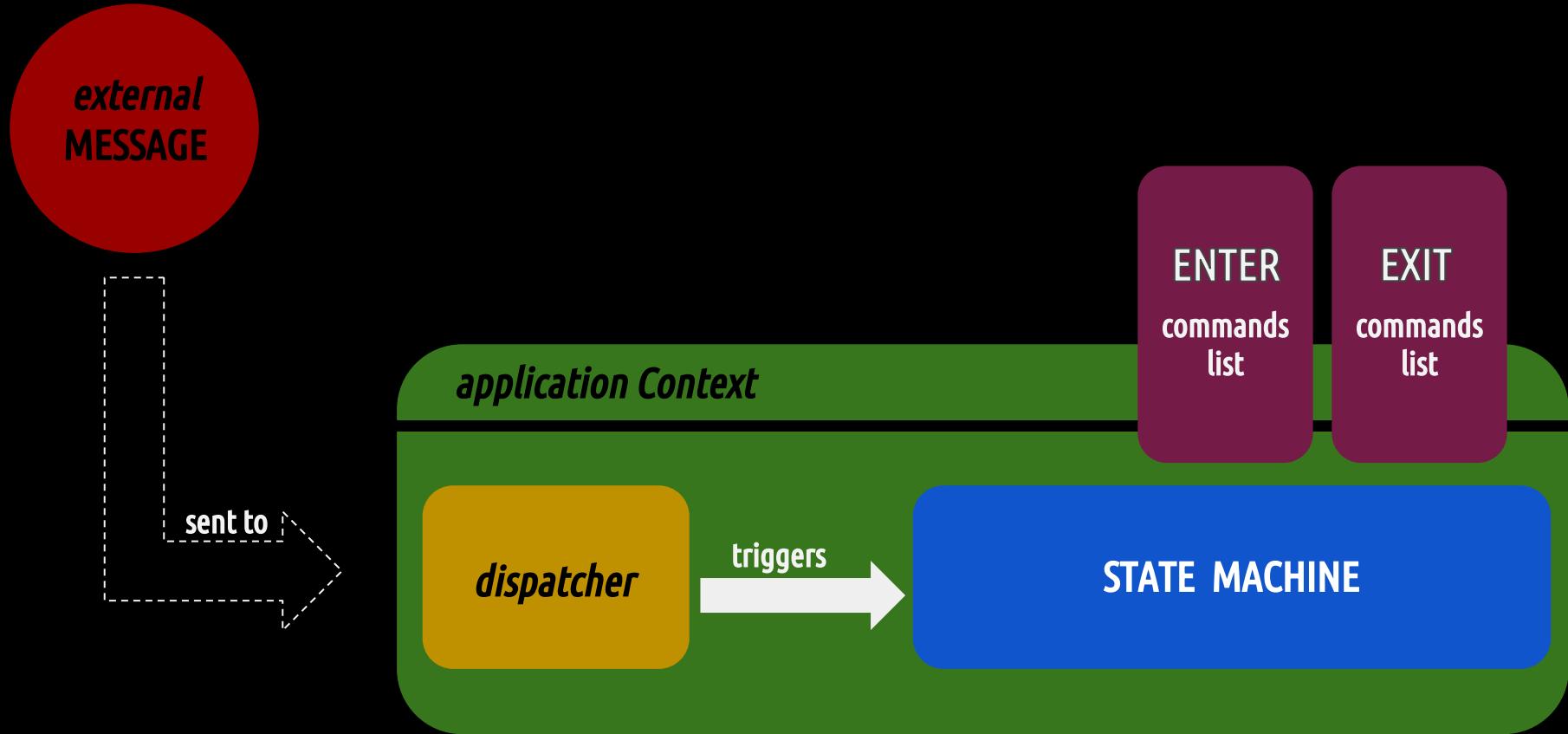
Context state machine



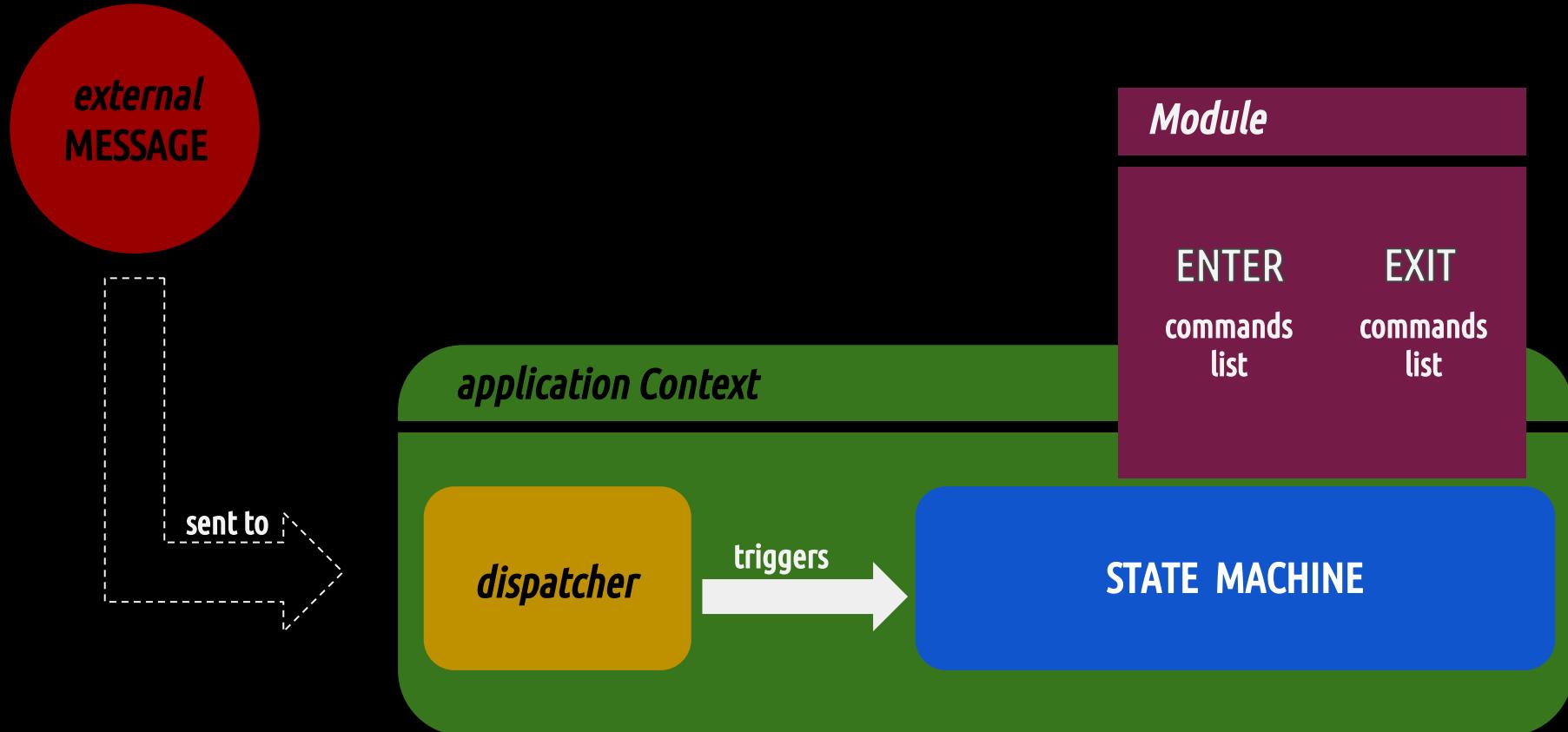
Context state machine



Context state machine



Context state machine



Context states

state flow described with DSL

```
1<root name="applicationContext">
2
3  <state id="assemblingStart" ref="applicationContext.state.ASSEMBLING_START">
4    <enter command-class="hex.ioc.SetUpApplication"/>
5  </state>
6
7</root>
```

Context states

module scope for execution

```
1<root name="applicationContext">
2
3  <state id="objectsBuilt" ref="applicationContext.state.OBJECTS_BUILT">
4    <enter command-class="hex.ioc.InitModule" context-owner="module"/>
5  </state>
6
7  <module id="module" type="hex.ioc.MockModule"/>
8
9</root>
```

hexUnit

UNIT TEST FRAMEWORK



TEST RESULTS NOTIFIERS

```
[[[ Start hex...MVCsuite -> run
Test suite: 'hex.MVCsuite'
Test suite: 'hex.control.Controlsuite'
Test suite: 'hex.control.async.Asyncsuite'
Test class: 'hex.control.async.AsyncCommandEventTest'
  ✓ testType() Test 'type' parameter passed to constructor [1ms]
  ✓ testTarget() Test 'target' parameter passed to constructor [0ms]
  ✓ testClone() Test clone method [0ms]
Test class: 'hex.control.async.AsyncCommandTest'
  ✓ testCallPreExecuteTwice() Test preExecute [2ms]
  ✓ testGetPayload() Test get payload [0ms]
  ✓ testGetOwner() Test owner [0ms]
  ✓ testExecute() Test execute [1ms]
  ✓ testCancel() Test cancel [2ms]
  ✓ testComplete() Test complete [0ms]
  ✓ testFail() Test fail [0ms]
Test class: 'hex.control.async.AsyncCommandUtilTest'
  ✓ testAddListenersToAsyncCommand() Test add listeners to async command
Test suite: 'hex.control.command.Commandsuite'
Test class: 'hex.control.command.CommandExecutionTest'
  ✓ textExcuteCommand() Test command execution
Test class: 'hex.control.command.CommandMappingTest'
  ✓ testGetCommandClass() Test getCommandClass [0ms]
  ✓ testGuards() Test guards [0ms]
  ✓ testIsFireOnce() Test isFireOnce [0ms]
  ✓ testPayloads() Test payloads [0ms]
  ✓ testCompleteHandlers() Test complete handlers [0ms]
  ✓ testFailHandlers() Test fail handlers [0ms]
  ✓ testCancelHandlers() Test cancel handlers [0ms]
Test class: 'hex.control.guard.GuardUtilTest'
  ✓ testGuardClassApproveWithoutInjector() Test guard-class approval without injector [0ms]
  ✓ testGuardClassApproveWithInjector() Test guard-class approval with injector [0ms]
Test class: 'hex.control.FrontControllerTest'
  ✓ testMap() Test map [1ms]
  ✓ testUnmap() Test unmap [0ms]
  ✓ testEventHandling() Functional test of event handling [1ms]
```

Unit Test Sessions				
Session	Test	Status	Time	Details
402	testAddAndRemoveHan	Success	0ms	
	testRemoveAllListeners	Success	0ms	
	testGetConfiguration	Success	0ms	
	testVirtualMethods	Success	1ms	
	testDefaultServiceTimeout	Success	0ms	
	testServiceTimeout	Success	0ms	
	testDefaultServiceTimeout	Success	0ms	
	testServiceTimeout	Success	0ms	
	testServiceURL	Success	0ms	
HexMVC	78 tests	Success	235ms	
Config	2 tests	Success	0ms	
Stateful	2 tests	Success	0ms	
testConfigureThrowsInje	Success	0ms		
testMapBehavior	Success	0ms		
Control	55 tests	Success	220ms	
Async	8 tests	Success	4ms	
testCallPreExecuteTwice	Success	0ms		
testGetResult	Success	0ms		
testGetOwner	Success	0ms		
testExecute	Success	1ms		
testCancel	Success	1ms		
testComplete	Success	1ms		
testFail	Success	1ms		
testAddListenersToAsyn	Success	0ms		
Command	11 tests	Success	3ms	
testExecuteCommand	Success	2ms		
testExecuteCommandW	Success	0ms		
testGetCommandClass	Success	0ms		
testGuards	Success	0ms		
testIsFireOnce	Success	0ms		
testPayloads	Success	0ms		
testCompleteHandlers	Success	0ms		
testFailHandlers	Success	1ms		
testCancelHandlers	Success	0ms		
testMappingResults	Success	0ms		
testSetLastCommandIns	Success	0ms		
testGuardClassApproveWith	Success	1ms		
testGuardClassApproveWith	Success	0ms		
testMap	Success	0ms		
testUnmap	Success	0ms		
testRequestHandling	Success	0ms		

hexCore

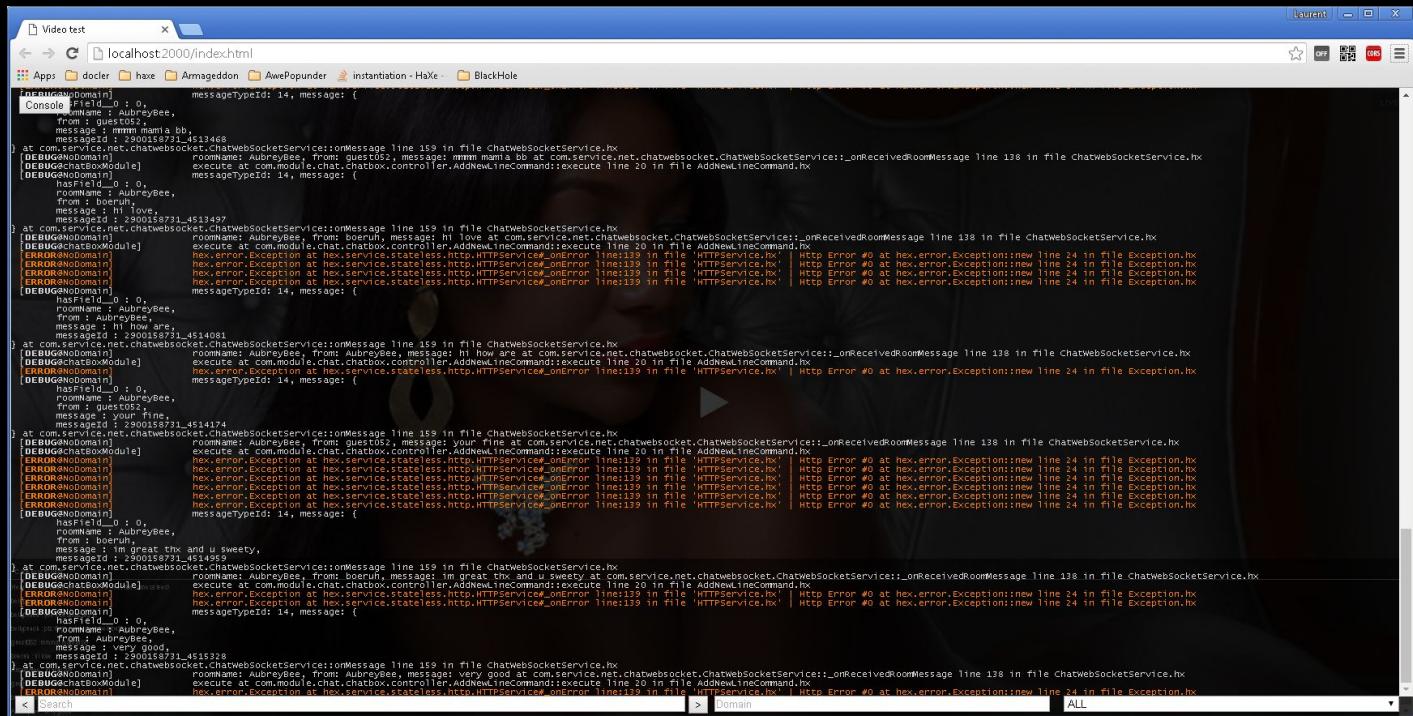
Logging API

filter logs by:

- Domain Levels

web console

- ## ● search text

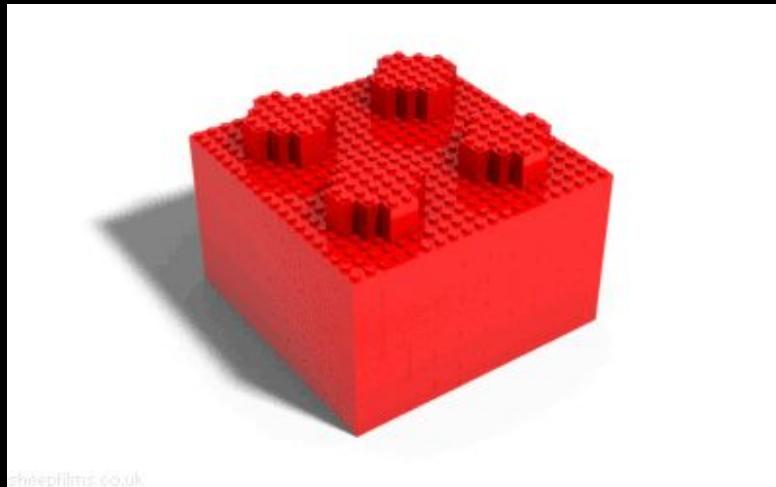


hexMachina

What's next?

Recursive composition

context aggregates contexts, that aggregates contexts, that...



Contexts hierarchy

hierarchy is handled by DSL

```
1<root name="applicationContext">
2
3    <import name="videoContext" file="folder/video.xml"/>
4    <import name="chatContext" file="folder/chat.xml"/>
5
6    <module id="application" type="com.docler.Application">
7        <argument ref="videoContext.videoModule"/>
8        <argument ref="chatContext.chatModule"/>
9    </module>
10 </root>
```

Other planned additions

- High level of code coverage
- More platforms testing
- More articles
- Add states transitions to DSL compiler
- hexDom (virtual DOM, databinding)
- Full context error-checking at compile time

Questions...

- @francisbourre
- francis_bourre@me.com
- linkedin.com/francisbourre
- hexmachina.org

WE HIRE HAXE DEVS



CONTACT US