hexMachina

urban framework for hardcore developers

WHO AM I ?

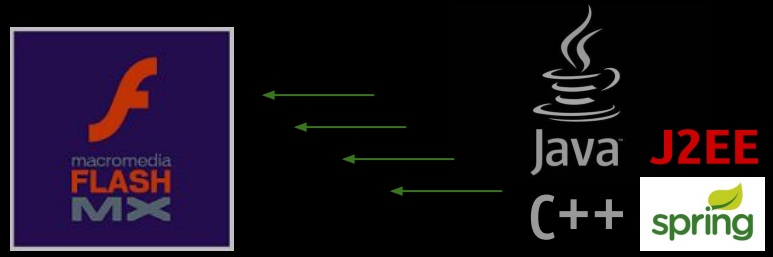
francis bourre / Docler Holding
author of **@pixlib**

pixLib adventures

in the web apps jungles

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

HAXE

Works

with extra benefits

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

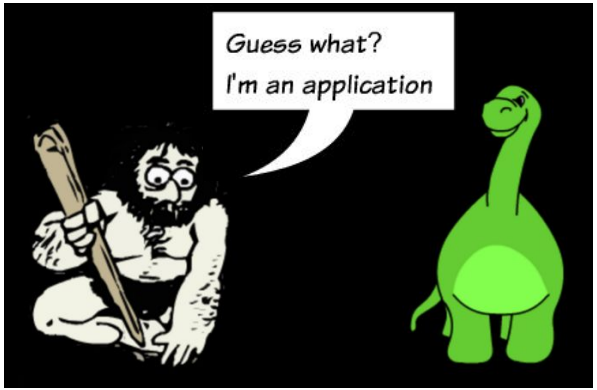
- Formal specs of your components
- Hide the implementation
- Easy team workflow
- And more...

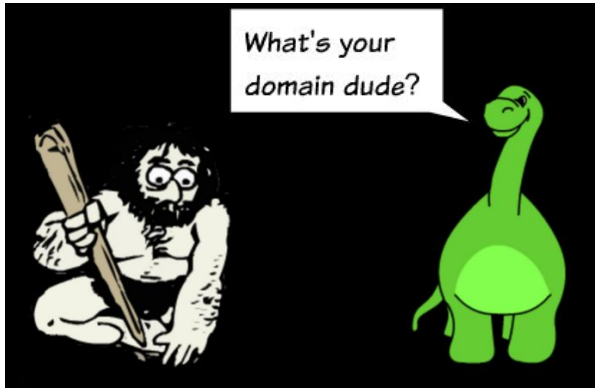
Code reusability

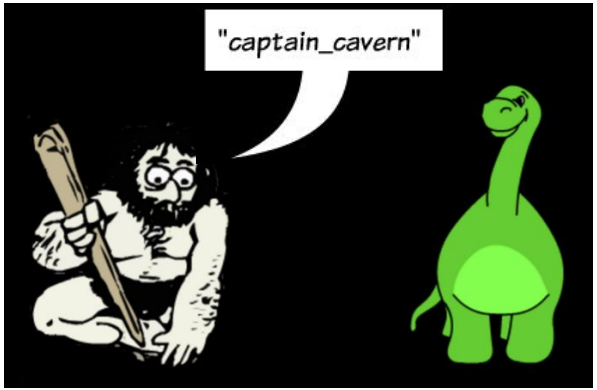
modular system and configurable components

MODULE

Every module is a micro-application



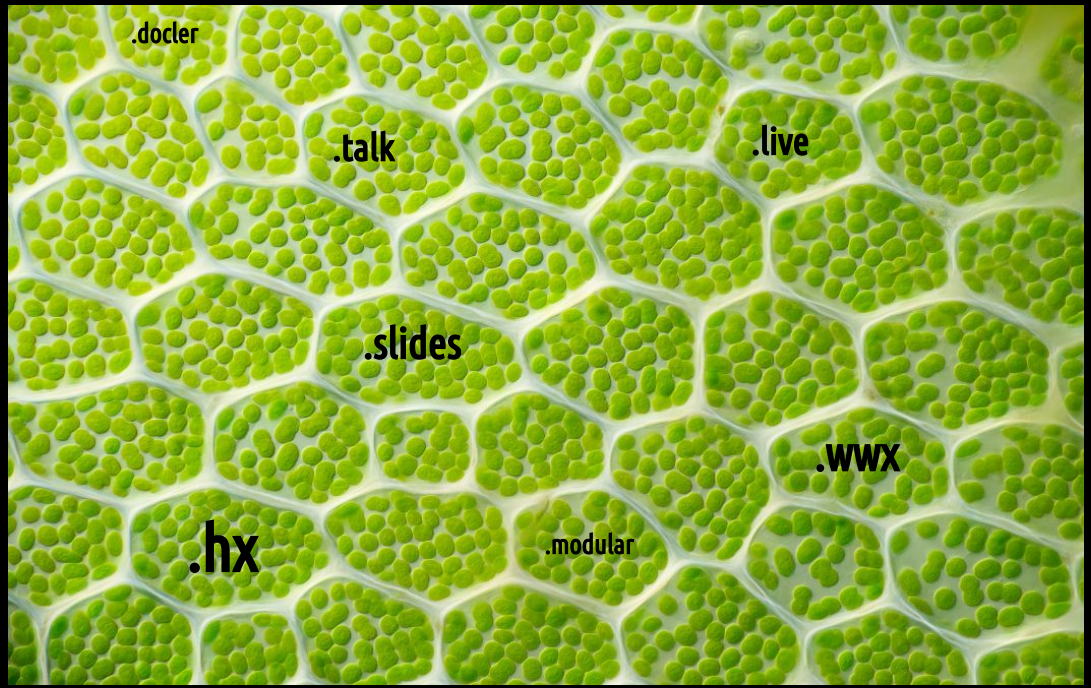




...And proud to be.

DOMAIN

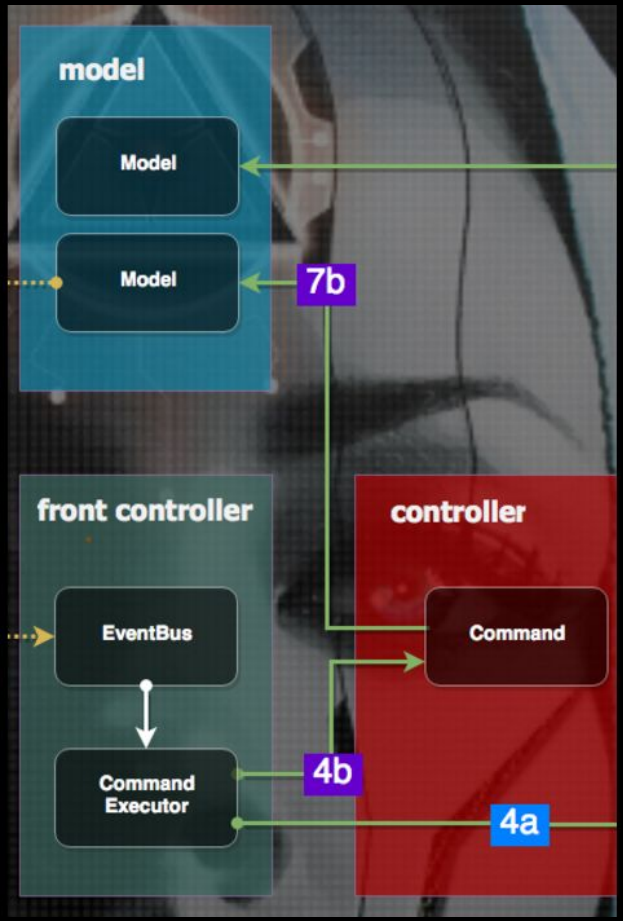
Every module belongs to an unique domain



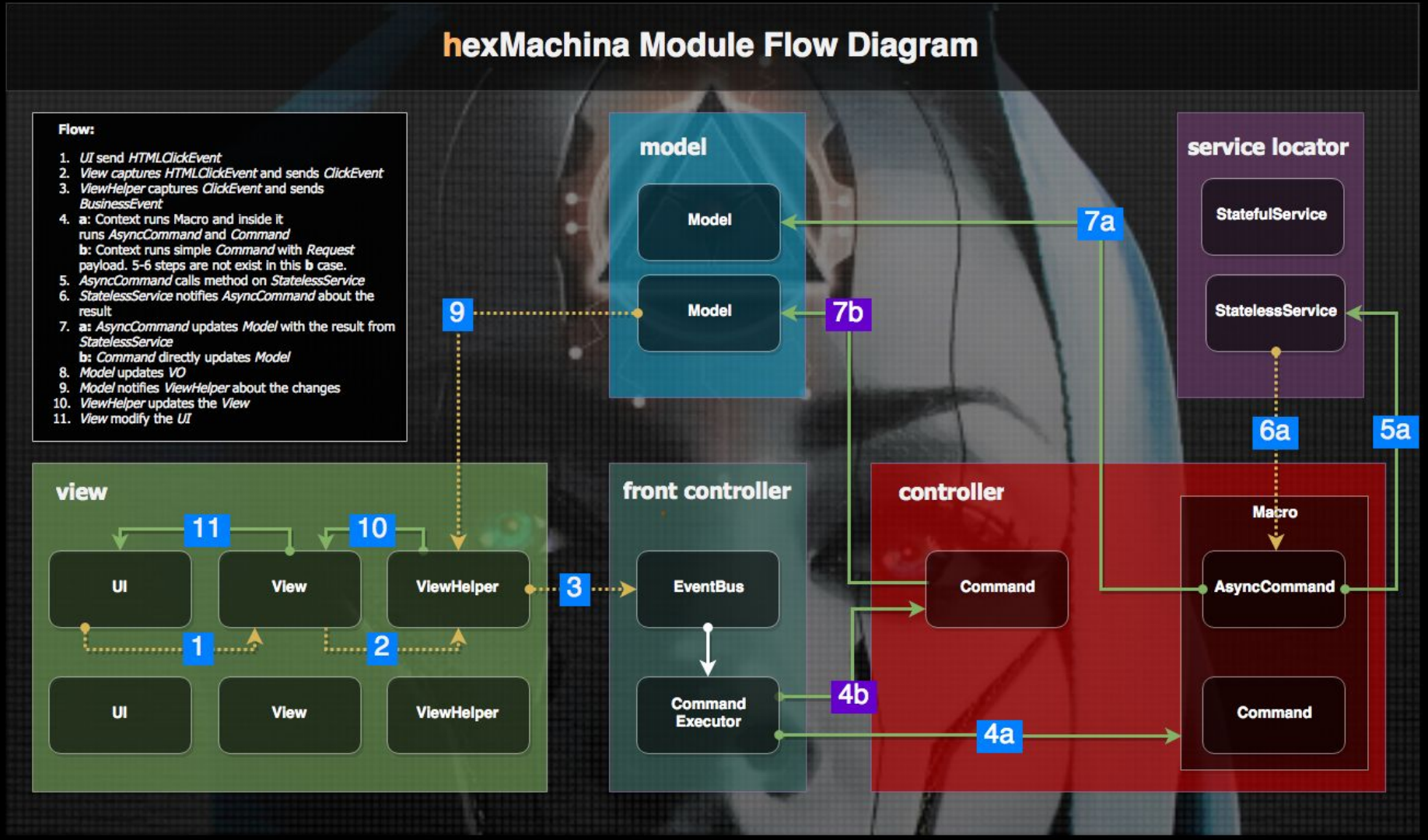
MODULE

micro-application

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



hexMachina Module Flow Diagram



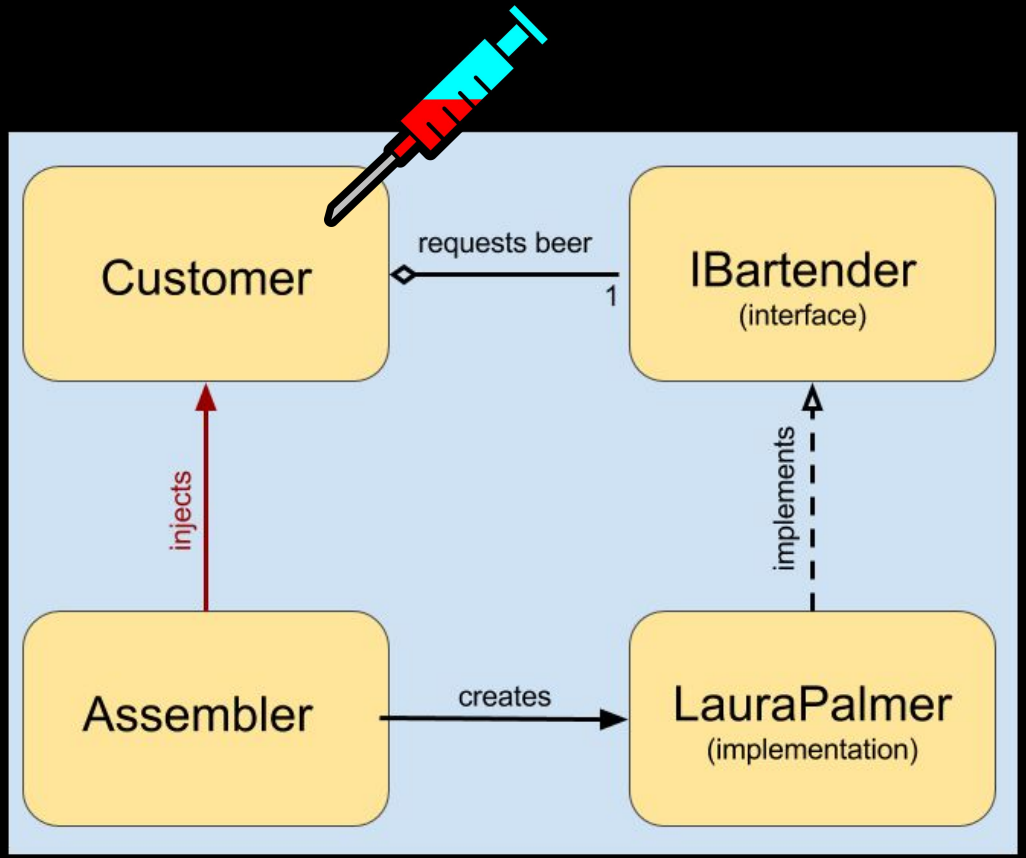
Low coupling

prefer runtime coupling - USE IoC

Hollywood principle

" Don’t call us, we will call you "

Dependency Injection



Automate coupling

Use automatic dependency injection

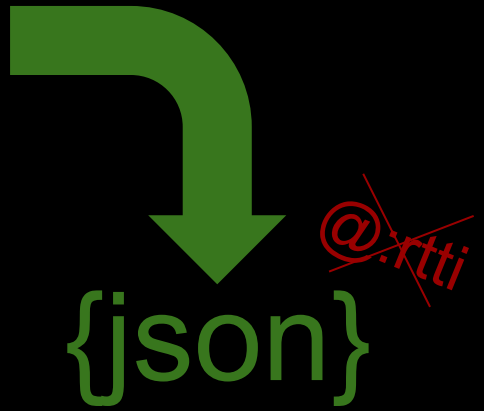
INJECTION with ANNOTATIONS

```
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

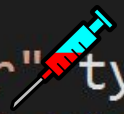
Build configurable architecture

use Domain Specific Language

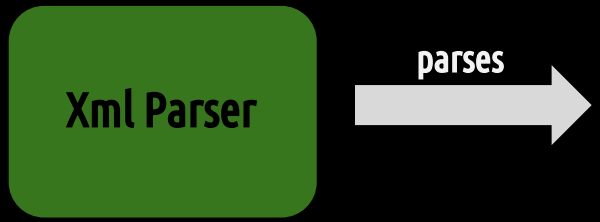
hexIoC

Injections with DSL

```
<root name="applicationContext">

    <service id="movieFinder" type="services.MovieFinder"/>

    <module id="movieLister" type="modules.MovieLister">
        <argument ref="movieFinder">
    </module>

</root>
```

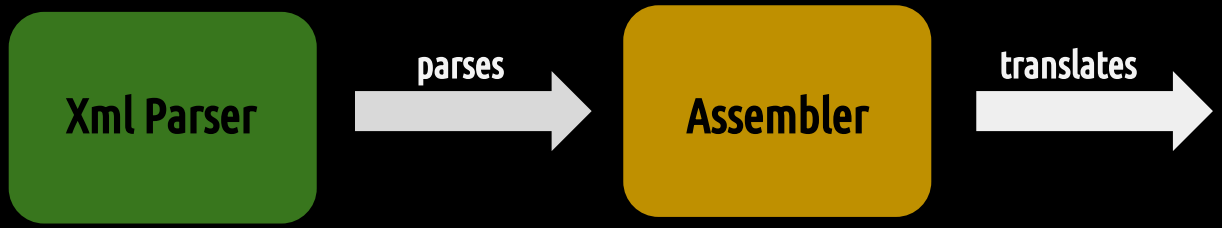


DSL PARSING/ASSEMBLING FLOW



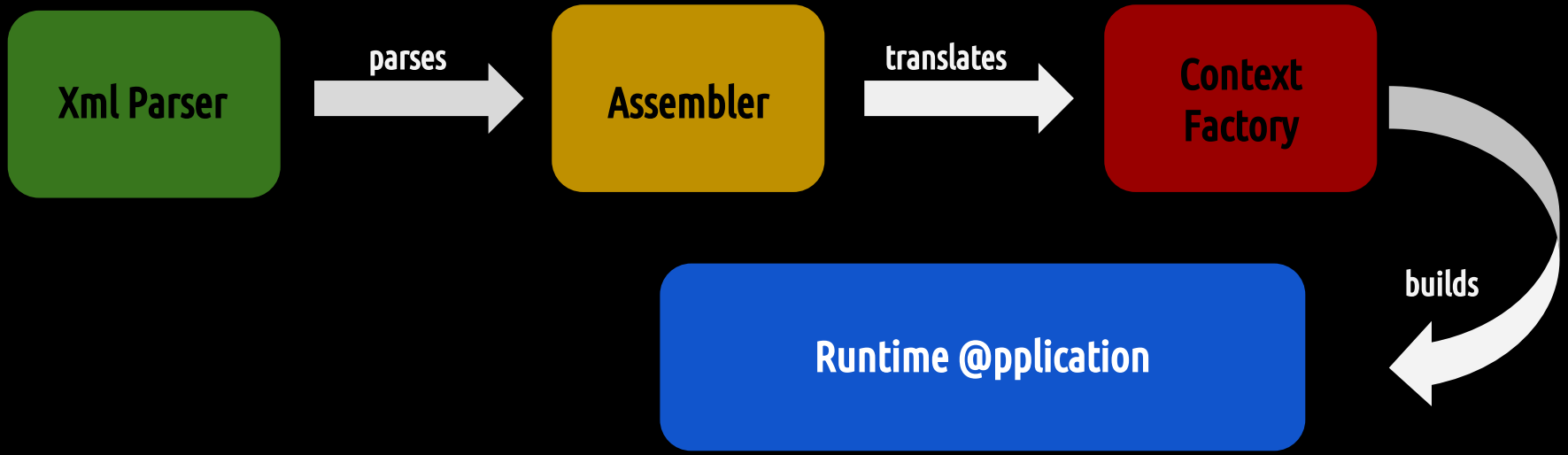

```
public function new()
{
    XmlReader.readXmlFile( "example/configuration/context.xml" );
}
```

DSL PARSING/ASSEMBLING FLOW



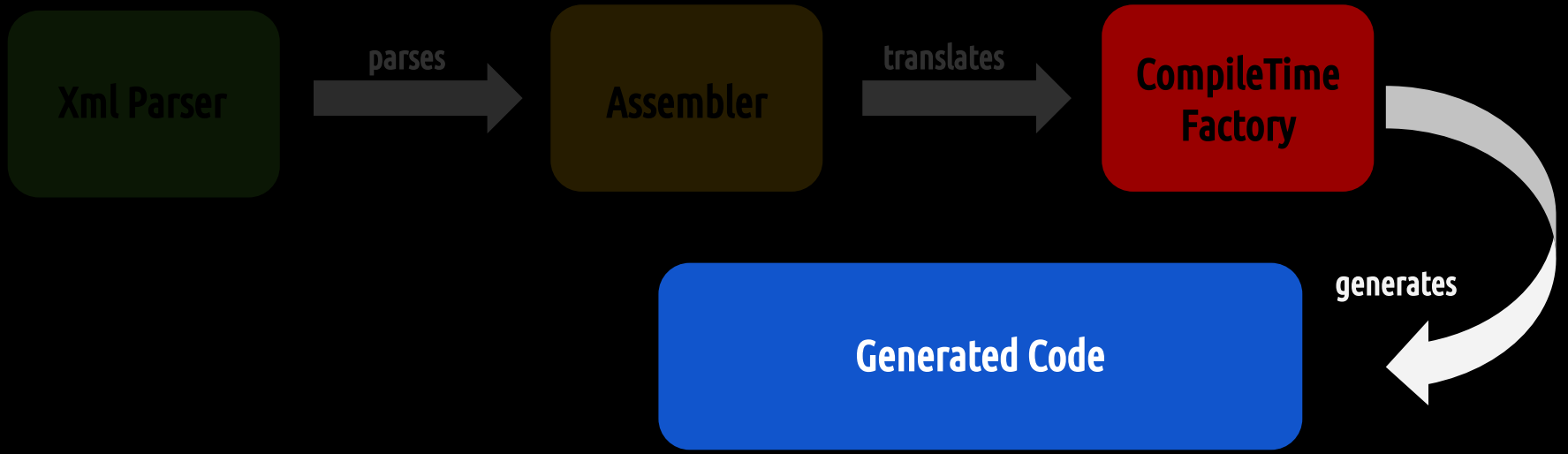

```
public function new()
{
    XmlReader.readXmlFile( "example/configuration/context.xml" );
}
```

DSL PARSING/ASSEMBLING FLOW




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

DSL PARSING/COMPILING 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

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

HEXMACHINA DSL

simple module creation

```
<root>
    <!-- MODULES -->
    <module id="gallery" type="example.module.gallery.GalleryModule">
        <argument ref="flickerServiceLocator"/>
    </module>
</root>
```

SERVICES

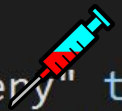
service locator / services mapping

```
<root>
    <serviceLocator id="flickerServiceLocator" type="hex.config.stateful.ServiceLocator" >
        <item>
            <key type="Class" value="example.module.gallery.service.IGetPhotosService"/>
            <value type="Class" value="example.service.flickr.FlickrGetPhotos"/>
        </item>
    </serviceLocator>
</root>
```

HEXMACHINA DSL

constructor injection by DSL

```
<root>
    <!-- MODULES -->
    <module id="gallery" type="example.module.gallery.GalleryModule">
        <argument ref="flickerServiceLocator"/>
    </module>
</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_servic
e_flickr_FlickrGetPhotos,null);
	coreFactory.register("flickerServiceLocator",flickerServiceLocator);
	var gallery = new example_module_gallery_GalleryModule(flickerServiceLocator);
	coreFactory.register("gallery",gallery);
	applicationAssembler;
};
```

INCLUDE

recursive files inclusion

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

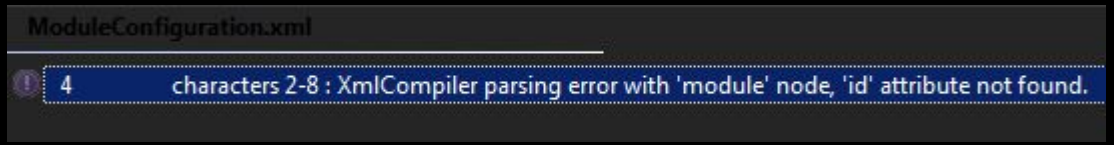
ModuleConfiguration.xml

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

DSL CHECKING @compile time

missing ID

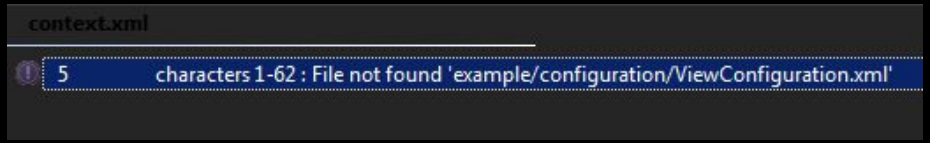
```xml
<?xml version="1.0" encoding="utf-8" ?>
<root>
    <!-- MODULES -->
    <module type="example.module.gallery.GalleryModule">
        <argument ref="flickerServiceLocator"/>
    </module>
</root>
```



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

Observable named **domain**

type

type

observer

observer

observer

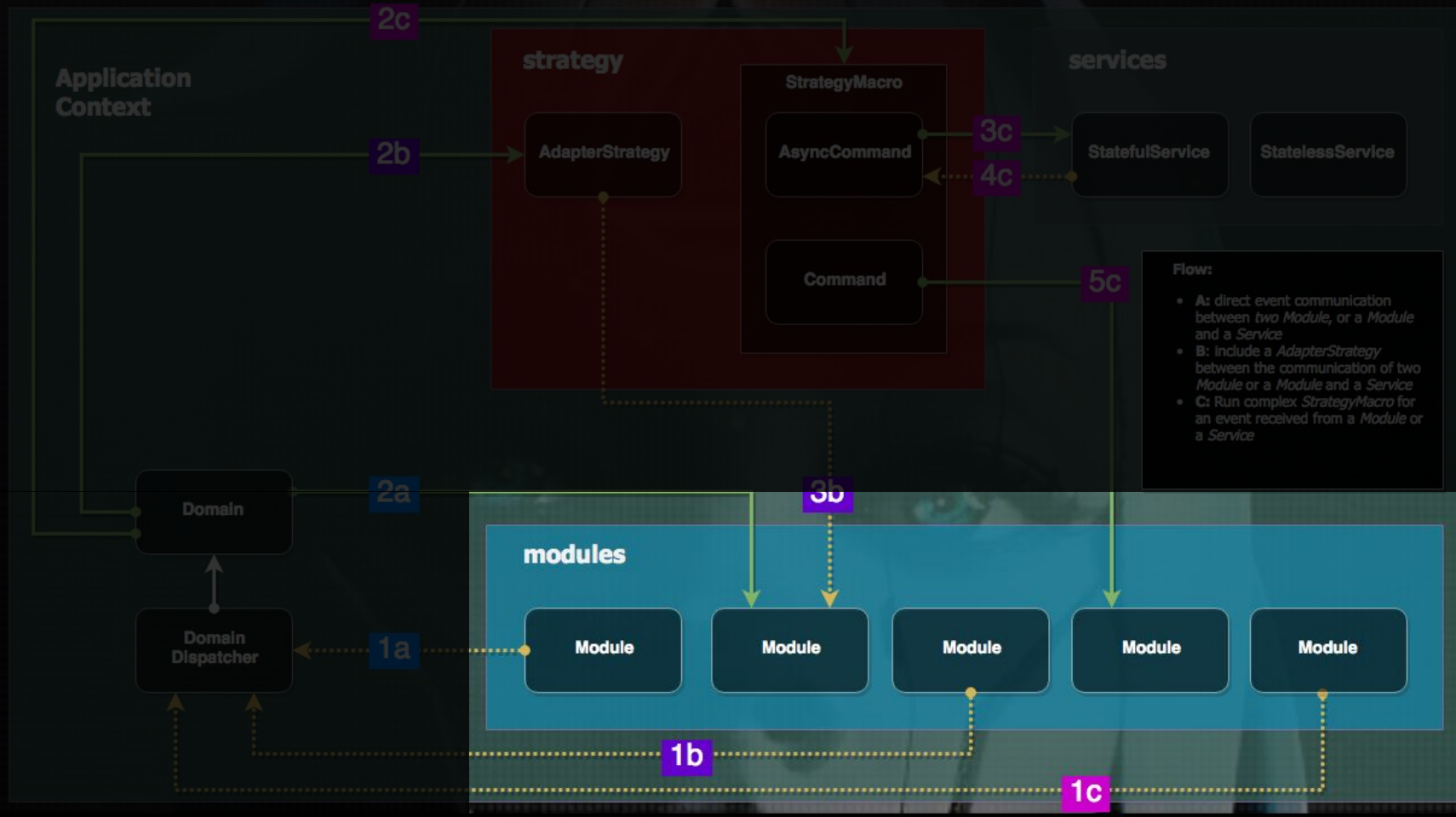
observer

observer

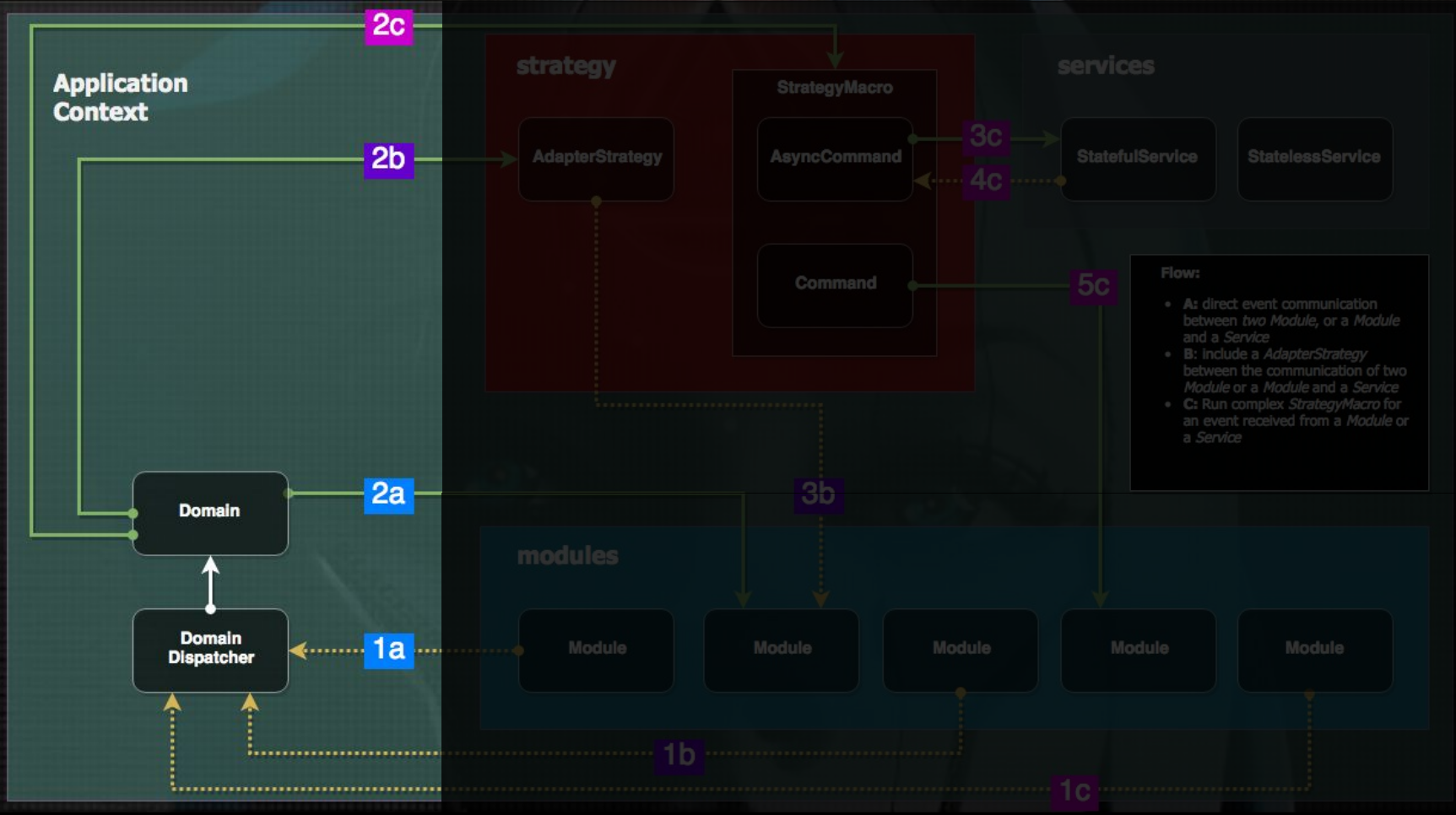
observer

Observer pattern
with domain messaging

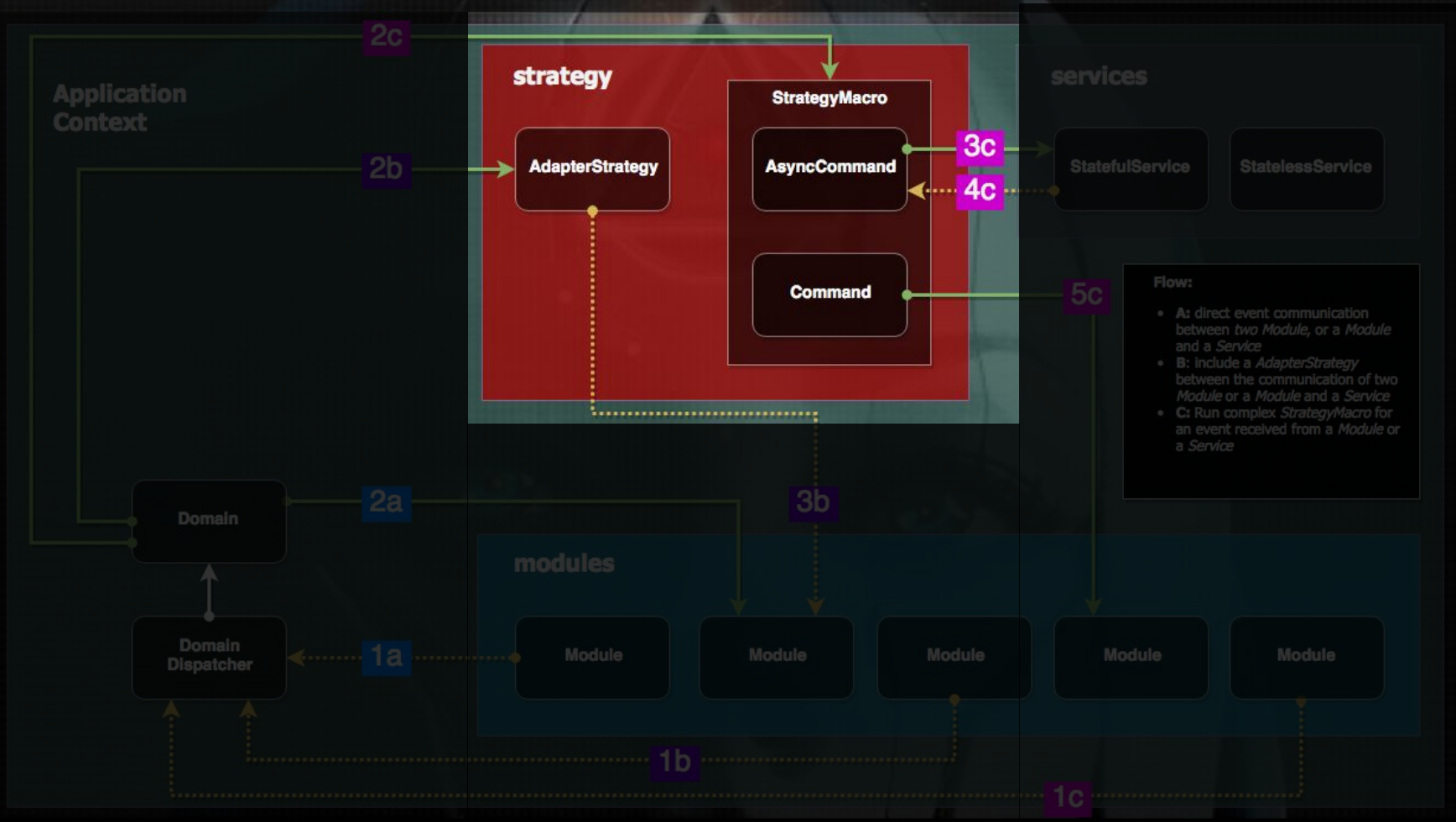
hexMachina ApplicationContext Flow Diagram



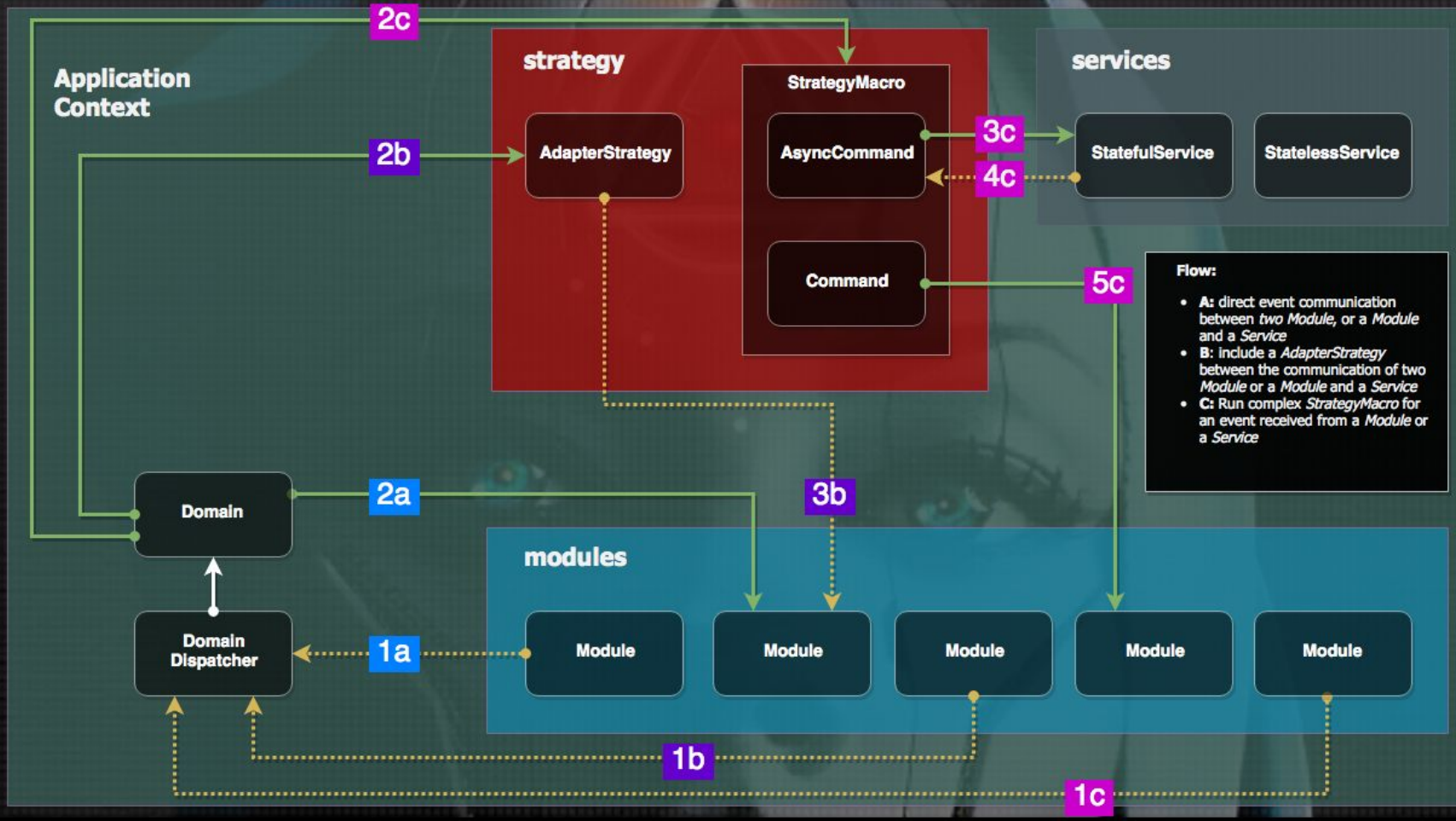
hexMachina ApplicationContext Flow Diagram



hexMachina ApplicationContext Flow Diagram



hexMachina ApplicationContext Flow Diagram



hexState

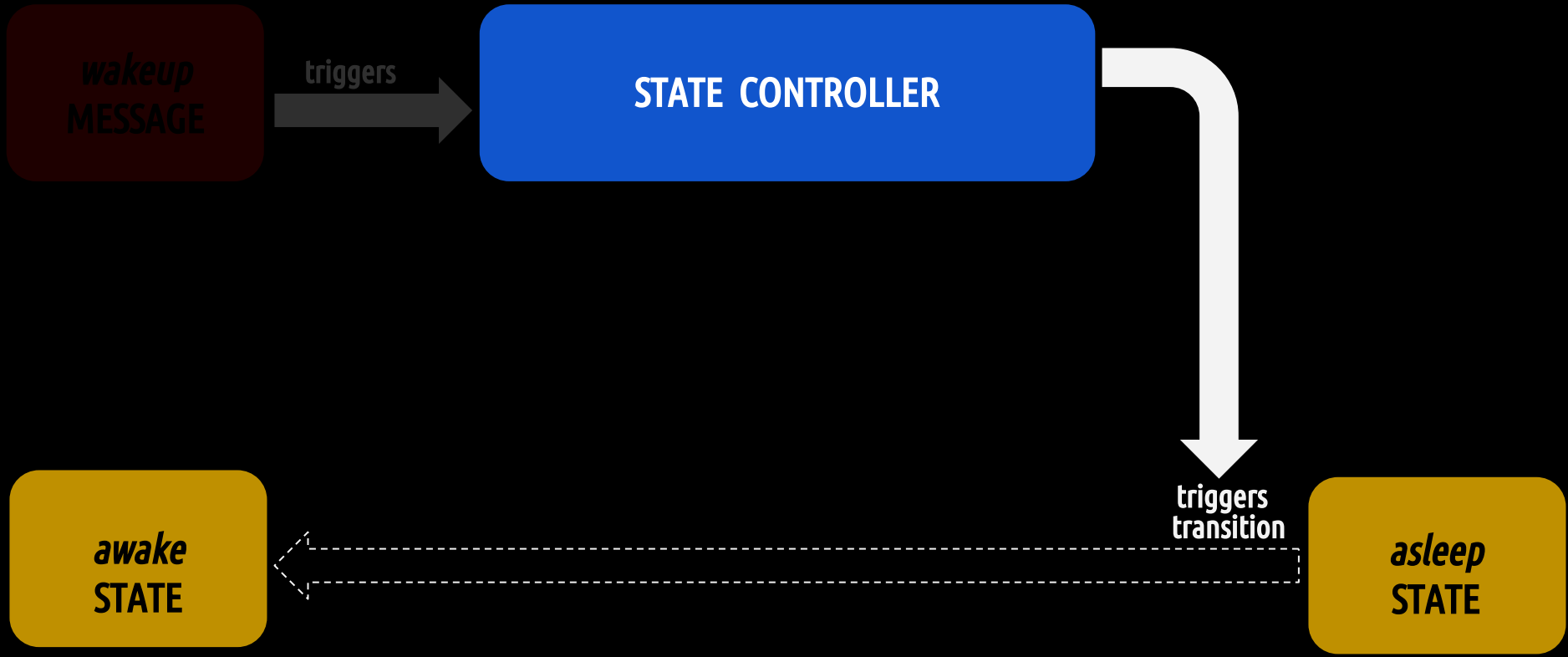
states flow configurable with DSL

state machine

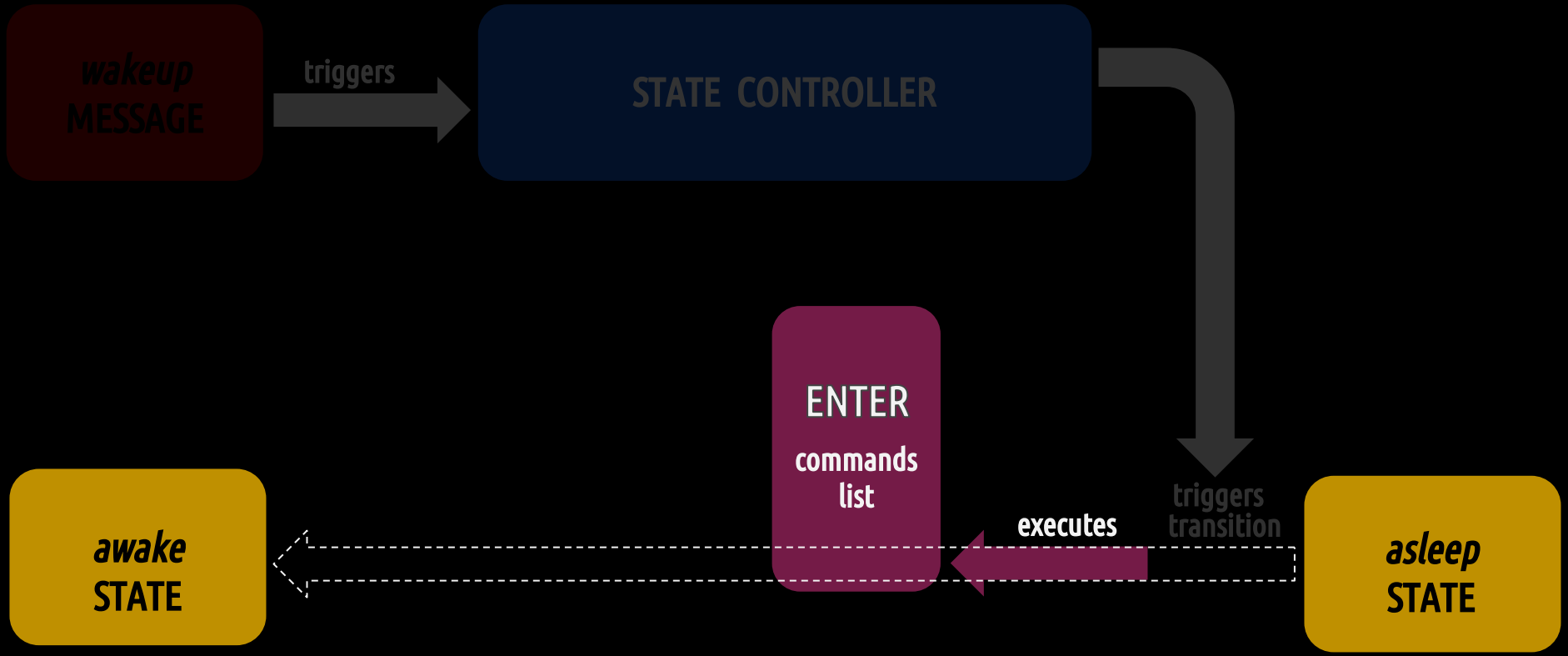
awake
STATE

asleep
STATE

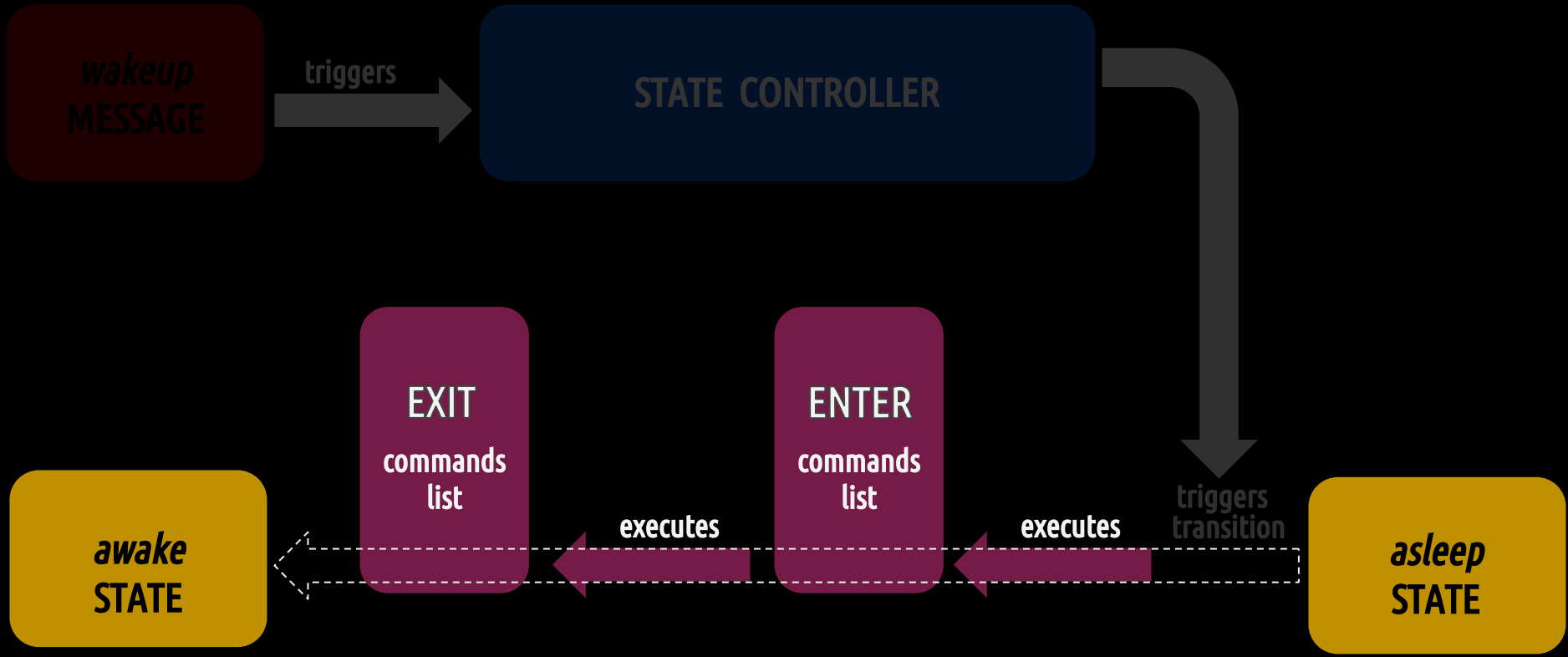
state machine



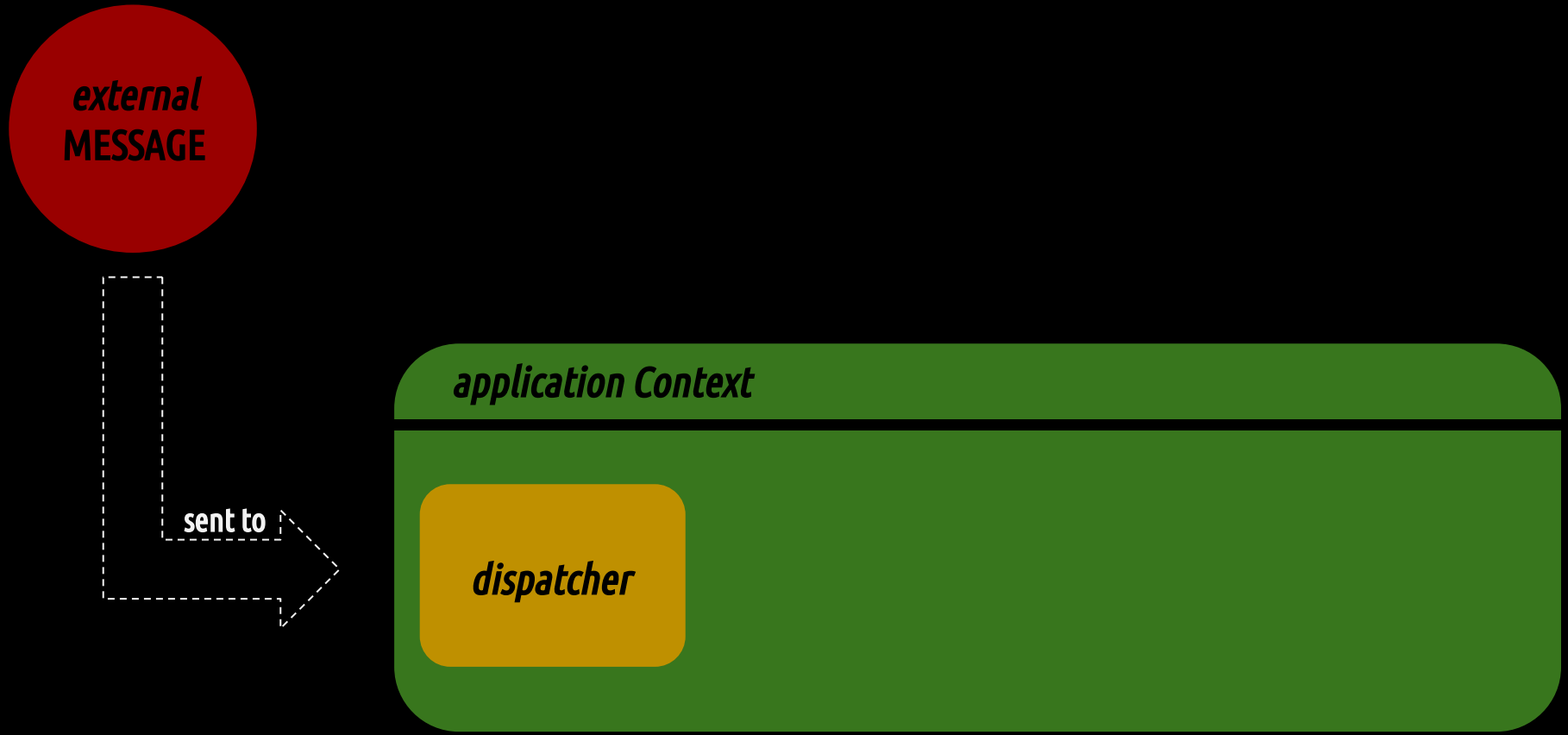
state machine



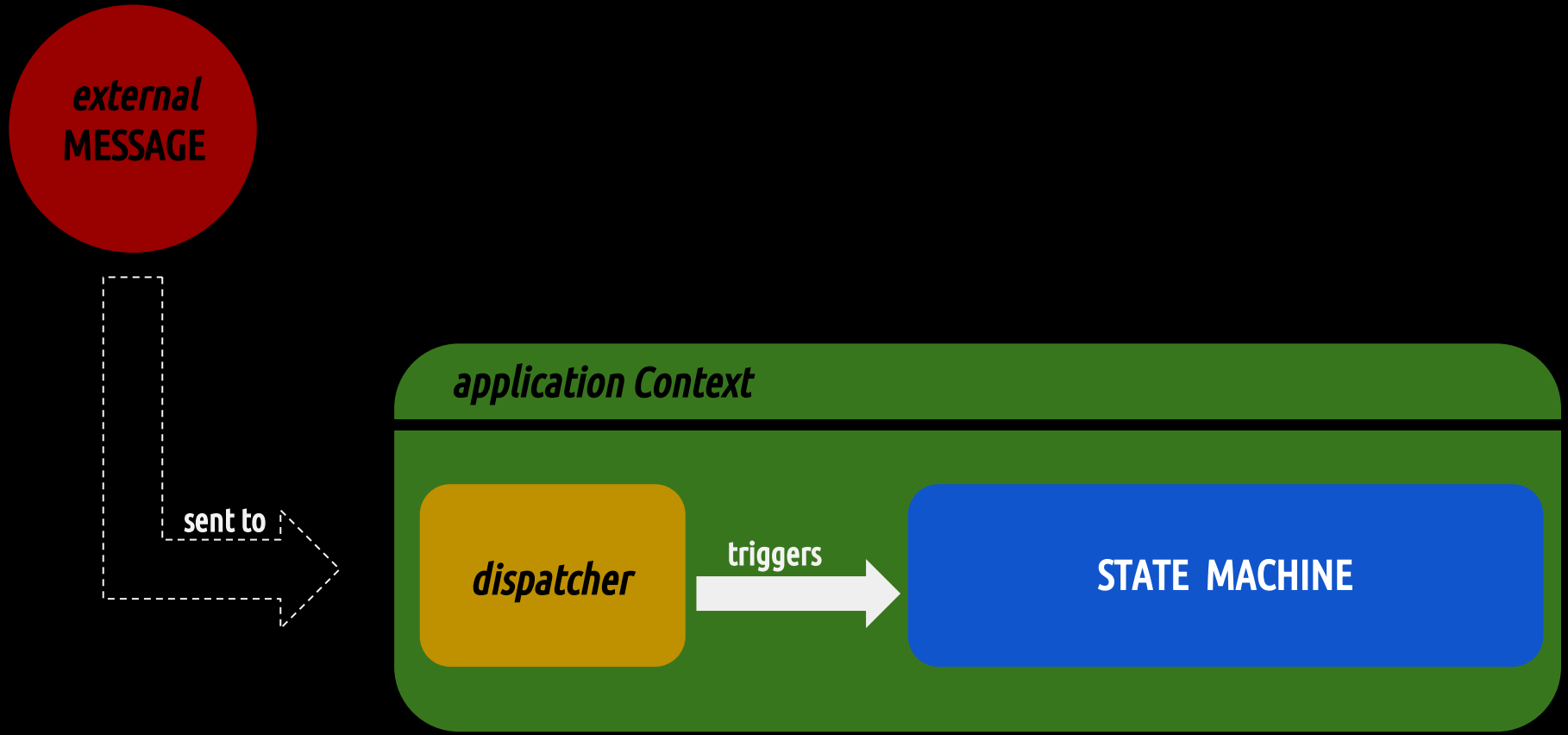
state machine



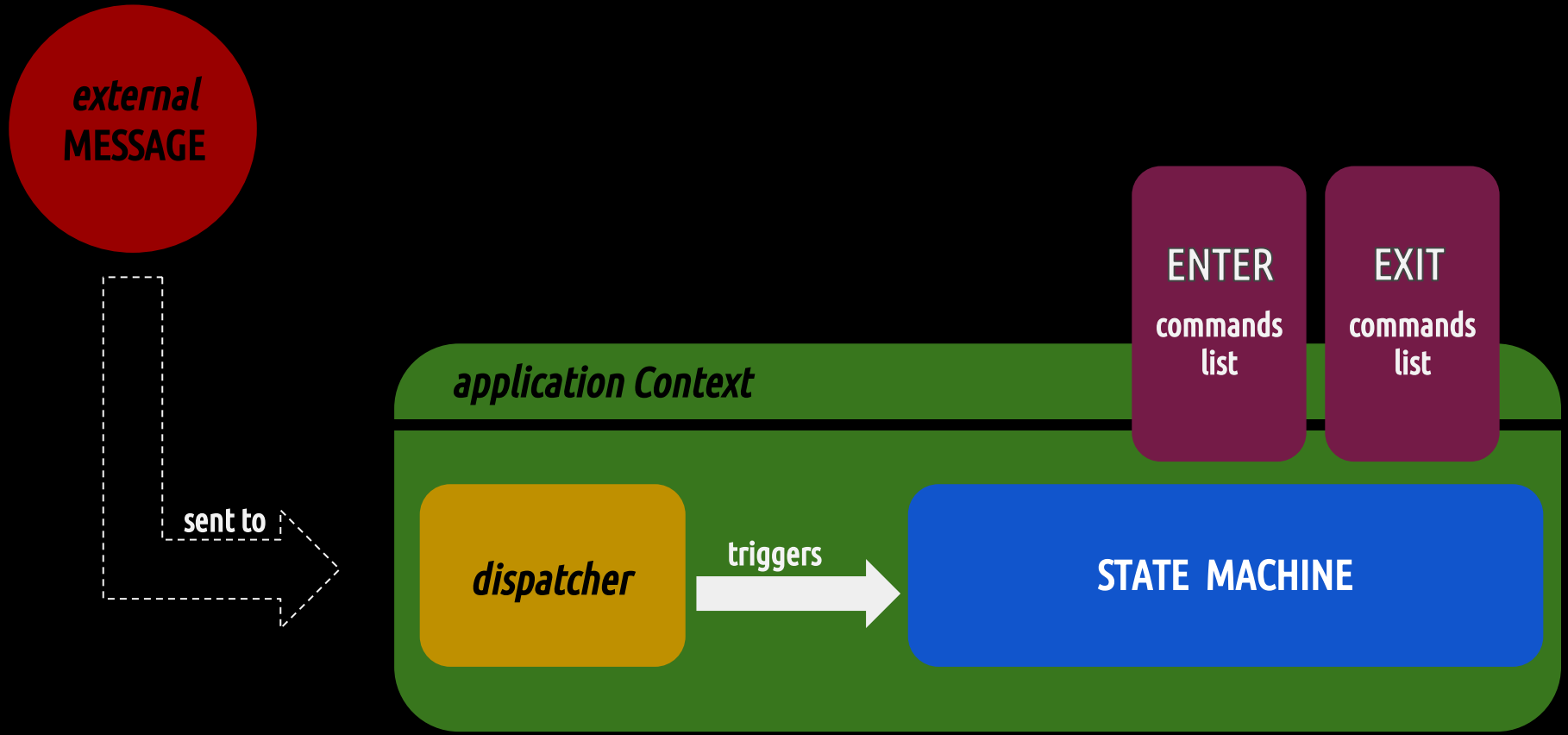
Context state machine



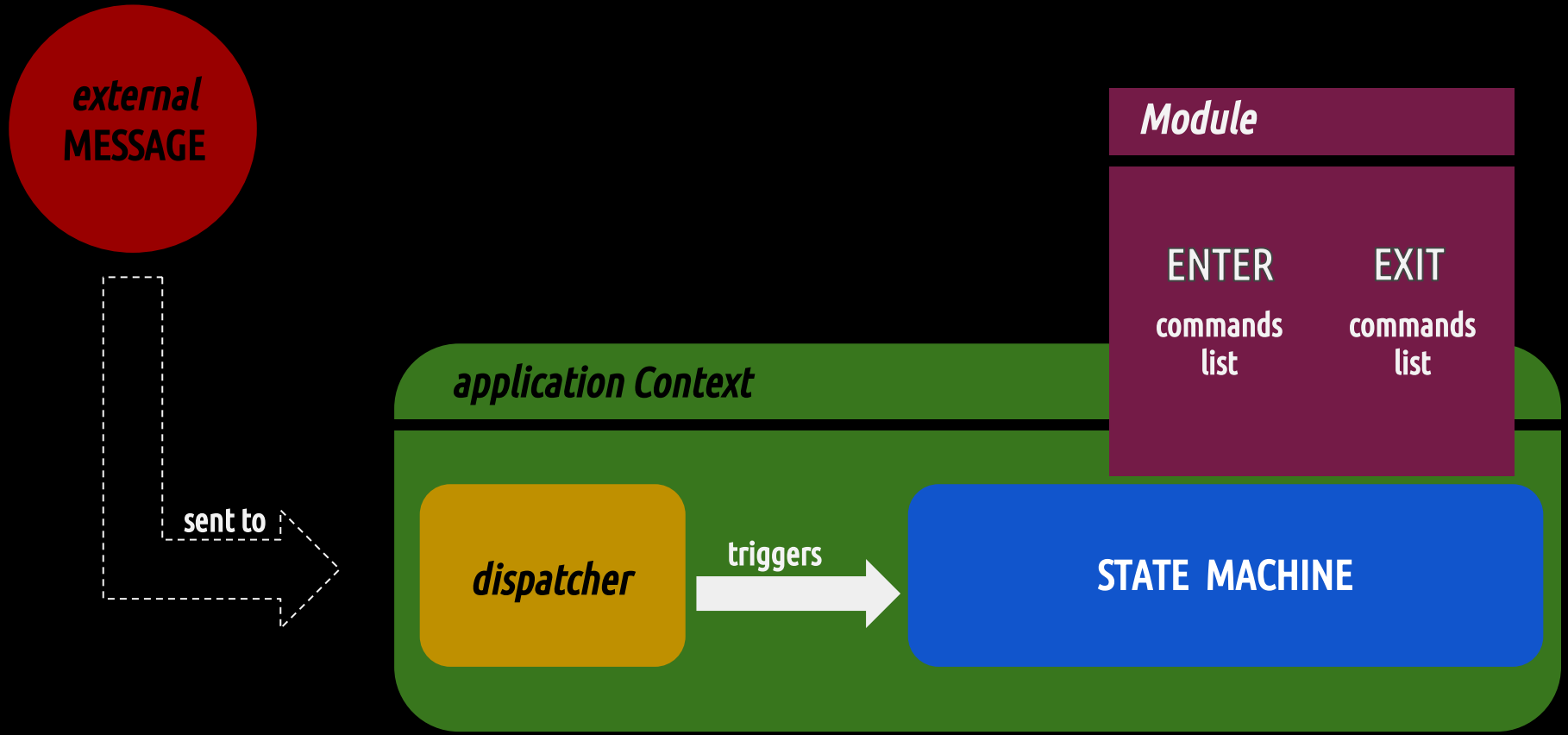
Context state machine



Context state machine



Context state machine



Context states

state flow described with DSL

```
<root name="applicationContext">

    <state id="assemblingStart" ref="applicationContext.state.ASSEMBLING_START">
        <enter command-class="hex.ioc.SetUpApplication"/>
    </state>

</root>
```

Context states

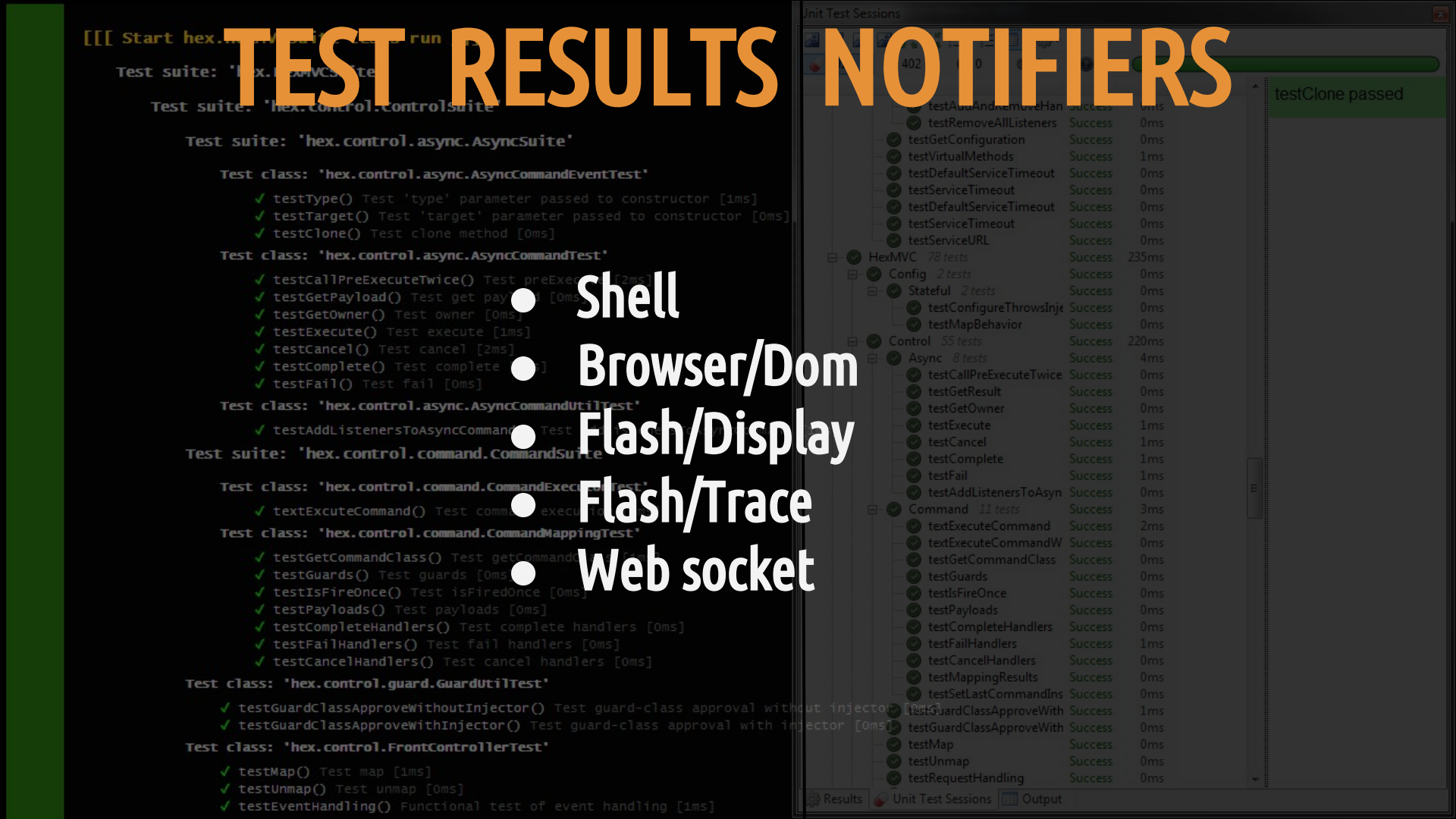
module scope for execution

```
<root name="applicationContext">

    <state id="objectsBuilt" ref="applicationContext.state.OBJECTS_BUILT">
        <enter command-class="hex.ioc.InitModule" context-owner="module"/>
    </state>

    <module id="module" type="hex.ioc.MockModule"/>

</root>
```

hexUnit

UNIT TEST FRAMEWORK

TEST RESULTS NOTIFIERS



- Shell
- Browser/Dom
- Flash/Display
- Flash/Trace
- Web socket

hexCore

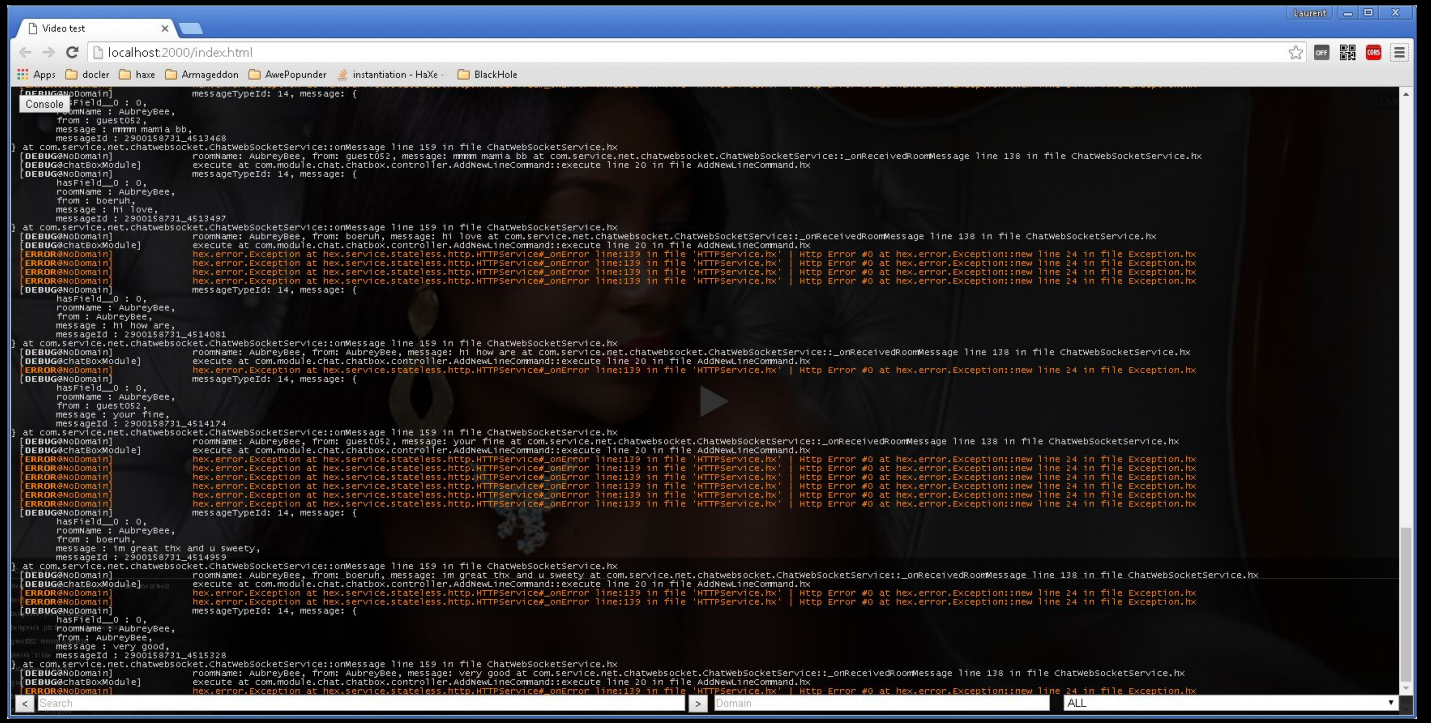
Logging API

filter logs by:

- Domain
- Levels

web console

- search text

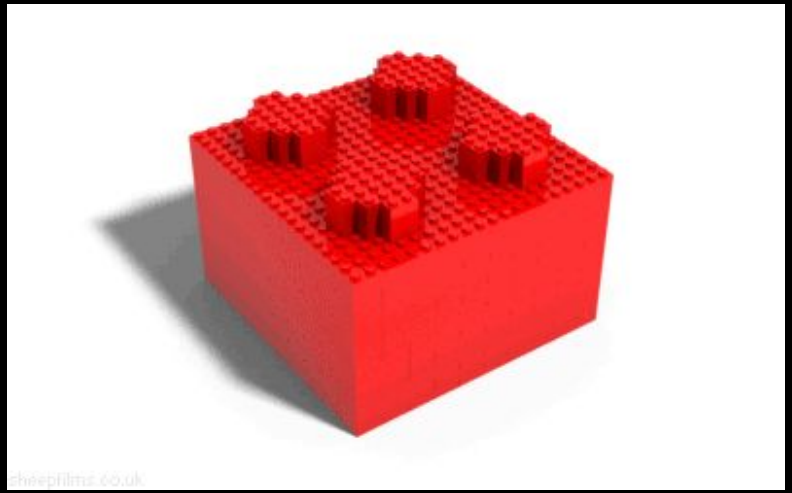


hexMachina

What’s next?

Recursive composition

context agregates contexts, that agregates contexts, that...



Contexts hierarchy

hierarchy is handled by DSL

```
<root name="applicationContext">

    <import name="videoContext" file="folder/video.xml"/>
    <import name="chatContext" file="folder/chat.xml"/>

    <module id="application" type="com.docler.Application"/>
        <argument ref="videoContext.videoModule"/>
        <argument ref="chatContext.chatModule"/>
    </module>
</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

DOCLER HOLDING

WE HIRE HAXE DEVS

CONTACT US

DOCLER HOLDING