什么是Camel

Camal 是开源企业集成框架。

1. spring xml config DSL
2. camel 支持多种协议. eg: http activeMQ JMS MINA CXF
3. tool box 支持200个左右的插件. eg: jolt json 到 json的协议转换插件

1.Camel 的几个要素

1) Endpoint

Endpoints are usually referred to in the DSL via their URIs.
schema:context:path?options. eg: file:inbox/orders?delete=true. timer:myTimer?period=2000

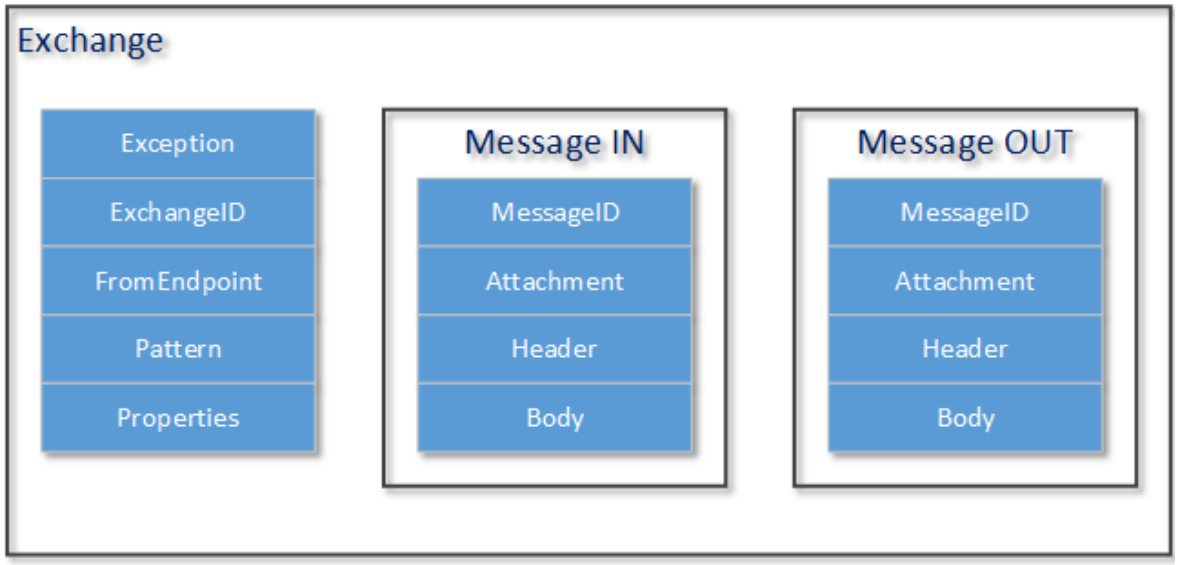
- consumers (from): 从外部资源创建的exchange object
- productor (to): 发送当前消息到外部接口endpoint

```
 */
public interface Component extends CamelContextAware {

    /**
     * Attempt to resolve an endpoint for the given URI if the co
     * capable of handling the URI.
     * <p/>
     * See {@link #useRawUri()} for controlling whether the passe
     * should be as-is (raw), or encoded (default).
     *
     * @param uri the URI to create; either raw or encoded (defau
     * @return a newly created {@link Endpoint} or null if this c
     *         {@link Endpoint} instances using the given uri
     * @throws Exception is thrown if error creating the endpoint
     * @see #useRawUri()
     */
    Endpoint createEndpoint(String uri) throws Exception;
```

2) Exchange

An Exchange is the message container holding the information during the entire routing of a Message received by a Consumer



Exchange:

- ExchaneId: 唯一编号信息
- FromEndpoint: Endpoint fromEndpoint; endpoint的实例对象
- Pattern :

```
public enum ExchangePattern {
    InOnly, RobustInOnly, InOut, InOptionalOut,
    OutOnly, RobustOutOnly, OutIn, OutOptionalIn;
    // TODO: We should deprecate and
     only support InOnly, InOut, and InOptionalOut
```

Camel 只支持:

InOnly (fire & forget)
InOut (request & response)
InOptionlOut

- properties : currentHashMap Exchang实例持有的属性信息
- exceptions: Exchange中的异常信息

Message:

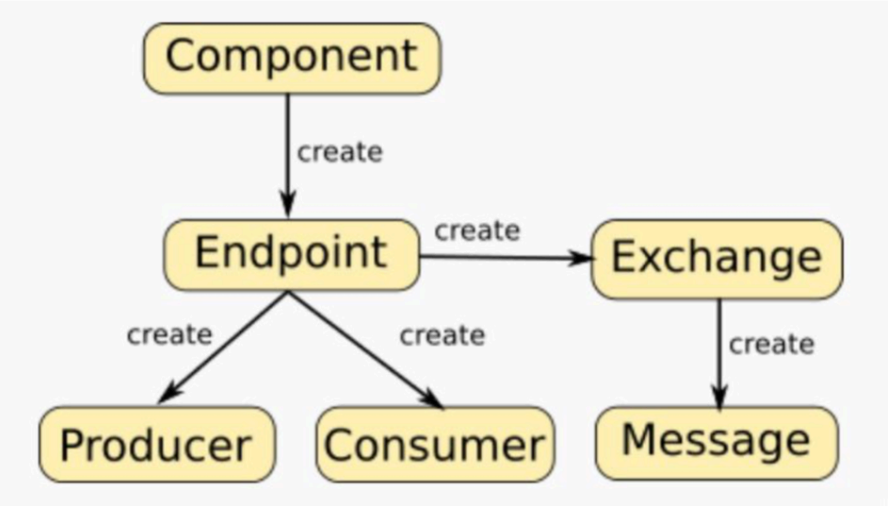
- messageId: 唯一消息ID
- header: private Map<String, Object> headers; like http headers;
- body: private Object body; 消息体
- attachments: private Map<String, DataHandler> attachments;
- fault: private boolean fault; 用来标记是异常信息还是正常信息

3) Processor

接口方法只有一个process 方法，主要用来做 Message Translate

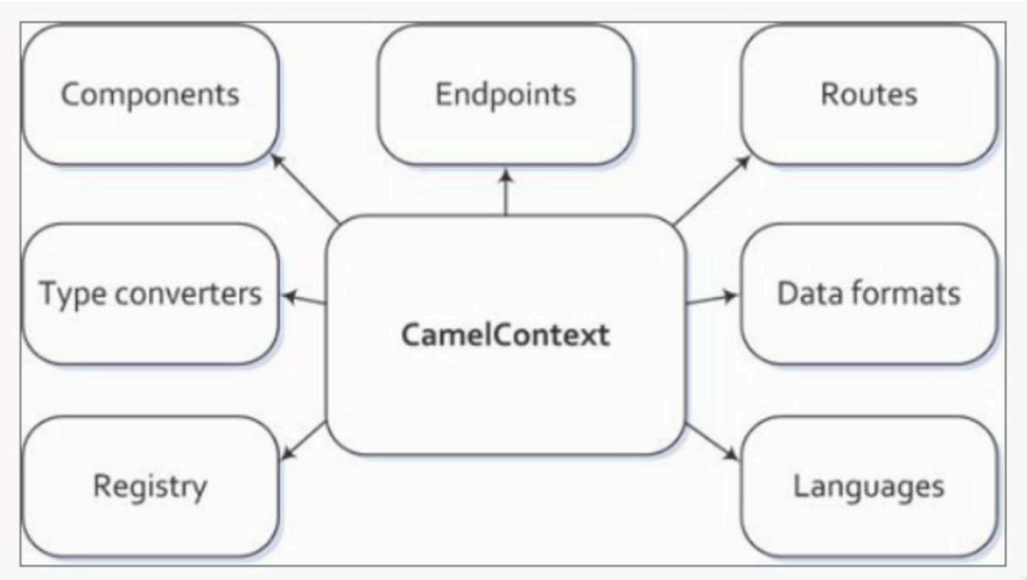
void process(Exchange exchange) throws Exception;

4) Component


Component
create
Endpoint
create
Exchange
create
create
create
Producer
Consumer
Message

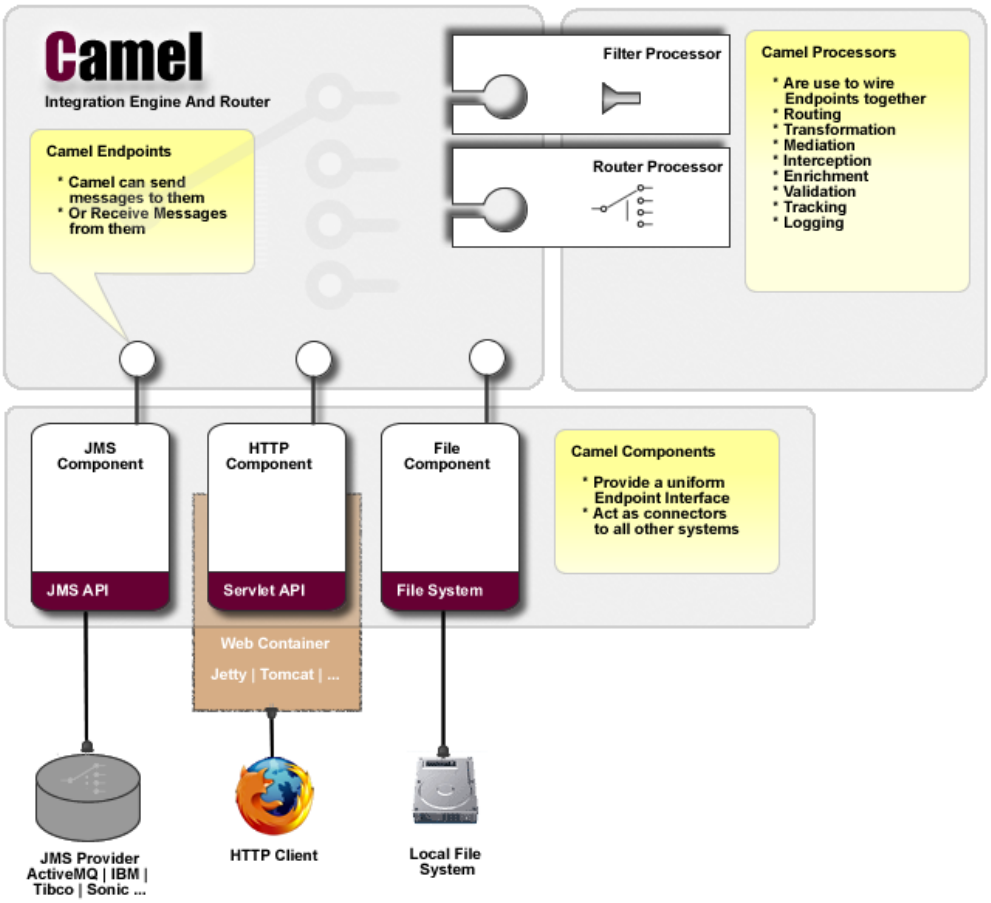
5) CamelContext

configure routes and the policies to use during message exchanges between endpoints.



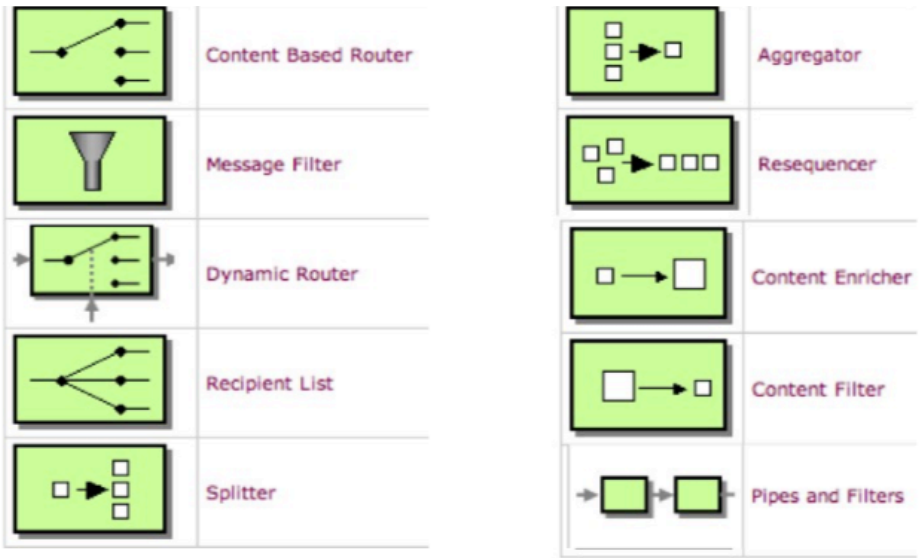
TYPE TO ENTER A CAPTION.

2.Camel 架构

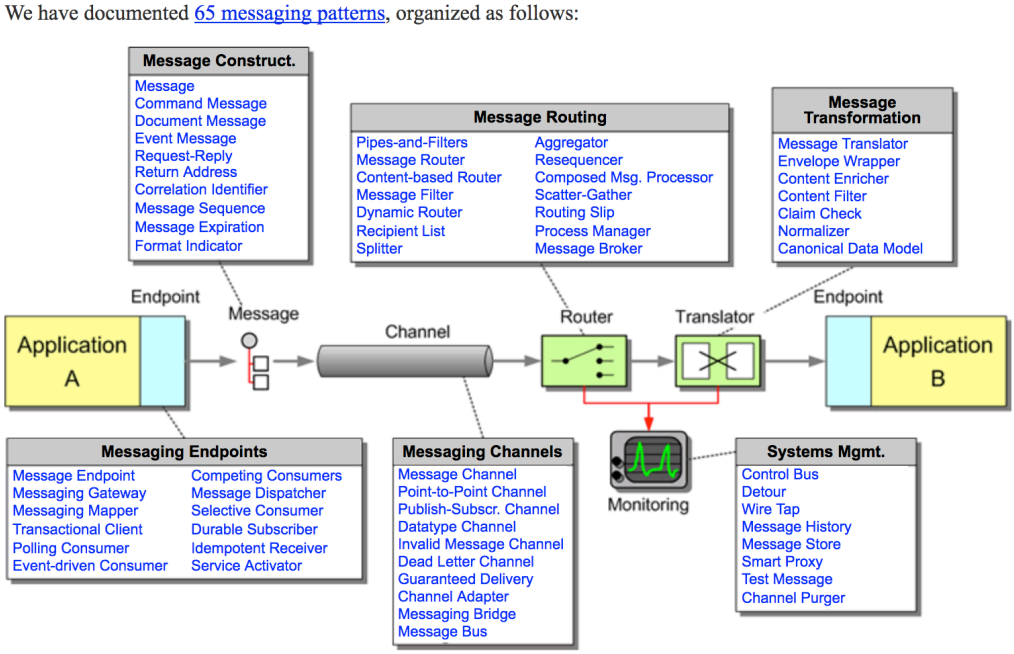


TYPE TO ENTER A CAPTION.

3. 企业集成模式



TYPE TO ENTER A CAPTION.



MESSAGE PATTERNS

link: https://www.enterpriseintegrationpatterns.com/patterns/messaging/

1) content Based Router

```
1  public class ContentBasedRouter extends BouteBuilder{
2    public void configure() throws Exception{
3    from("activemq:queue:newOrder")
4    .choice()
5    .when(xpath("/order/product = 'A'")
6    .to("activemq:queue:A")
7    .otherwise()
8    .to("activemq:queue:B")
9    .end();
10   }
11 }
```

CONTEND BASED

2)….

4. jolt rules

说明	原始JSON	转换规则	目标JSON

简单属性转换	{
	"name": "haha",
	"age": 1,
	"code": "200"
	}
		[
		{
		"operation": "shift",
		"spec": {
		"name": "mapping-name",
		"age": "age",
		“code”:”code"
		}
		}
		]
			{
			"age": 1,
			"code": "200",
			"mapping-name": "haha"
			}

嵌套字典转换	{
	"name": "haha",
	"age": 1,
	"code": "200"
	}
		[
		{
		"operation": "shift",
		"spec": {
		"name": "name",
		"age": "age",
		"code": "data.code"
		}
		}
		]
			{
			"name": "haha",
			"age": 1,
			"data": {
			"code": "200"
			}
			}

忽略转换的字段	{
	"name": "haha",
	"age": 1,
	"code": "200"
	}
		[
		{
		"operation": "shift",
		"spec": {
		"name": "changed-name",
		"code": "data.code",
		"*": "&"
		}
		}
		]
			{
			"changed-name" : "haha",
			"age" : 1,
			"data" : {
			"code" : "200"
			}
			}

嵌套到平级的转换	{
	"name": "haha",
	"age": 1,
	"data": {
	"code": "200"
	}
	}
		[
		{
		"operation": "shift",
		"spec": {
		"data": { "*": "&" },
		"*": "&"
		}
		}
		]
			{
			"name" : "haha",
			"age" : 1,
			"code" : "200"
			}

删除某个key	{
	"name": "haha",
	"age": 1,
	"data": {
	"code": "200"
	}
	}
		[
		{
		"operation": "remove",
		"spec": {
		"data": ""
		}
		}
		]
			{
			"name" : "haha",
			"age" : 1
			}

说明	原始JSON	转换规则	目标JSON

遍历转换json数组元素,并使用Tree的@第一层中属性是name的元素作为key	{
	"size": "xxl",
	"band": "Prada",
	"colors": [
	{
	"name": "red",
	"code": "#0F0"
	},
	{
	"name": "blue",
	"code": "#FF0"
	}
	]
	}
		[
		{
		"operation": "shift",
		"spec": {
		"colors": {
		"*": {
		"code": "color_codes.[].@(1,name)"
		}
		},
		"*": "&"
		}
		}
		]
			{
			"size" : "xxl",
			"band" : "Prada",
			"color_codes" : [{
			"red" : "#0F0"
			}, {
			"blue" : "#FF0"
			}]
			}

同上无@符号	同上	[
		{
		"operation": "shift",
		"spec": {
		"colors": {
		"*": {
		"code": "color_codes.[]"
		}
		},
		"*": "&"
		}
		}
		]
			{
			"size" : "xxl",
			"band" : "Prada",
			"color_codes" : ["#0F0", "#FF0"]
			}

默认值设置	{
	"size": "xxl",
	"band": "Prada",
	"colors": [
	{
	"name": "red",
	"code": "#0F0"
	},
	{
	"name": "blue",
	"code": "#FF0"
	}
	]
	}
		[
		{
		"operation": "shift",
		"spec": {
		"colors": {
		"*": {
		"code": "color_codes.[].@(1,name)"
		}
		},
		"*": "&"
		}
		},
		{
		"operation": "default",
		"spec": {
		"price": "20000"
		}
		}
		]
			{
			"size" : "xxl",
			"band" : "Prada",
			"color_codes" : [{
			"red" : "#0F0"
			}, {
			"blue" : "#FF0"
			}],
			"price" : "20000"
			}