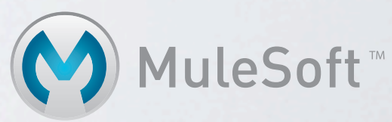
DEVKIT MONTHLY WORKSHOP

December 2011

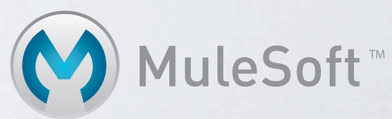


AGENDA

- What is the DevKit? ~ 5 min
- Deep Dive: DevKit Architecture 101 ~ 25 min
- Q&A ~ 20 min
- Upcoming Features: Studio Integration ~ 10 min

MuleSoft™

WHAT IS THE DEVKIT?



WHAT IS THE DEVKIT?

The DevKit is a tool for accelerating the development of Mule Modules.

WHAT IS THE DEVKIT?

The DevKit is a tool for **accelerating** the development of Mule Modules.

MuleSoft™

WHAT IS THE DEVKIT?

The DevKit is a tool for **accelerating** the development of Mule Modules.

We dont do everything
for you, we just make you
do what you already
do... faster.

MuleSoft™

WHAT IS THE DEVKIT?

The DevKit accelerates by...

WHAT IS THE DEVKIT?

The DevKit accelerates by...

Abstracting Mule internals

WHAT IS THE DEVKIT?

The DevKit accelerates by...

Abstracting Mule internals

Generating boiler-plate code

WHAT IS THE DEVKIT?

The DevKit accelerates by...

Abstracting Mule internals

Generating boiler-plate code

Providing initial templates

MuleSoft™

WHAT IS THE DEVKIT?

The DevKit accelerates by...

Abstracting Mule internals

Generating boiler-plate code

Providing initial templates

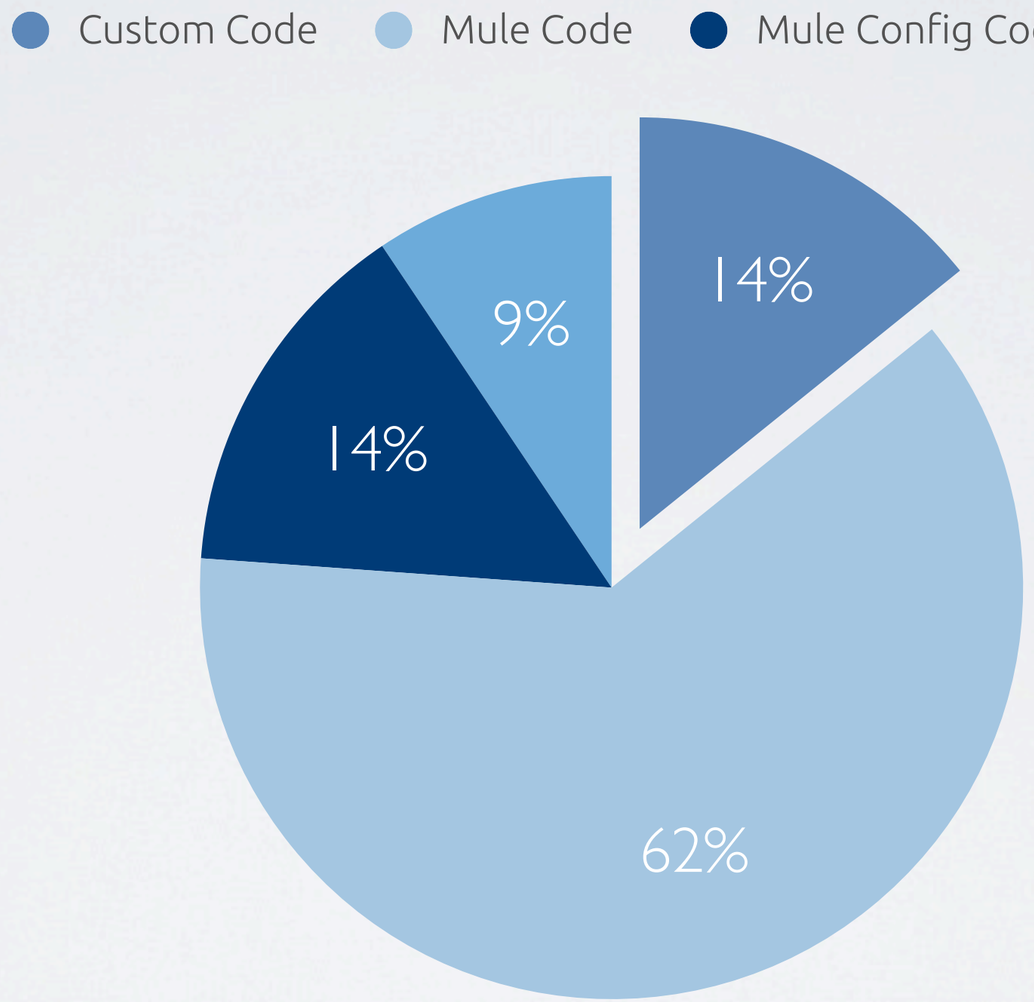
Generating documentation

Abstracting Mule internals

```
@Module(name="mod")
public class MyModule {
  @Configurable
  private String

  @Processor
  public void doSomething(String param)
  { ... }
}
```

The DevKit offers an annotation model that mimics Mule concepts such as MessageProcessors, Transformers, etc.

Generating boiler-plate code



Salesforce Connector LoC Breakdown

MuleSoft™

Providing initial templates

The DevKit offers skeleton project generation via Maven Archetypes.

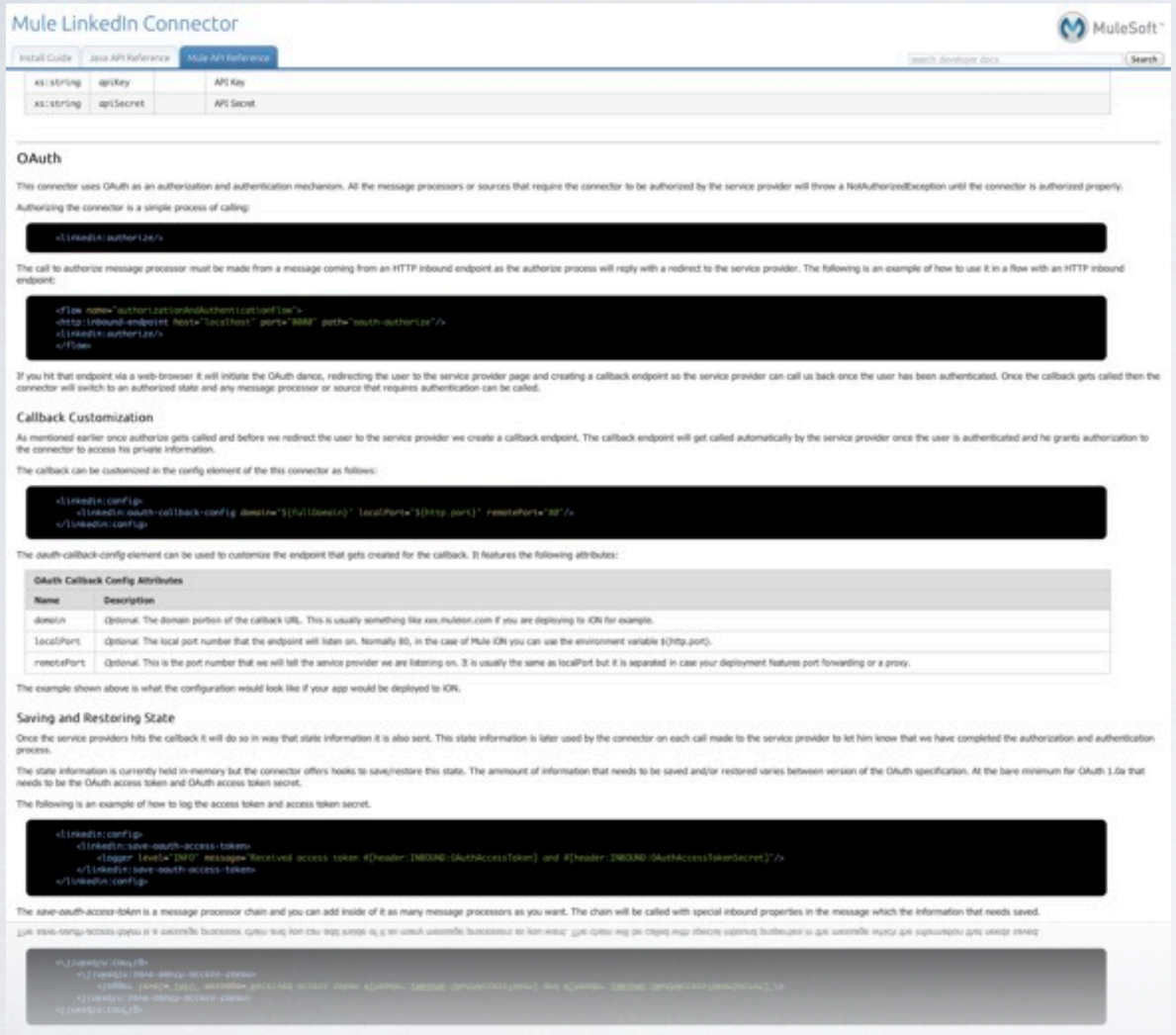
Providing initial templates

The DevKit offers skeleton project generation via Maven Archetypes.

Generating documentation

The DevKit is capable of generating fully comprehensive documentation right from the source code of the module.

Generating documentation



Mule LinkedIn Connector

MuleSoft

Install Guide | Java API Reference | Mule API Reference

search developer docs | Search

xs:string	apiKey		API Key
xs:string	apiSecret		API Secret

OAuth

This connector uses OAuth as an authorization and authentication mechanism. All the message processors or sources that require the connector to be authorized by the service provider will throw a NotAuthorizedException until the connector is authorized properly.

Authorizing the connector is a simple process of calling:

```
<linkedin:authorize/>
```

The call to authorize message processor must be made from a message coming from an HTTP inbound endpoint as the authorize process will reply with a redirect to the service provider. The following is an example of how to use it in a flow with an HTTP inbound endpoint:

```
<flow name="authorizationAndAuthenticationFlow">
    <http:inbound-endpoint host="localhost" port="8080" path="oauth-authorize"/>
    <linkedin:authorize/>
</flow>
```

If you hit that endpoint via a web-browser it will initiate the OAuth dance, redirecting the user to the service provider page and creating a callback endpoint so the service provider can call us back once the user has been authenticated. Once the callback gets called then the connector will switch to an authorized state and any message processor or source that requires authentication can be called.

Callback Customization

As mentioned earlier once authorize gets called and before we redirect the user to the service provider we create a callback endpoint. The callback endpoint will get called automatically by the service provider once the user is authenticated and he grants authorization to the connector to access his private information.

The callback can be customized in the config element of the this connector as follows:

```
<linkedin:config>
    <linkedin:oauth-callback-config domain="${fullDomain}" localPort="${http.port}" remotePort="80"/>
</linkedin:config>
```

The *oauth-callback-config* element can be used to customize the endpoint that gets created for the callback. It features the following attributes:

OAuth Callback Config Attributes	
Name	**Description**
domain	*Optional.* The domain portion of the callback URL. This is usually something like xxx.mulesoft.com if you are deploying to iON for example.
localPort	*Optional.* The local port number that the endpoint will listen on. Normally 80, in the case of Mule iON you can use the environment variable ${http.port}.
remotePort	*Optional.* This is the port number that we will tell the service provider we are listening on. It is usually the same as localPort but it is separated in case your deployment features port forwarding or a proxy.

The example shown above is what the configuration would look like if your app would be deployed to iON.

Saving and Restoring State

Once the service providers hits the callback it will do so in way that state information it is also sent. This state information is later used by the connector on each call made to the service provider to let him know that we have completed the authorization and authentication process.

The state information is currently held in-memory but the connector offers hooks to save/restore this state. The ammount of information that needs to be saved and/or restored varies between version of the OAuth specification. At the bare minimum for OAuth 1.0a that needs to be the OAuth access token and OAuth access token secret.

The following is an example of how to log the access token and access token secret.

```
<linkedin:config>
    <linkedin:save-oauth-access-token>
        <logger level="INFO" message="Received access token #[header:INBOUND:OAuthAccessToken] and #[header:INBOUND:OAuthAccessTokenSecret]"/>
    </linkedin:save-oauth-access-token>
</linkedin:config>
```

The *save-oauth-access-token* is a message processor chain and you can add inside of it as many message processors as you want. The chain will be called with special inbound properties in the message which the information that needs saved.

The DevKit is capable of generating fully comprehensive documentation right from the source code of the module.

DEEP DIVE: DEVKIT ARCHITECTURE 101

MuleSoft™

TECHNOLOGY BEHIND THE DEVKIT

Abstracting Mule internals

Generating boiler-plate code

Providing initial templates

Generating documentation

MuleSoft™

TECHNOLOGY BEHIND THE DEVKIT

Abstracting Mule internals ⟶ Java Annotations (JSR-175)

Generating boiler-plate code

Providing initial templates

Generating documentation

MuleSoft™

TECHNOLOGY BEHIND THE DEVKIT

Abstracting Mule internals → Java Annotations (JSR-175)

Generating boiler-plate code → Java APT (JSR-269)

Providing initial templates

Generating documentation

MuleSoft™

TECHNOLOGY BEHIND THE DEVKIT

Abstracting Mule internals → Java Annotations (JSR-175)

Generating boiler-plate code → Java APT (JSR-269)

Providing initial templates → Maven Archetypes

Generating documentation

MuleSoft™

TECHNOLOGY BEHIND THE DEVKIT

Abstracting Mule internals → Java Annotations (JSR-175)

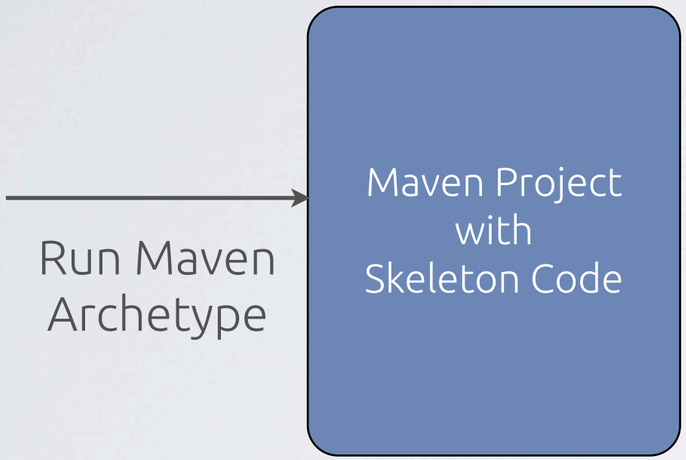
Generating boiler-plate code → Java APT (JSR-269)

Providing initial templates → Maven Archetypes

Generating documentation → Java Doclet (JSR-260)

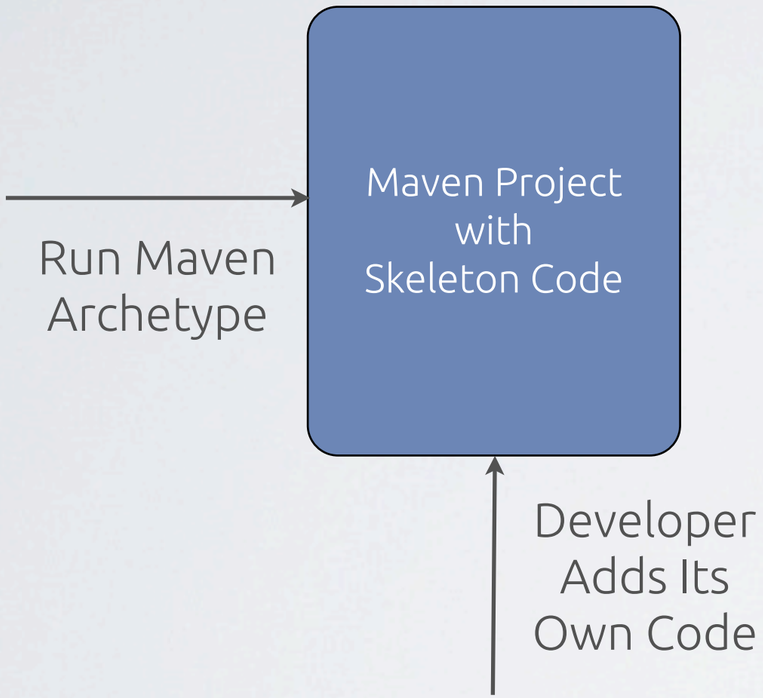
MuleSoft™

DEVKIT WORK PIPELINE



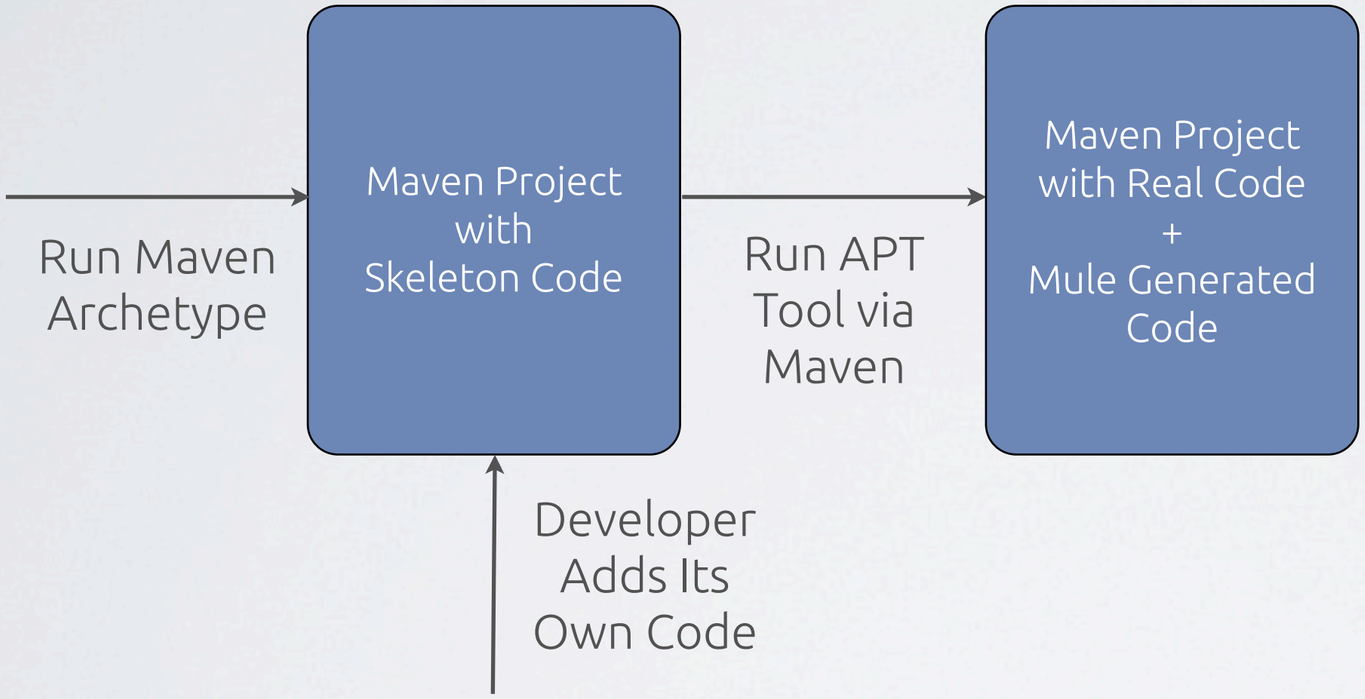
MuleSoft™

DEVKIT WORK PIPELINE



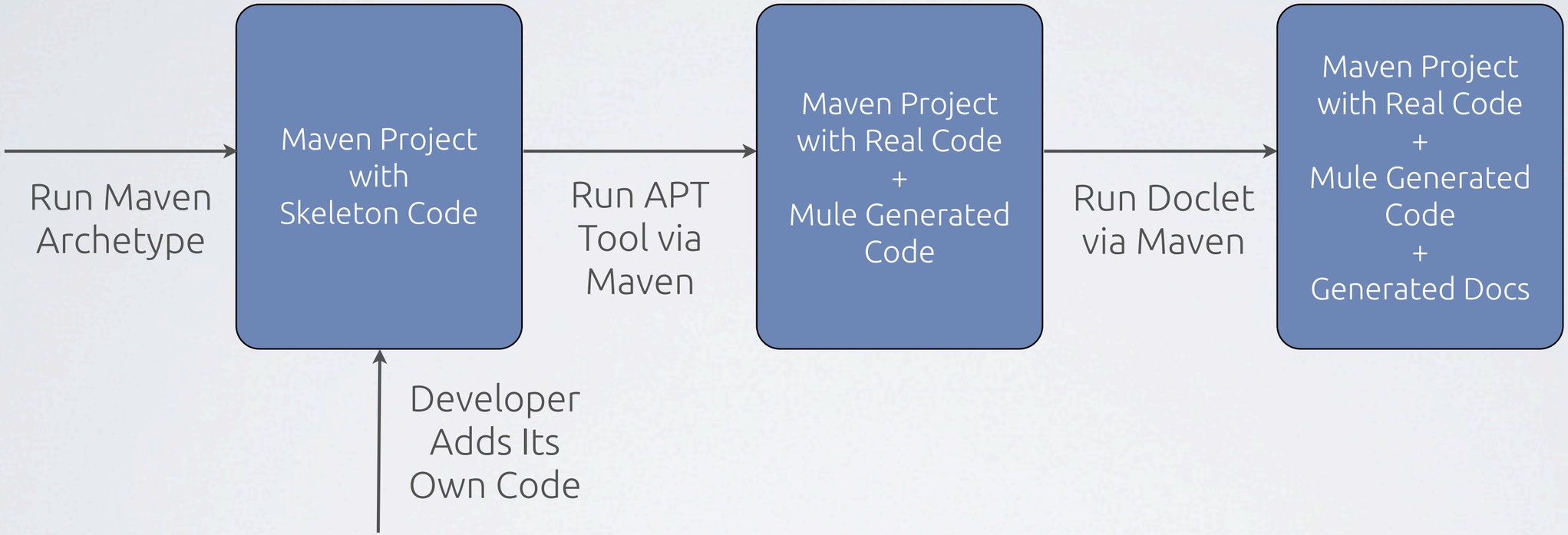
MuleSoft™

DEVKIT WORK PIPELINE



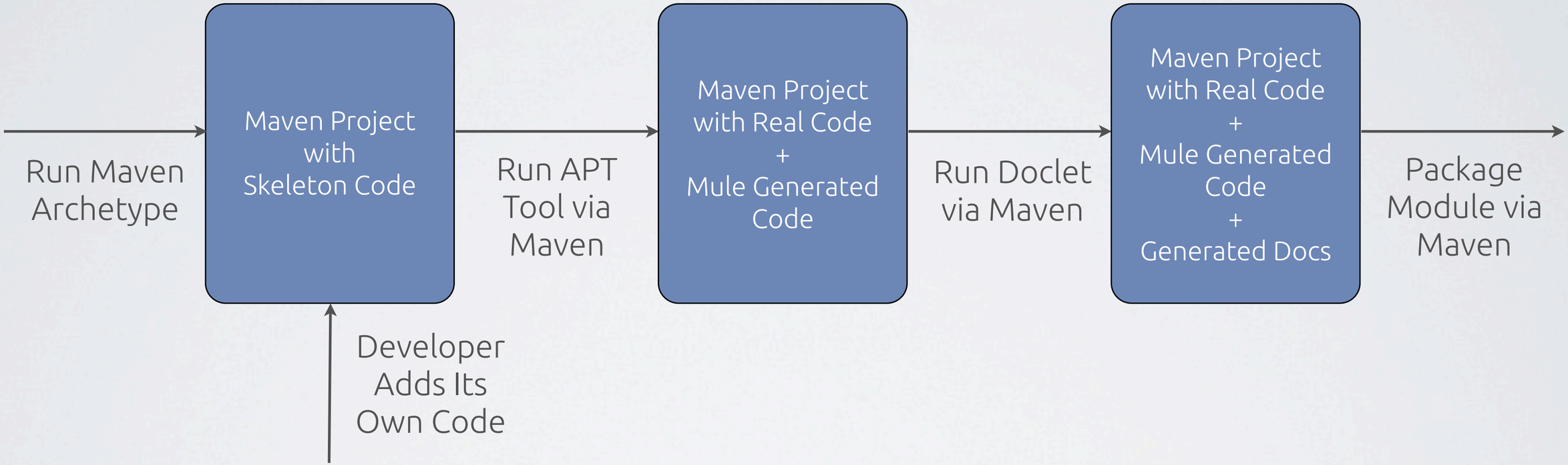
MuleSoft™

DEVKIT WORK PIPELINE



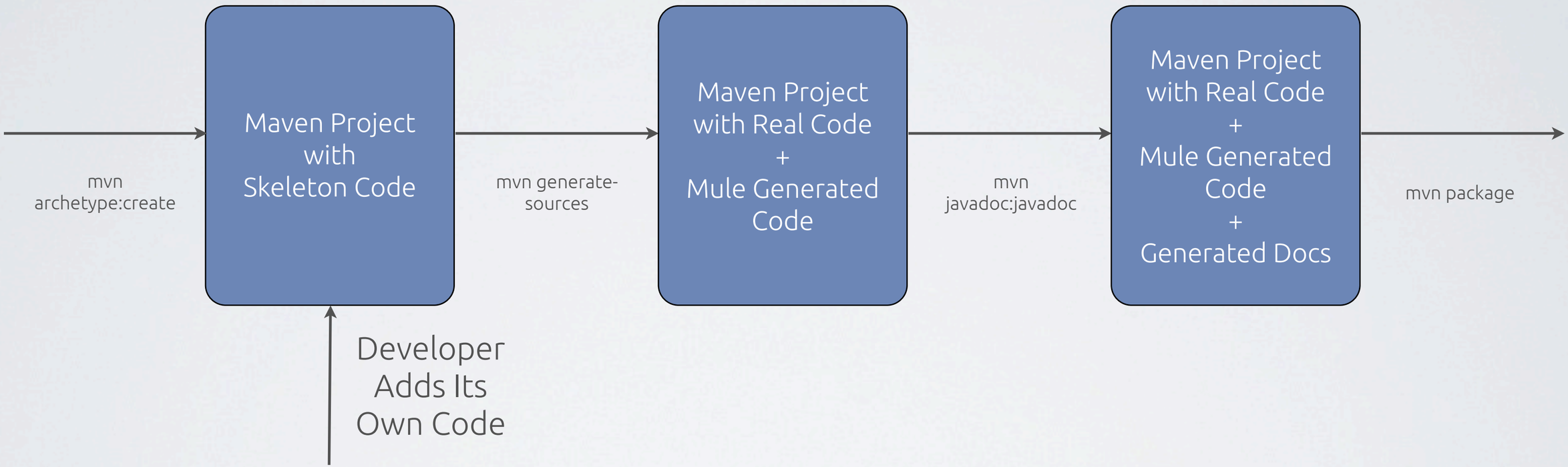
MuleSoft™

DEVKIT WORK PIPELINE



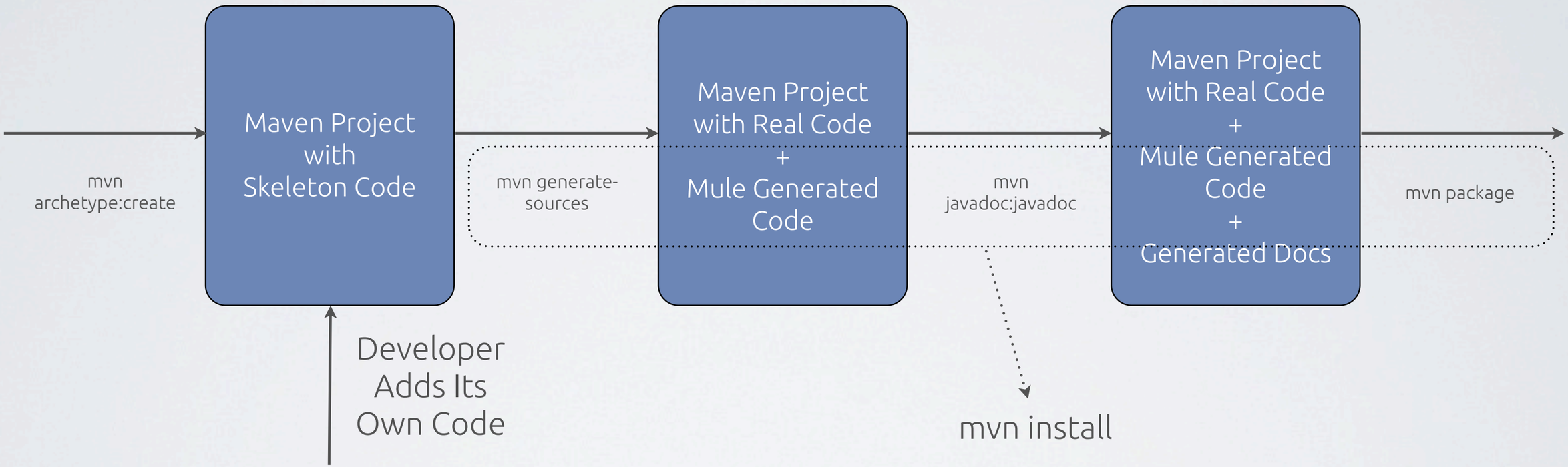
MuleSoft™

DEVKIT WORK PIPELINE



MuleSoft™

DEVKIT WORK PIPELINE



MuleSoft™

DEVKIT ANNOTATION SET

@Default

@Connector

@Disconnect

@Source

@OutboundHeaders

@Module

@Connect

@OAuth2

@Payload

@OAuth

DEVKIT ANNOTATION SET

@Stop

@Start

@ValidateConnection

@Processor

@InvalidateConnectionOn

@Configurable

@ConnectionKey

@Display

@Optional

@InvocationHeaders

MuleSoft™

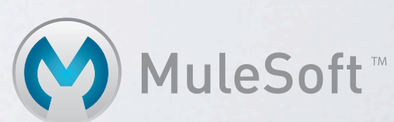
DEVKIT ANNOTATION SET

```
@Module(name=”mod”)
public class MyModule {
   @Configurable
   private String modVariable;

   @Processor
   public void doSomething(String param)
   { ... }
}
```

DEVKIT ANNOTATION SET

```
@Module(name="mod")
public class MyModule {
   @Configurable
   private String modVariable;

   @Processor
   public void doSomething(String param)
   { ... }
}
```

```
<mod:config modVariable="xxx"/>

<flow name="modtest">
    <mod:do-something param="yyy"/>
</flow>
```



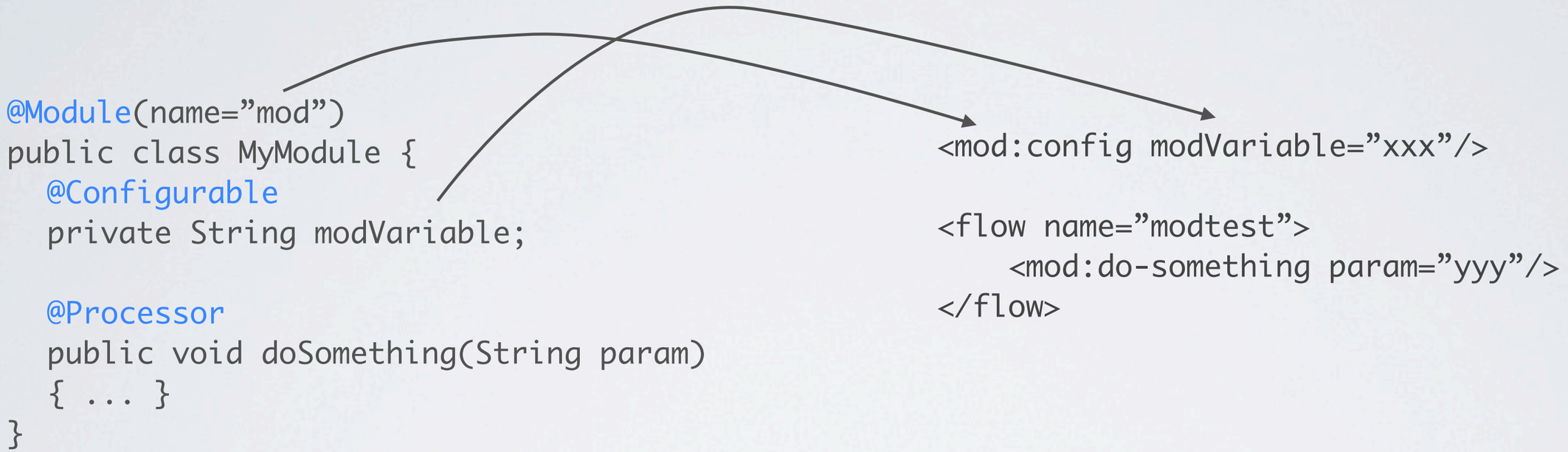
DEVKIT ANNOTATION SET

```
@Module(name="mod")
public class MyModule {
   @Configurable
   private String modVariable;

   @Processor
   public void doSomething(String param)
   { ... }
}
```

```
<mod:config modVariable="xxx"/>

<flow name="modtest">
    <mod:do-something param="yyy"/>
</flow>
```

MuleSoft™

DEVKIT ANNOTATION SET



```
@Module(name="mod")
public class MyModule {
  @Configurable
  private String modVariable;

  @Processor
  public void doSomething(String param)
  { ... }
}
```

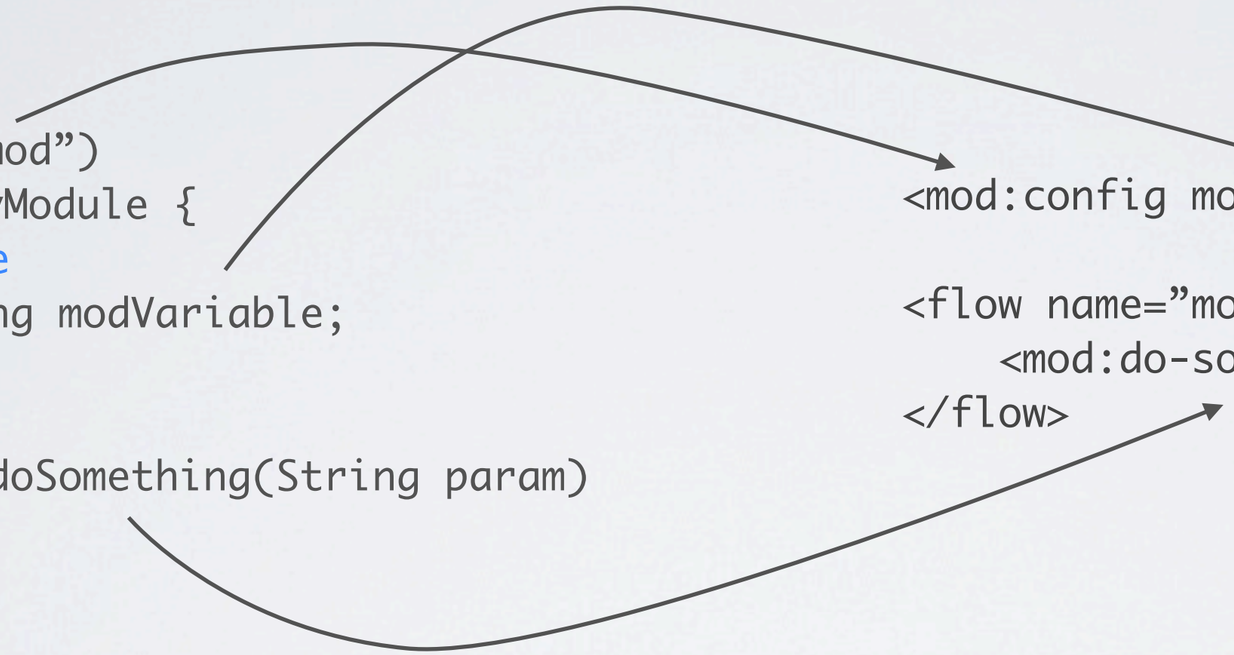
```
<mod:config modVariable="xxx"/>

<flow name="modtest">
    <mod:do-something param="yyy"/>
</flow>
```

MuleSoft™

DEVKIT ANNOTATION SET

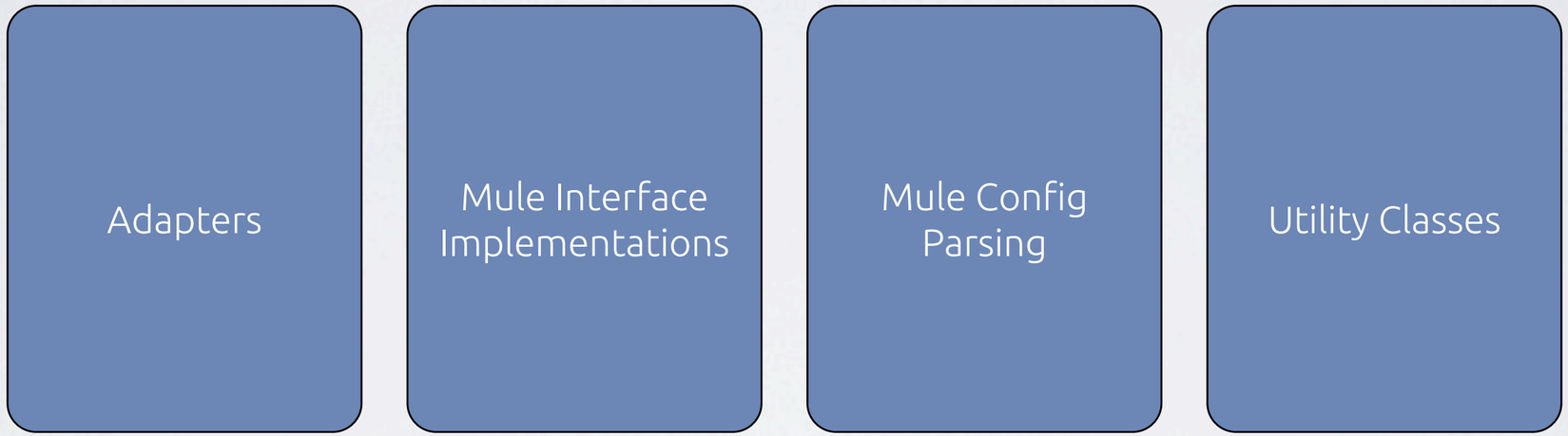
```
@Module(name="mod")
public class MyModule {
  @Configurable
  private String modVariable;

  @Processor
  public void doSomething(String param)
  { ... }
}
```

```
<mod:config modVariable="xxx"/>

<flow name="modtest">
    <mod:do-something param="yyy"/>
</flow>
```

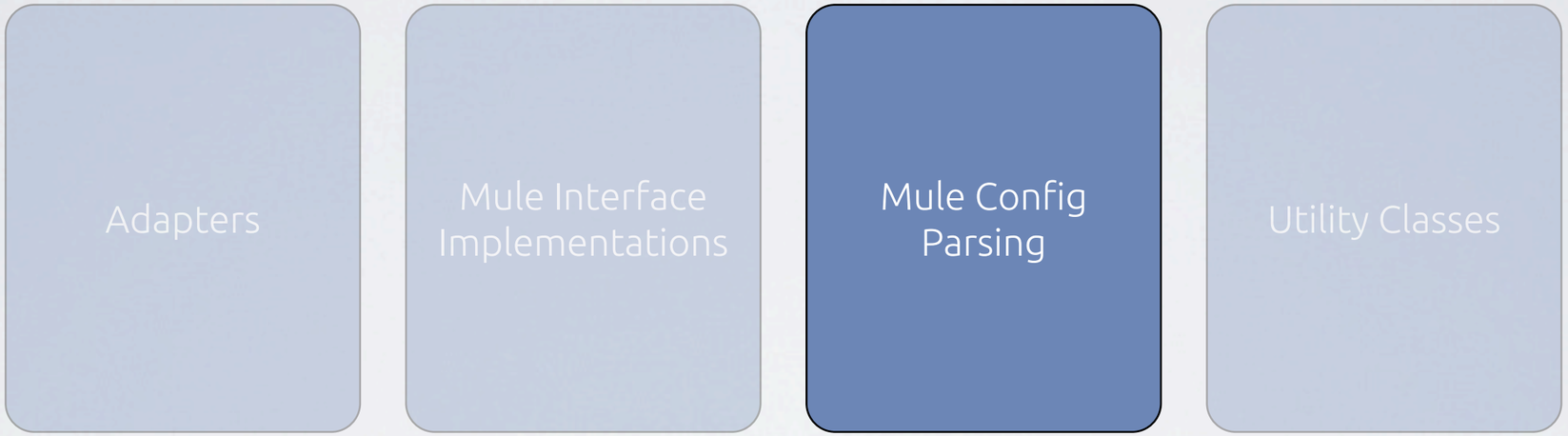


MuleSoft™

DEVKIT CODE GENERATION



DEVKIT CODE GENERATION

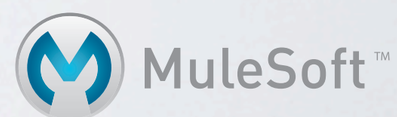


Implement Spring XML interfaces for parsing module-specific portions of Mule configs.

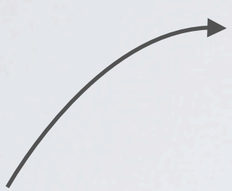
MuleSoft™

DEVKIT CONFIG PARSING

```
<mod:config modVariable=”xxx”/>

<flow name=”modtest”>
    <mod:do-something param=”yyy”/>
</flow>
```



DEVKIT CONFIG PARSING




```
<mod:config modVariable=”xxx”/>

<flow name=”modtest”>
    <mod:do-something param=”yyy”/>
</flow>
```

MuleSoft™

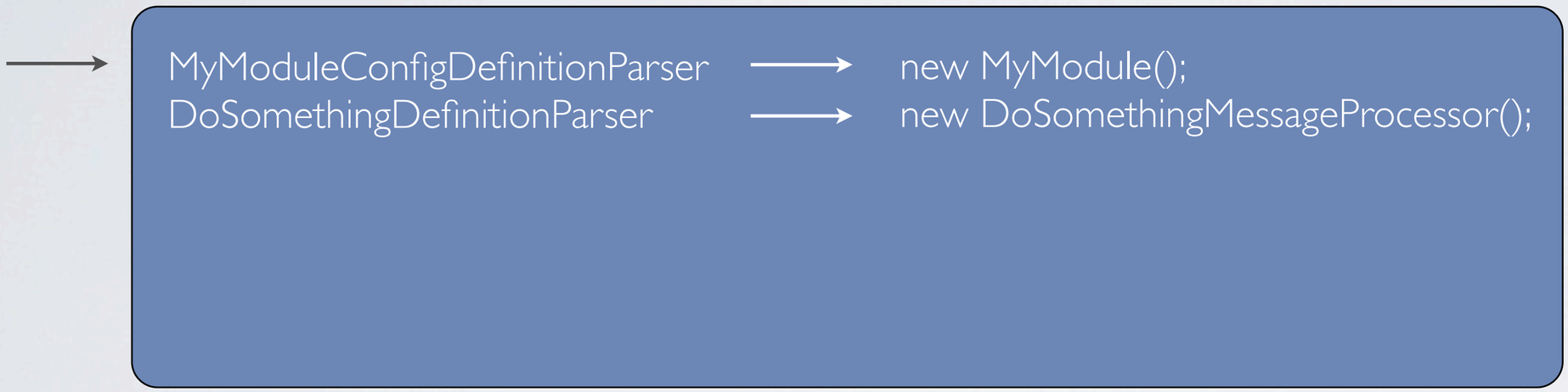
DEVKIT CONFIG PARSING

MyModuleNamespaceHandler

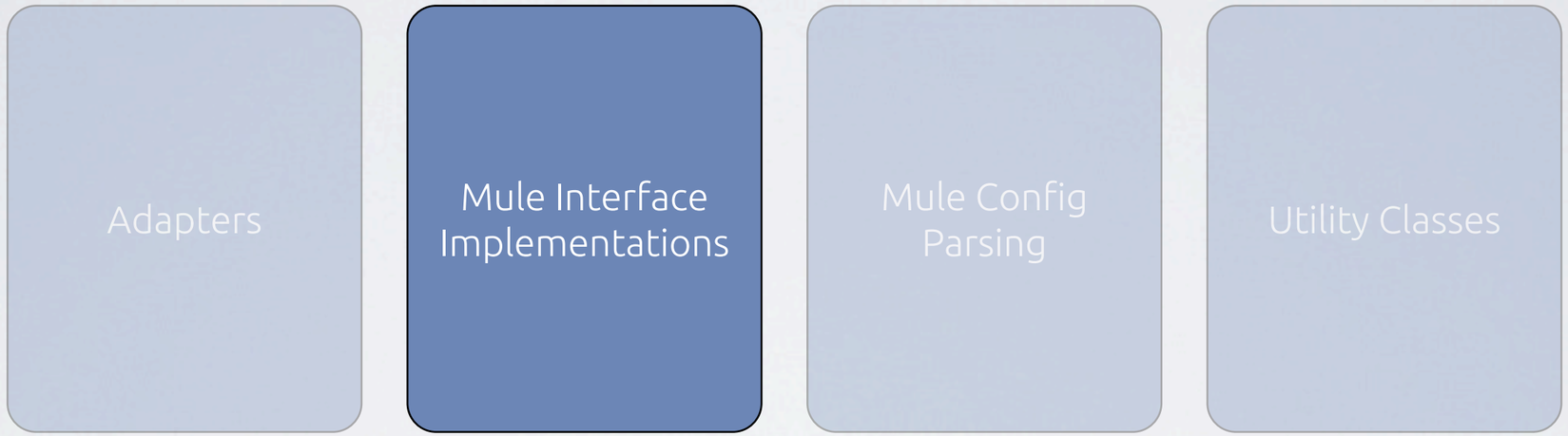
```
<mod:config modVariable=”xxx”/>

<flow name=”modtest”>
    <mod:do-something param=”yyy”/>
</flow>
```

:config → MyModuleConfigDefinitionParser
:do-something → DoSomethingDefinitionParser

DEVKIT CONFIG PARSING



DEVKIT CODE GENERATION



Classes that implement Mule interfaces like MessageProcessor, MessageSource, Transformer, etc.

MuleSoft™

DEVKIT INTERFACE IMPLEMENTATIONS

```
<mod:config modVariable=”xxx”/>

<flow name=”modtest”>
    <mod:do-something param=”yyy”/>
</flow>
```

DEVKIT INTERFACE IMPLEMENTATIONS

```
<mod:config modVariable=”xxx”/>

<flow name=”modtest”>
    <mod:do-something param=”yyy”/>
</flow>
```

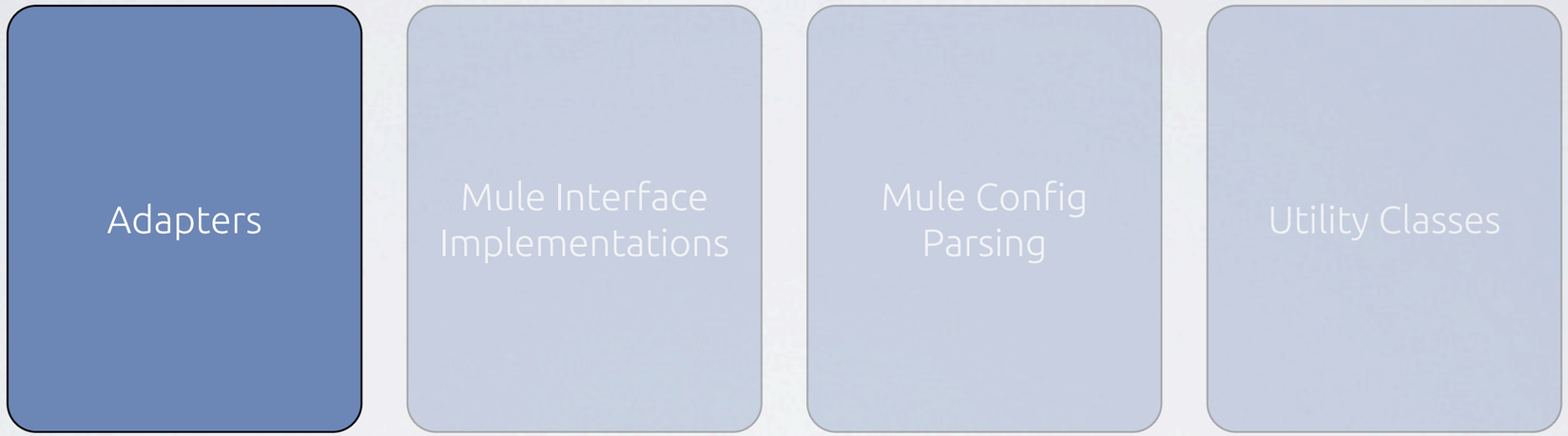
DoSomethingMessageProcessor

DEVKIT INTERFACE IMPLEMENTATIONS

```
<mod:config modVariable=”xxx”/>

<flow name=”modtest”>
    <mod:do-something param=”yyy”/>
</flow>
```

MuleSoft™

DEVKIT CODE GENERATION



Adapters infuse POJO with extra capabilities like OAuth clients, connection management, etc.

MuleSoft™

DEVKIT ADAPTERS

```
@Module(name="mod")
public class MyModule {
   @Configurable
   private String modVariable;

   @Start
   public void myStart() { ... }

   @Stop
   public void myStop() { ... }

   @Processor
   public void doSomething(String param)
   { ... }
}
```

DEVKIT ADAPTERS

```
@Module(name="mod")
public class MyModule {
   @Configurable
   private String modVariable;

   @Start
   public void myStart() { ... }

   @Stop
   public void myStop() { ... }

   @Processor
   public void doSomething(String param)
   { ... }
}
```

DEVKIT ADAPTERS

```
@Module(name="mod")
public class MyModule {
  @Configurable
  private String modVariable;

  @Start
  public void myStart() { ... }

  @Stop
  public void myStop() { ... }

  @Processor
  public void doSomething(String param)
  { ... }
}
```

Lifecycle Annotations

MuleSoft™

DEVKIT ADAPTERS

```
@Module(name="mod")
public class MyModule {
  @Configurable
  private String modVariable;

  @Start
  public void myStart() { ... }

  @Stop
  public void myStop() { ... }

  @Processor
  public void doSomething(String param)
  { ... }
}
```

```
public class MyModuleLifecycleAdapter
     extends MyModule
  implements Startable, Stoppable {

  public void start() throws MuleException {
    super.myStart();
  }

  public void stop() throws MuleException {
    super.myStop();
  }
}
```

DEVKIT ADAPTERS

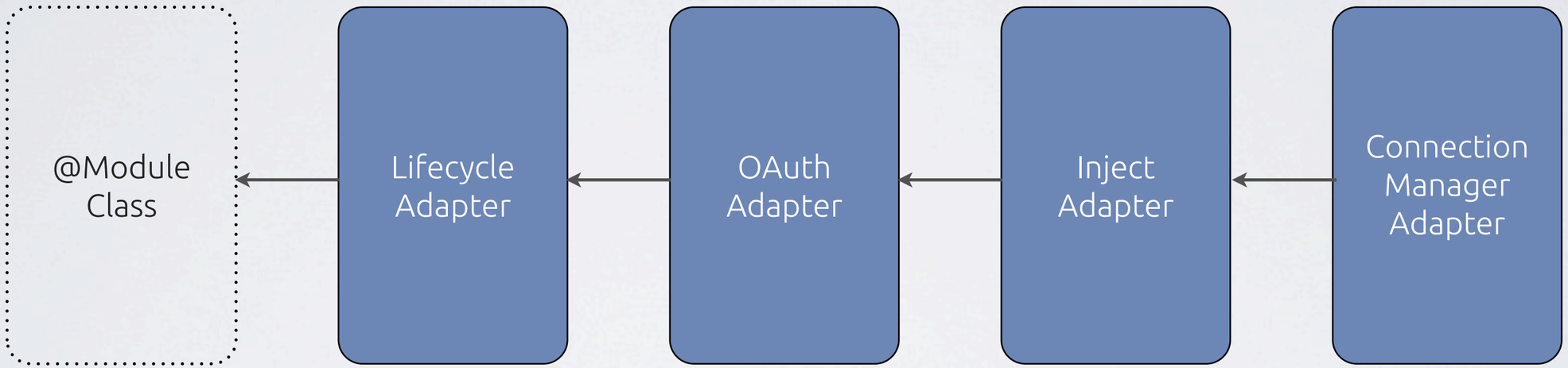
```
@Module(name="mod")
public class MyModule {
  @Configurable
  private String modVariable;

  @Start
  public void myStart() { ... }

  @Stop
  public void myStop() { ... }

  @Processor
  public void doSomething(String param)
  { ... }
}
```

```
public class MyModuleLifecycleAdapter
     extends MyModule
  implements Startable, Stoppable {

  public void start() throws MuleException {
    super.myStart();
  }

  public void stop() throws MuleException {
    super.myStop();
  }
}
```

Mule Interfaces

MuleSoft™

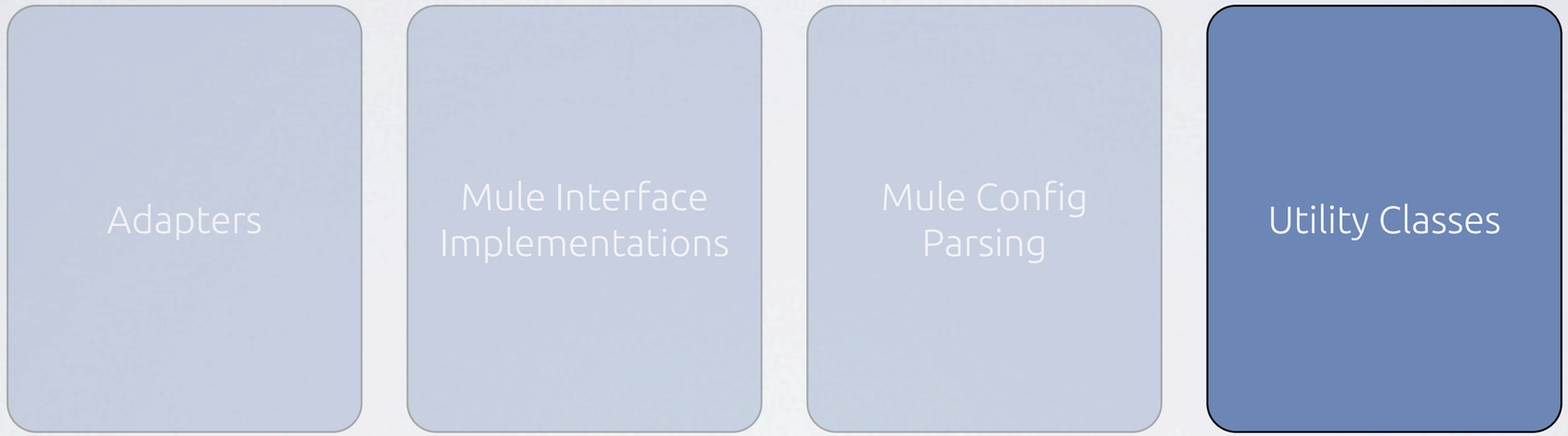
DEVKIT ADAPTERS



Java Class Hierarchy

MuleSoft™

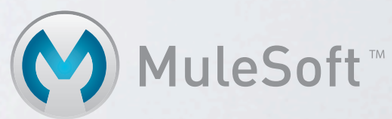
DEVKIT CODE GENERATION



Utility classes most of the time provide reusable functionality not available in Mule.

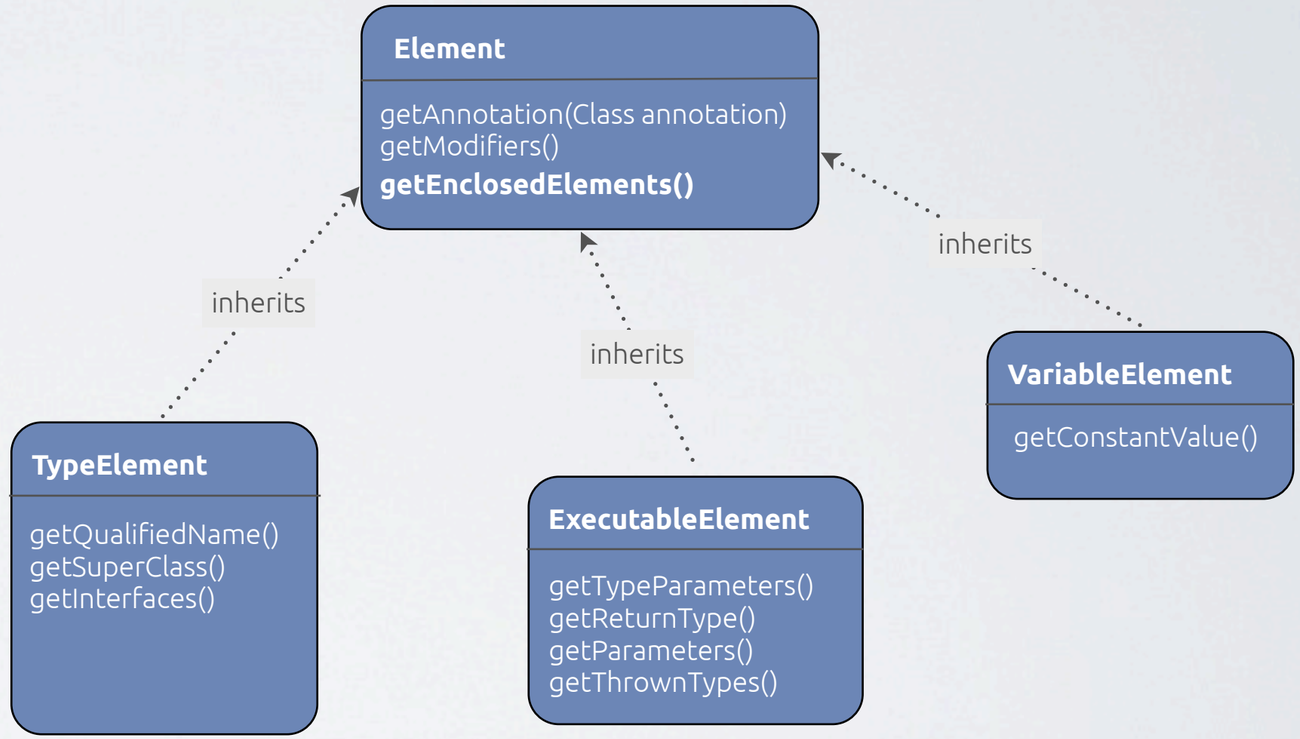
MuleSoft™

HOW CODE GETS GENERATED?



JAVA APT MODEL

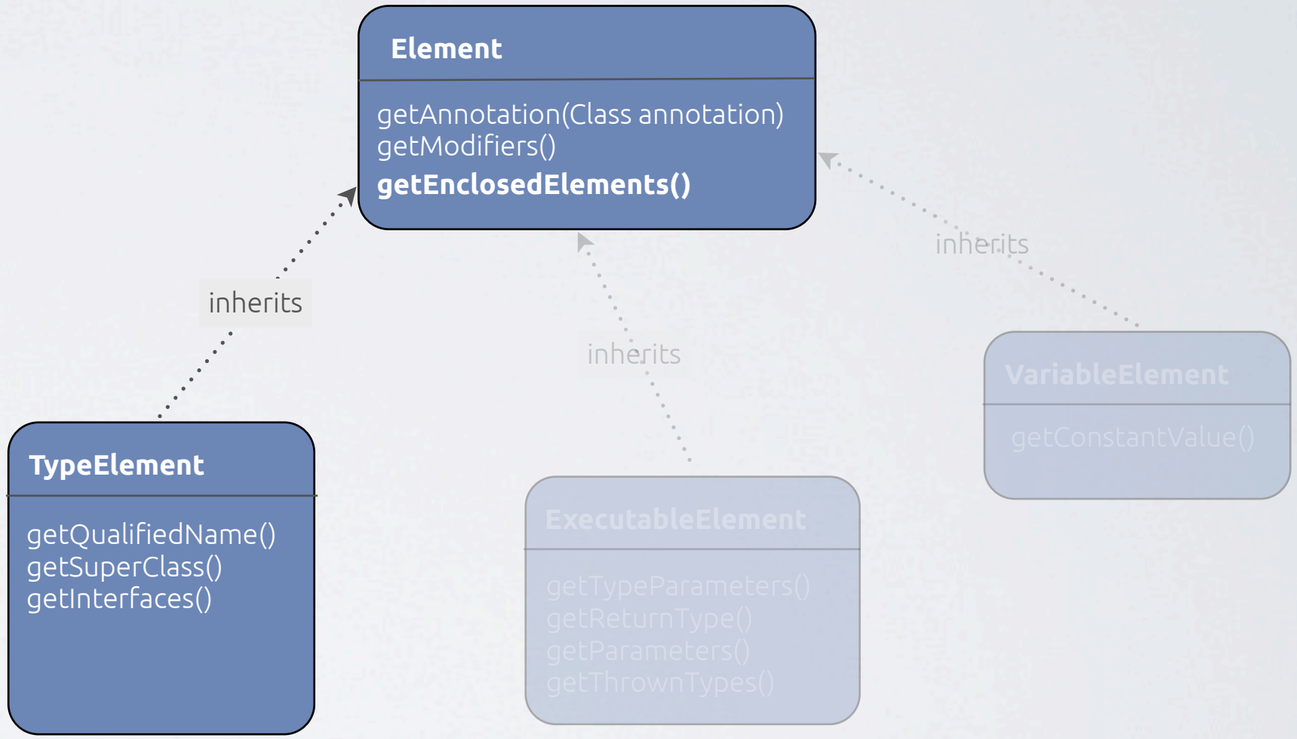
```
@Module(name=”mod”)
public class MyModule {
  @Configurable
  private String modVariable;

  @Start
  public void myStart() { ... }

  @Stop
  public void myStop() { ... }

  @Processor
  public void doSomething(String param)
  { ... }
}
```



MuleSoft™

JAVA APT MODEL

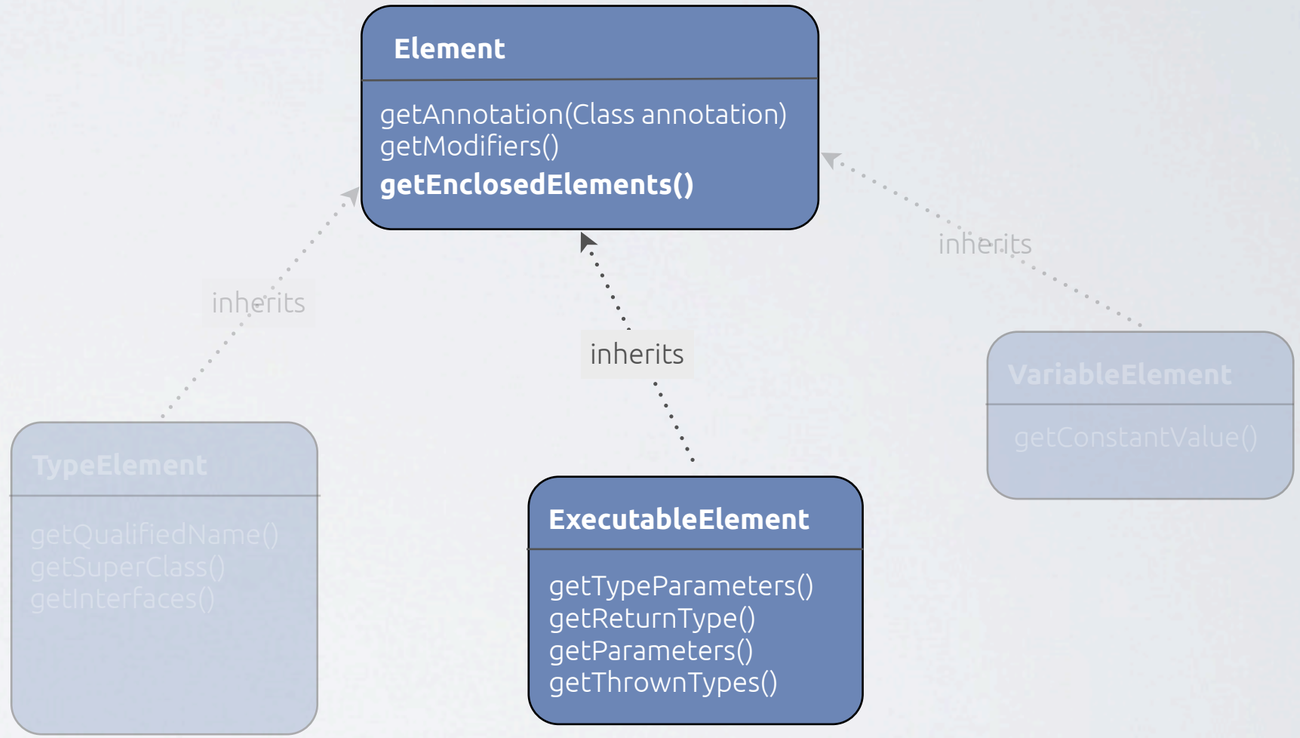
```
@Module(name="mod")
public class MyModule {
   @Configurable
   private String modVariable;

   @Start
   public void myStart() { ... }

   @Stop
   public void myStop() { ... }

   @Processor
   public void doSomething(String param)
   { ... }
}
```



MuleSoft™

JAVA APT MODEL

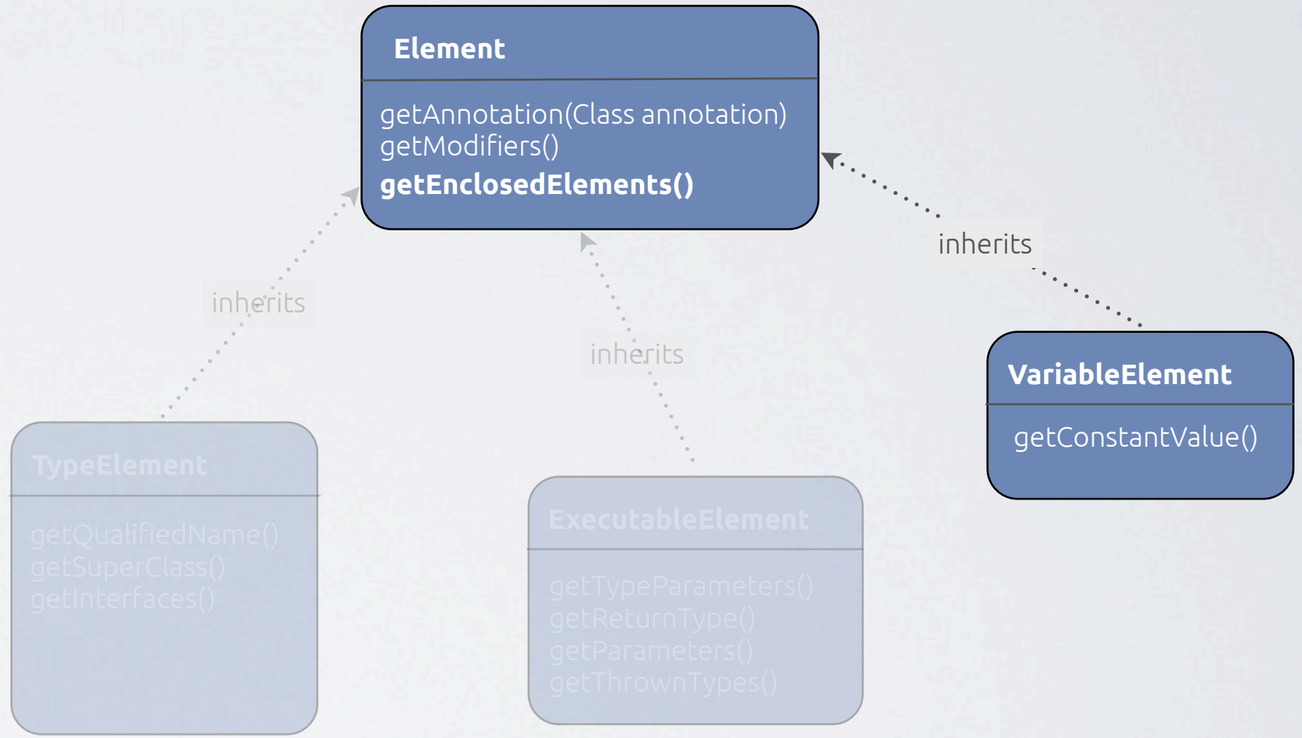
```
@Module(name=”mod”)
public class MyModule {
   @Configurable
   private String modVariable;

   @Start
   public void myStart() { ... }

   @Stop
   public void myStop() { ... }

   @Processor
   public void doSomething(String param)
   { ... }
}
```



MuleSoft™

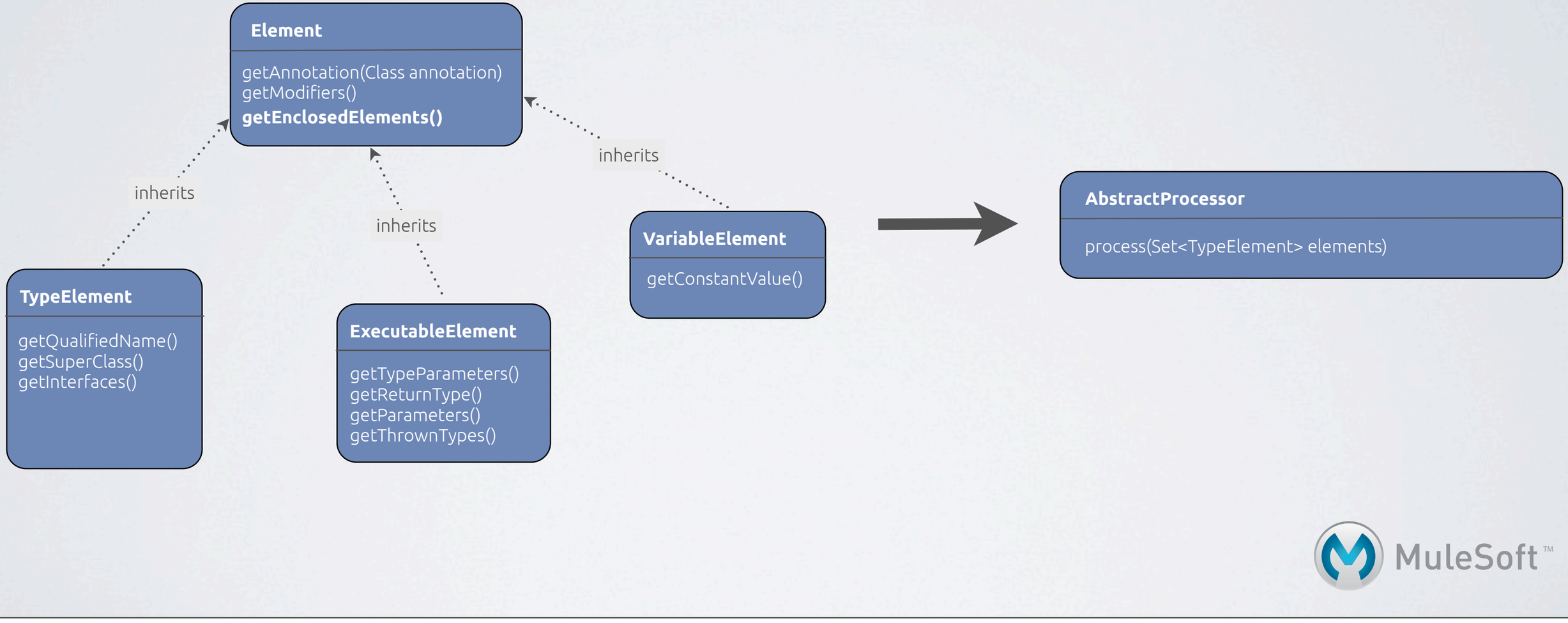
JAVA APT MODEL

```
@Module(name="mod")
public class MyModule {
   @Configurable
   private String modVariable;

   @Start
   public void myStart() { ... }

   @Stop
   public void myStop() { ... }

   @Processor
   public void doSomething(String param)
   { ... }
}
```

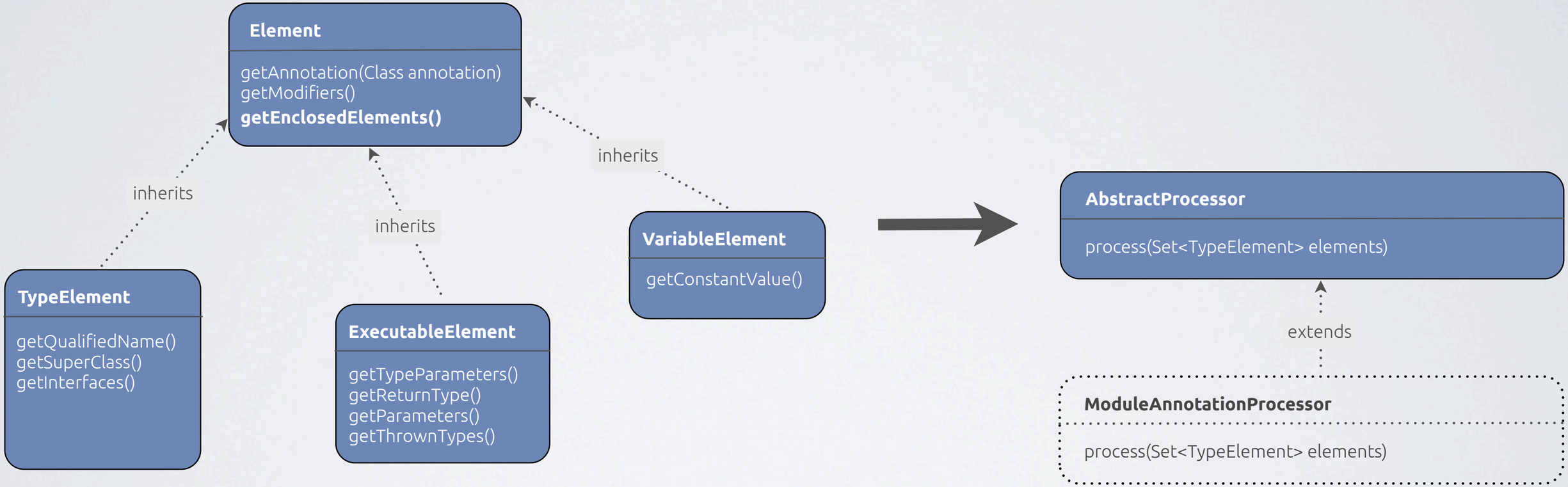


MuleSoft™

JAVA APT MODEL

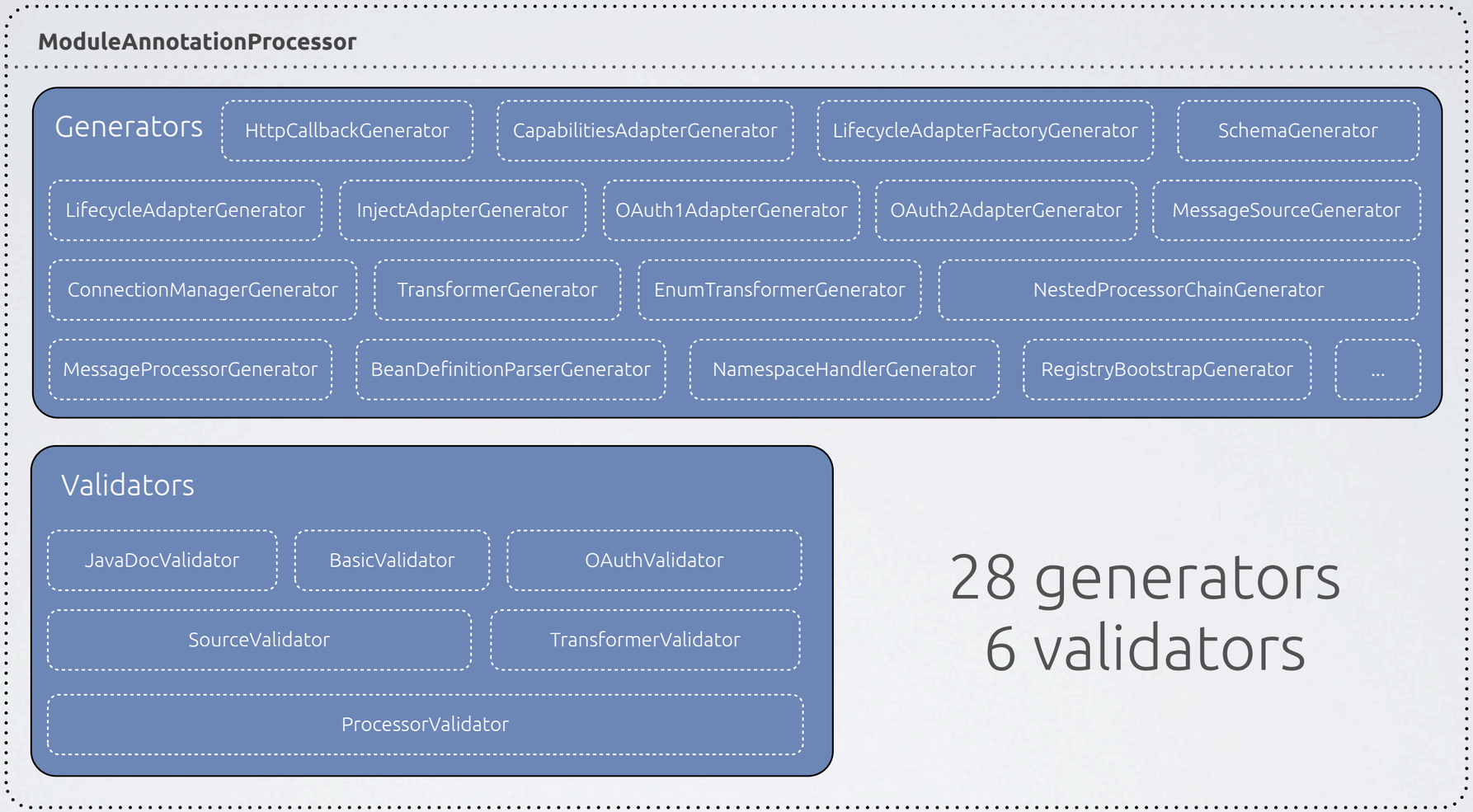


JAVA APT MODEL



MuleSoft™

MODULE ANNOTATION PROCESSOR



28 generators
6 validators

MuleSoft™

MODULE ANNOTATION PROCESSOR

ModuleAnnotationProcessor

process(Set<TypeElement> elements)

```
@Override
public boolean process(Set<? extends TypeElement> annotations, RoundEnvironment env) {
    ...

    for (Validator validator : getValidators()) {
        for (TypeElement e : typeElements) {
            try {
                if (validator.shouldValidate(e, context)) {
                    validator.validate(e, context);
                }
            } catch (ValidationException tve) {
                ...
            }
        }
    }
    for (Generator generator : getGenerators()) {
        for (TypeElement e : typeElements) {
            try {
                generator.generate(e, context);
            } catch (GenerationException ge) {
                ...
            }
        }
    }

    ...

    return true;
}
```

MuleSoft™

MODULE ANNOTATION PROCESSOR

ModuleAnnotationProcessor

process(Set<TypeElement> elements)

```
@Override
public boolean process(Set<? extends TypeElement> annotations, RoundEnvironment env) {
    ...

    for (Validator validator : getValidators()) {
        for (TypeElement e : typeElements) {
            try {
                if (validator.shouldValidate(e, context)) {
                    validator.validate(e, context);
                }
            } catch (ValidationException tve) {
                ...
            }
        }
    }
    for (Generator generator : getGenerators()) {
        for (TypeElement e : typeElements) {
            try {
                generator.generate(e, context);
            } catch (GenerationException ge) {
                ...
            }
        }
    }

    ...

    return true;
}
```

MuleSoft™

MODULE ANNOTATION PROCESSOR

ModuleAnnotationProcessor

process(Set<TypeElement> elements)

```
@Override
public boolean process(Set<? extends TypeElement> annotations, RoundEnvironment env) {
    ...

    for (Validator validator : getValidators()) {
        for (TypeElement e : typeElements) {
            try {
                if (validator.shouldValidate(e, context)) {
                    validator.validate(e, context);
                }
            } catch (ValidationException tve) {
                ...
            }
        }
    }
    for (Generator generator : getGenerators()) {
        for (TypeElement e : typeElements) {
            try {
                generator.generate(e, context);
            } catch (GenerationException ge) {
                ...
            }
        }
    }

    ...

    return true;
}
```

MuleSoft™

Q&A

UPCOMING FEATURES
STUDIO INTEGRATION