# EIS 整合介紹

Songlin Li

**2019/11/08**

# 主要內容

- 整合簡介
  *整合要做的工作*
- Web 服務(Web Service)
  *對內與對外的服務*
- 批次處理
  *如何創建*
- Lookup
  *如何使用*

## 整合簡介

- 處理EIS和外部系統交互
  - 功能上交互 -> Web 服務
  - 數據上交互 -> 批處理（文件、FTP）

- 處理和內部核心模塊的交互
  - 包裝外部系統功能
  - 通用功能，如果日誌。

## 整合的主要工作內容

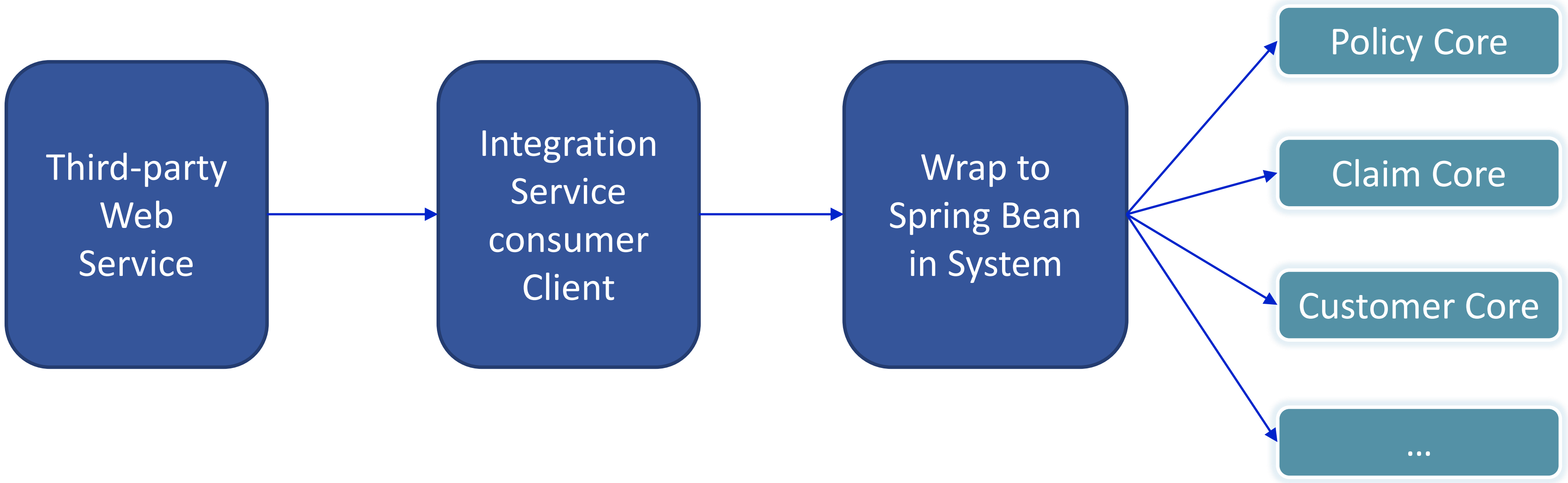
- 開發Web 服務
- 創建批次處理
- 擴展Lookup (不只是整合)

EIS GROUP®

## Web Service

## Web 服務

- 入站服務(inbound service)


Third-party Web Service
Integration Service consumer Client
Wrap to Spring Bean in System
Policy Core
Claim Core
Customer Core
...

## Web 服務

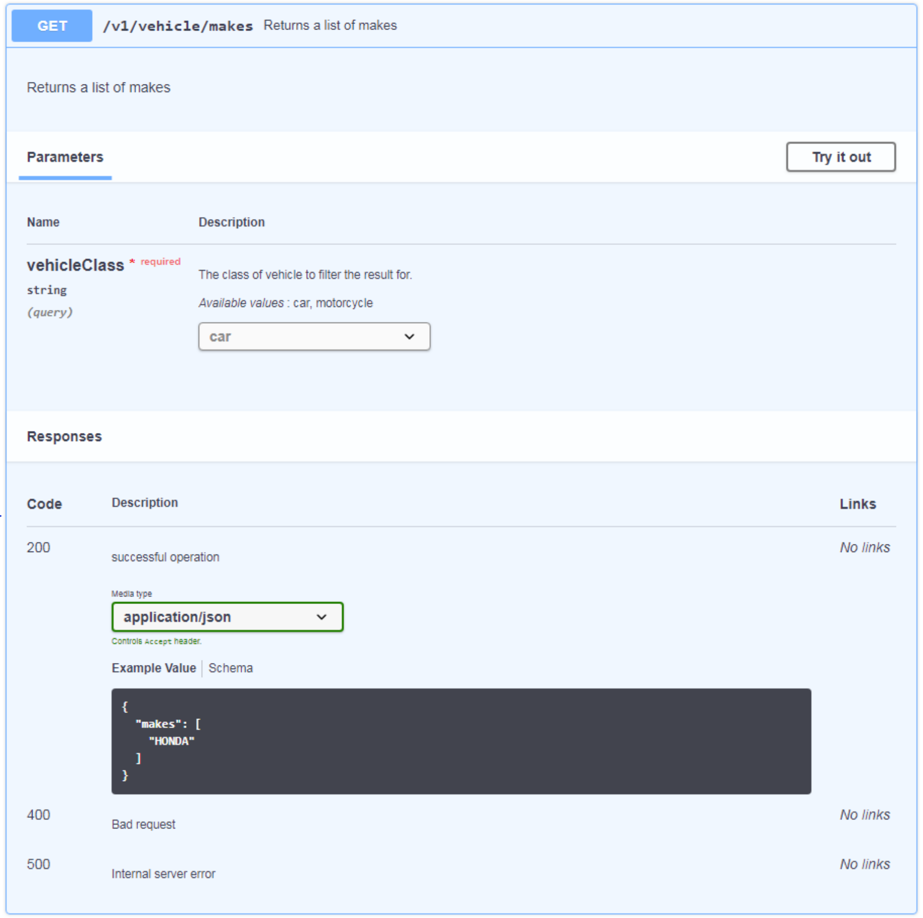
- 入站服務開發示例 (inbound service example)

例如，有外部的RESTful web service:**返回汽車製造廠商的列表**

服務地址 （真實地址VS模擬地址）：
http://production.eqxdev.exigengroup.com:3000/vehicleRegistry/v1/vehicle/makes?vehicleClass=car

http://localhost:3000/vehicleRegistry/v1/vehicle/makes?vehicleClass=car

詳細規格描述：swagger UI



GET `/v1/vehicle/makes` Returns a list of makes

Returns a list of makes

**Parameters** | Try it out

| Name | Description |
|---|---|
| **vehicleClass** * required<br>`string`<br>*(query)* | The class of vehicle to filter the result for.<br>*Available values* : car, motorcycle<br>car |

**Responses**

| Code | Description | Links |
|---|---|---|
| 200 | successful operation | *No links* |
| 400 | Bad request | *No links* |
| 500 | Internal server error | *No links* |

Media type: application/json
Controls `Accept` header.

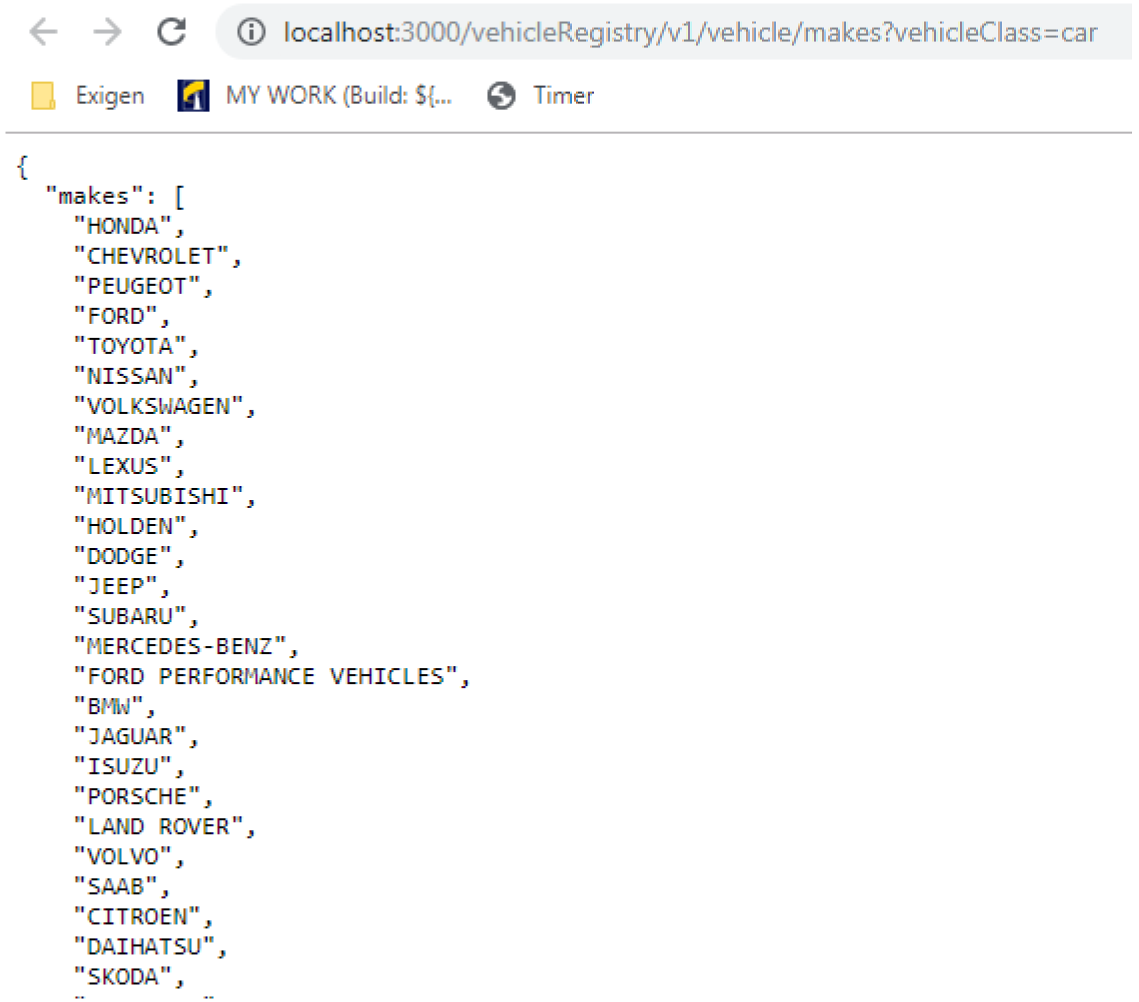
**Example Value** | Schema

```
{
  "makes": [
    "HONDA"
  ]
}
```

# Web 服務

- 入站服務開發示例 (inbound service example)

具體數據的返回：


localhost:3000/vehicleRegistry/v1/vehicle/makes?vehicleClass=car

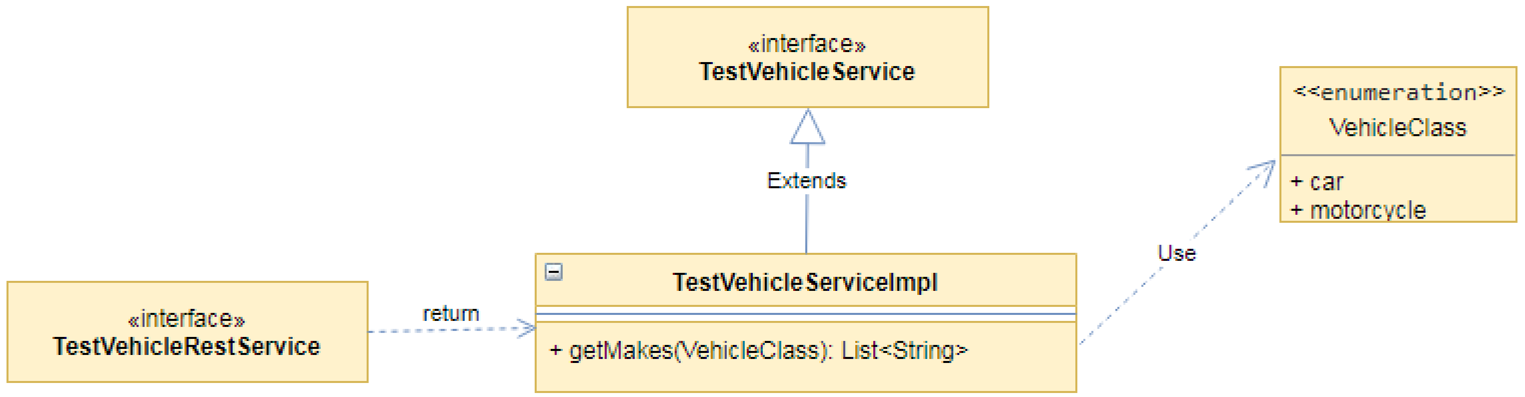
Exigen   MY WORK (Build: ${...   Timer

```
{
  "makes": [
    "HONDA",
    "CHEVROLET",
    "PEUGEOT",
    "FORD",
    "TOYOTA",
    "NISSAN",
    "VOLKSWAGEN",
    "MAZDA",
    "LEXUS",
    "MITSUBISHI",
    "HOLDEN",
    "DODGE",
    "JEEP",
    "SUBARU",
    "MERCEDES-BENZ",
    "FORD PERFORMANCE VEHICLES",
    "BMW",
    "JAGUAR",
    "ISUZU",
    "PORSCHE",
    "LAND ROVER",
    "VOLVO",
    "SAAB",
    "CITROEN",
    "DAIHATSU",
    "SKODA",
```

## Web 服務

- 入站服務開發示例 (inbound service example)

Web服務消費端代碼設計



## Web 服務

- 入站服務開發示例 (inbound service example)

Web服務消費端實現代碼與配置

TestVehicleService

```
/**
 * Get all vehicle makes with vehicle class
 * @param vehicleClass the enum type of vehicle class: car, motorcycle
 * @return list of makes
 * @throws IntegrationException unknow error happens
 * @throws IntegrationNotFoundException can’t find any makes
 */
List<String> getMakes(VehicleClass vehicleClass) throws IntegrationException,IntegrationNotFoundException;
```

TestVehicleServiceImpl

```
@Override
public List<String> getMakes(VehicleClass vehicleClass) throws IntegrationException, IntegrationNotFoundException
{
    Vehicle vehicle = getMakesEndpoint.get(vehicleClass);
    if(vehicle == null || CollectionUtils.isEmpty(vehicle.getMakes())){
        throw new IntegrationNotFoundException("Can not found Vehicle makes");
    }
    return vehicle.getMakes();
}
```

# Web 服務

- 入站服務開發示例 (inbound service example)

Web服務消費端實現代碼與配置

TestVehicleRestService

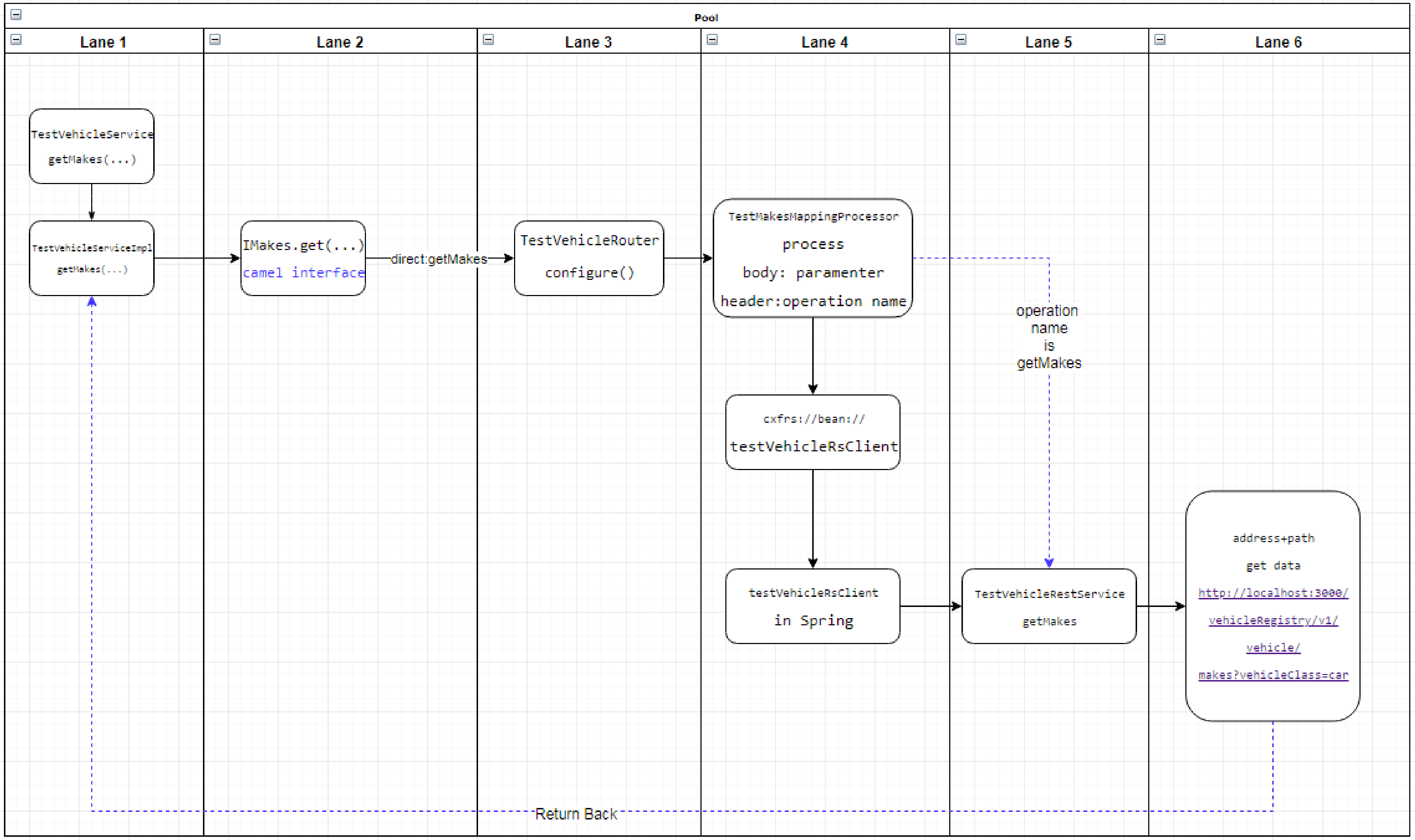
```
@GET
@Path(value = "vehicle/makes")
@Produces({MediaType.APPLICATION_JSON})
Vehicle getMakes(@QueryParam("vehicleClass")VehicleClass vehicleClass)
        throws IntegrationException,IntegrationNotFoundException;
```

Spring configuration XML:

```
<cxf:rsClient id= “testVehicleRsClient" address=" http://localhost:3000/vehicleRegistry/v1/"
              serviceClass="com.exigen.test.interfaces.vehicle.camel.rest.TestVehicleRestService">
    <cxf:providers>
        <ref bean="jsonProvider"/>
    </cxf:providers>
</cxf:rsClient>
```

# Web 服務

- 入站服務開發示例 (inbound service example)

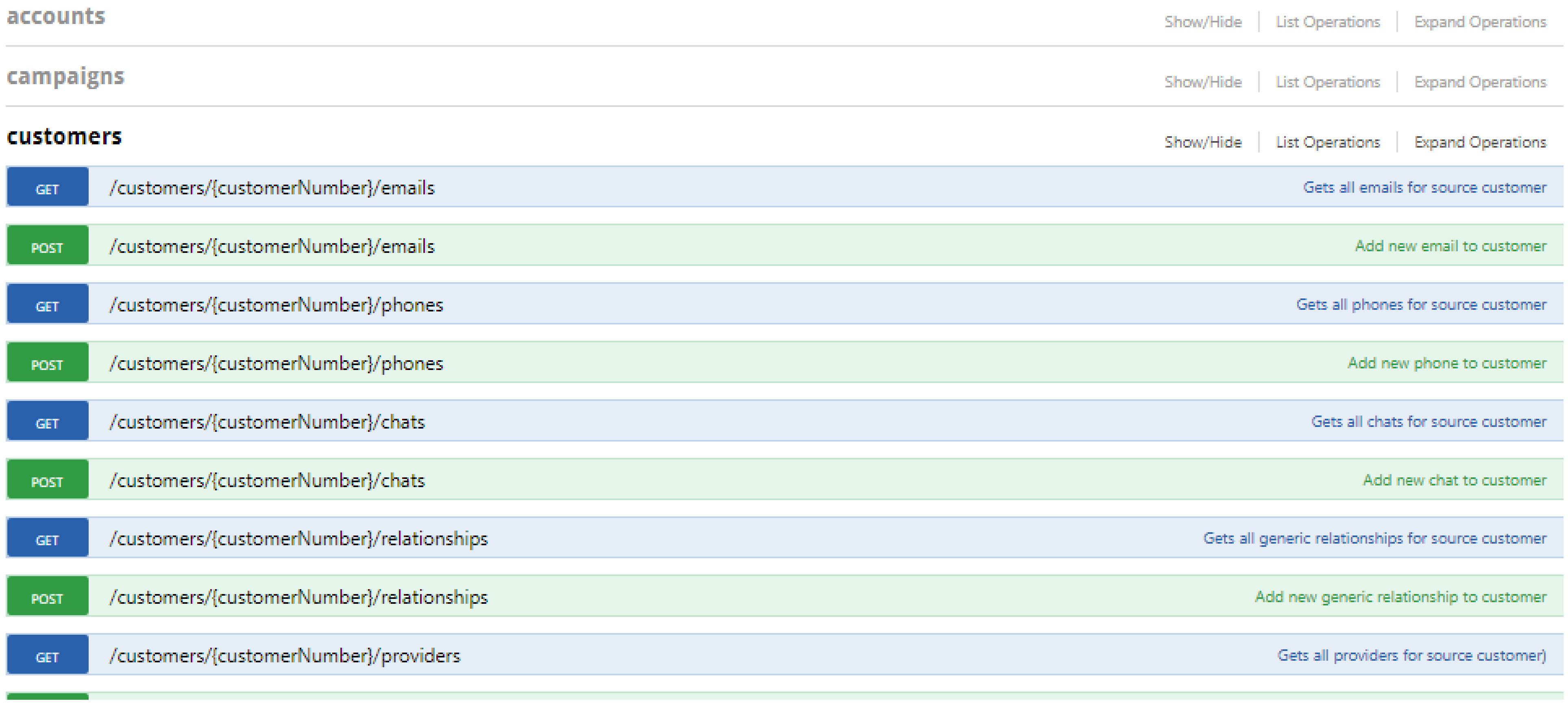


Pool
Lane 1
Lane 2
Lane 3
Lane 4
Lane 5
Lane 6
TestVehicleService getMakes(...)
TestVehicleServiceImpl getMakes(...)
IMakes.get(...) camel interface
direct:getMakes
TestVehicleRouter configure()
TestMakesMappingProcessor process body: paramenter header:operation name
operation name is getMakes
cxfrs://bean:// testVehicleRsClient
testVehicleRsClient in Spring
TestVehicleRestService getMakes
address+path get data http://localhost:3000/ vehicleRegistry/v1/ vehicle/ makes?vehicleClass=car
Return Back

## Web 服務

- 出站服務-已有(Outbound Service – OOTB)

https://dev5-hot-app01.eisgroup.com/hot-app/swagger-ui.xhtml#/



accounts — Show/Hide | List Operations | Expand Operations

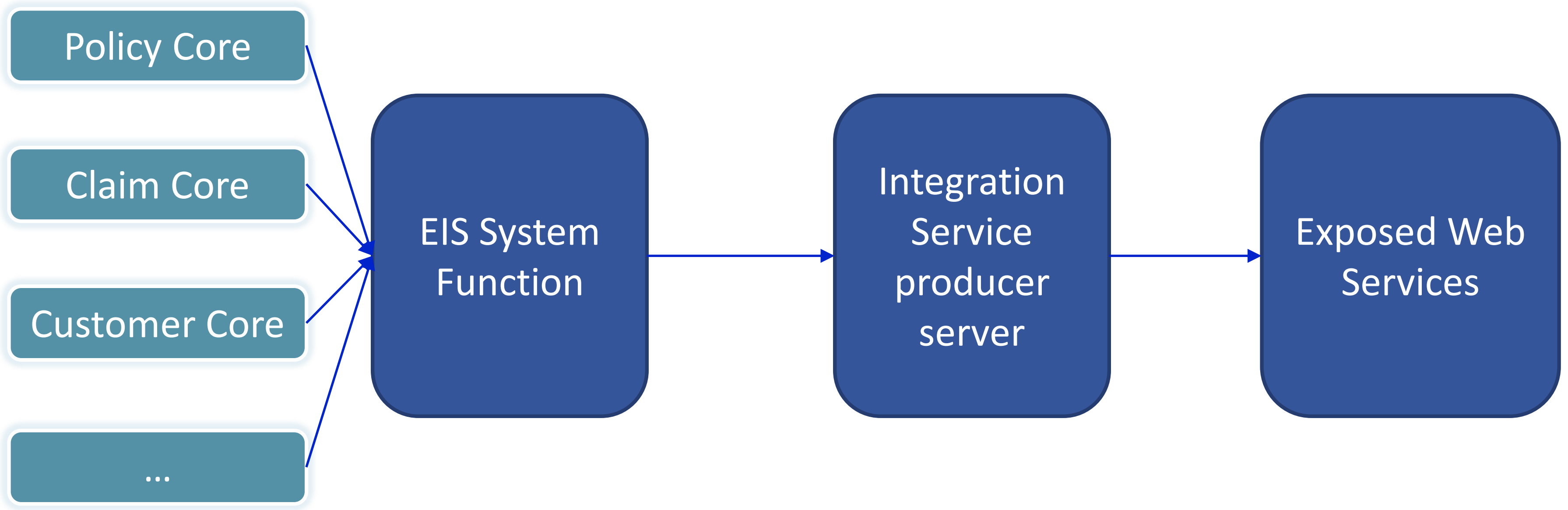
campaigns — Show/Hide | List Operations | Expand Operations

customers — Show/Hide | List Operations | Expand Operations

| | | |
|---|---|---|
| GET | /customers/{customerNumber}/emails | Gets all emails for source customer |
| POST | /customers/{customerNumber}/emails | Add new email to customer |
| GET | /customers/{customerNumber}/phones | Gets all phones for source customer |
| POST | /customers/{customerNumber}/phones | Add new phone to customer |
| GET | /customers/{customerNumber}/chats | Gets all chats for source customer |
| POST | /customers/{customerNumber}/chats | Add new chat to customer |
| GET | /customers/{customerNumber}/relationships | Gets all generic relationships for source customer |
| POST | /customers/{customerNumber}/relationships | Add new generic relationship to customer |
| GET | /customers/{customerNumber}/providers | Gets all providers for source customer) |

## Web 服務

- 出站服務-客制 (Outbound Service – Customized)



## Web 服務

- 出站服務開發示例(Outbound Service – Customized example)

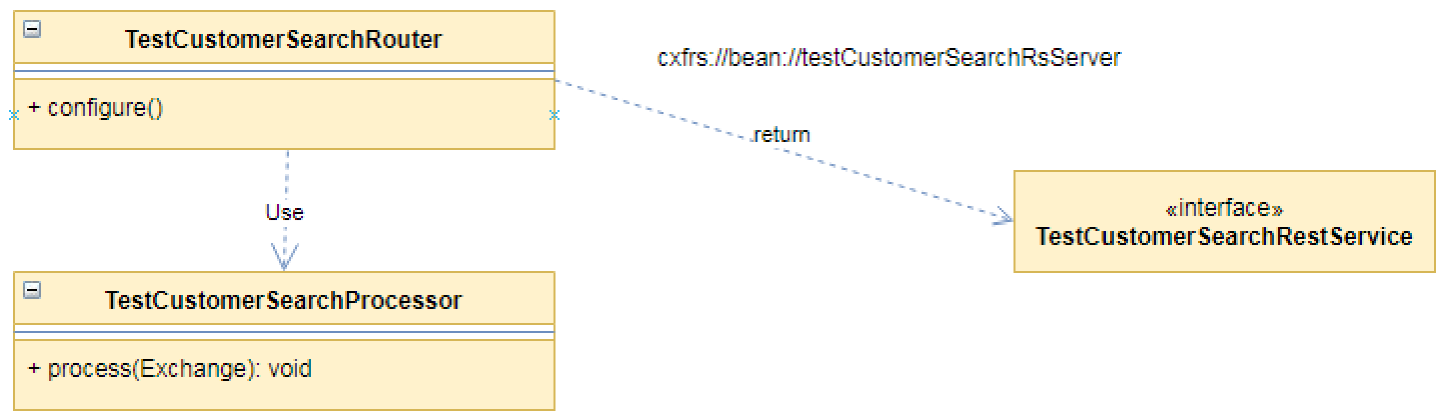
舉例
實現一個**客戶查詢**的服務， 能夠根據以下字段查詢：
1.1 Firstname (必要參數)
1.2 Lastname (必要參數)
1.3 Phone and/or Email

查詢邏輯：返回所有完全匹配Firstname和LastName, 任意匹配phone或者email的客戶列表

## Web 服務

- 出站服務開發示例(Outbound Service – Customized example)

Web服務服務端代碼設計


TestCustomerSearchRouter
+ configure()
Use
TestCustomerSearchProcessor
+ process(Exchange): void
cxfrs://bean://testCustomerSearchRsServer
.return
«interface»
TestCustomerSearchRestService

## Web 服務

- 出站服務開發示例(Outbound Service – Customized example)

TestCustomerSearchRestService:

```
@Path( “/” )
@Api( “/” )
public interface TestCustomerSearchRestService {

    @GET
    @Path( “/customers” )
    @Produces(MediaType.APPLICATION_JSON)
    @ApiOperation(
            value =  “Search Customer by firstName, lastName, phone and email.” ,
            response = CustomerDTO.class,
            responseContainer = "List"
    )
    Response searchCustomer(@ApiParam(value = "First name",example = "First", required = true) @Nonnull @QueryParam("firstName") String firstName,
                            @ApiParam(value = "Last name",example = "Last", required = true) @Nonnull @QueryParam(“lastName") String lastName,
                            @ApiParam(value = "Customer phone number",example = "123456789012") @Nullable @QueryParam(“phone") String phone,
                            @ApiParam(value = "Customer email address",example = "mail@mail.com") @Nullable @QueryParam(“email") String email)
throws Exception;
}
```

## Web 服務

- 出站服務開發示例(Outbound Service – Customized example)

Spring configuration XML:

```
<cxf:rsServer id= “testCustomerSearchRsServer" address="/customerSearch/v1"
              serviceClass="com.exigen.test.interfaces.customer.rest.TestCustomerSearchRestService">
    <cxf:providers>
        <ref bean="jsonProvider"/>
    </cxf:providers>
    <cxf:features>
        <ref bean= “testCustomerSearchRestSwagger" />
    </cxf:features>
</cxf:rsServer>
```

# Web 服務

- 出站服務開發示例(Outbound Service – Customized example)

TestCustomerSearchRouter

```
public class TestCustomerSearchRouter extends TestAbstractRSRouter {
    private static final Logger LOGGER = LoggerFactory.getLogger(TestCustomerSearchRouter.class);
    private static final String ROUTE_ID = "TestCustomerSearchRouter_id";
    private static final String ROUTE_FROM = "cxfrs://bean://xxxCustomerSearchRsServer";

    private Processor xxxCustomerSearchProcessor;

    @Override
    public void configure() {
        // Create router for customer searching
        choiceWithSwagger(from(ROUTE_FROM).routeId(ROUTE_ID));
    }
```

## Web 服務

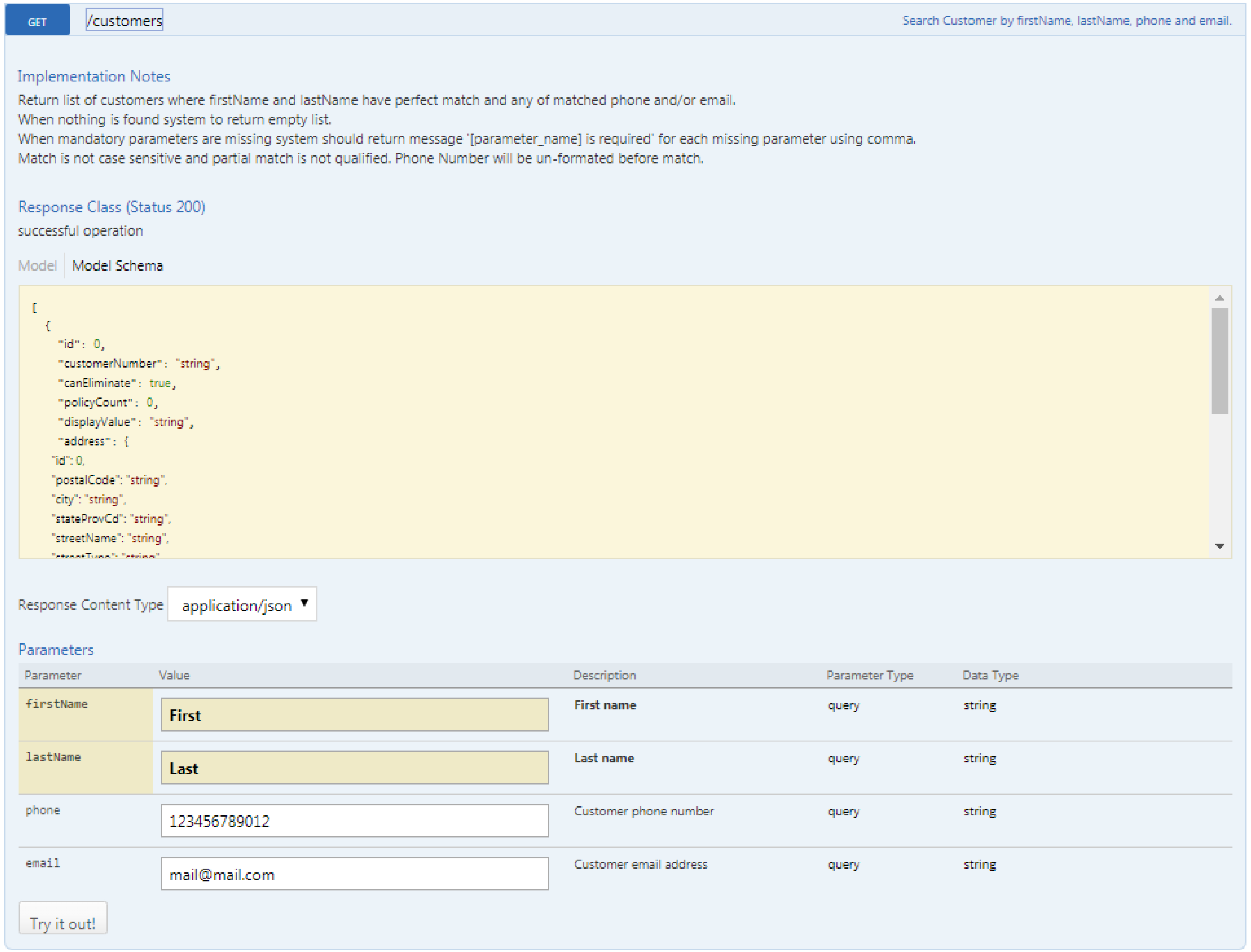
- 出站服務開發示例(Outbound Service – Customized example)

TestCustomerSearchProcessor

```
public class TestCustomerSearchProcessor implements Processor {
    @Override
    public void process(Exchange exchange) throws Exception {
        List allCustomers = customerService.findCustomers(firstName, lastName);
        List<CustomerDTO> dtos = matchCustomer(allCustomers, phone, email);
        response = Response.ok().entity(dtos).build();

    }
}
```

## Web 服務

- 出站服務開發示例(Outbound Service – Customized example)



GET /customers — Search Customer by firstName, lastName, phone and email.

Implementation Notes

Return list of customers where firstName and lastName have perfect match and any of matched phone and/or email.
When nothing is found system to return empty list.
When mandatory parameters are missing system should return message '[parameter_name] is required' for each missing parameter using comma.
Match is not case sensitive and partial match is not qualified. Phone Number will be un-formated before match.

Response Class (Status 200)

successful operation

Model | Model Schema

```
[
  {
    "id": 0,
    "customerNumber": "string",
    "canEliminate": true,
    "policyCount": 0,
    "displayValue": "string",
    "address": {
   "id": 0,
   "postalCode": "string",
   "city": "string",
   "stateProvCd": "string",
   "streetName": "string",
```

Response Content Type: application/json

Parameters

| Parameter | Value | Description | Parameter Type | Data Type |
|---|---|---|---|---|
| firstName | First | First name | query | string |
| lastName | Last | Last name | query | string |
| phone | 123456789012 | Customer phone number | query | string |
| email | mail@mail.com | Customer email address | query | string |

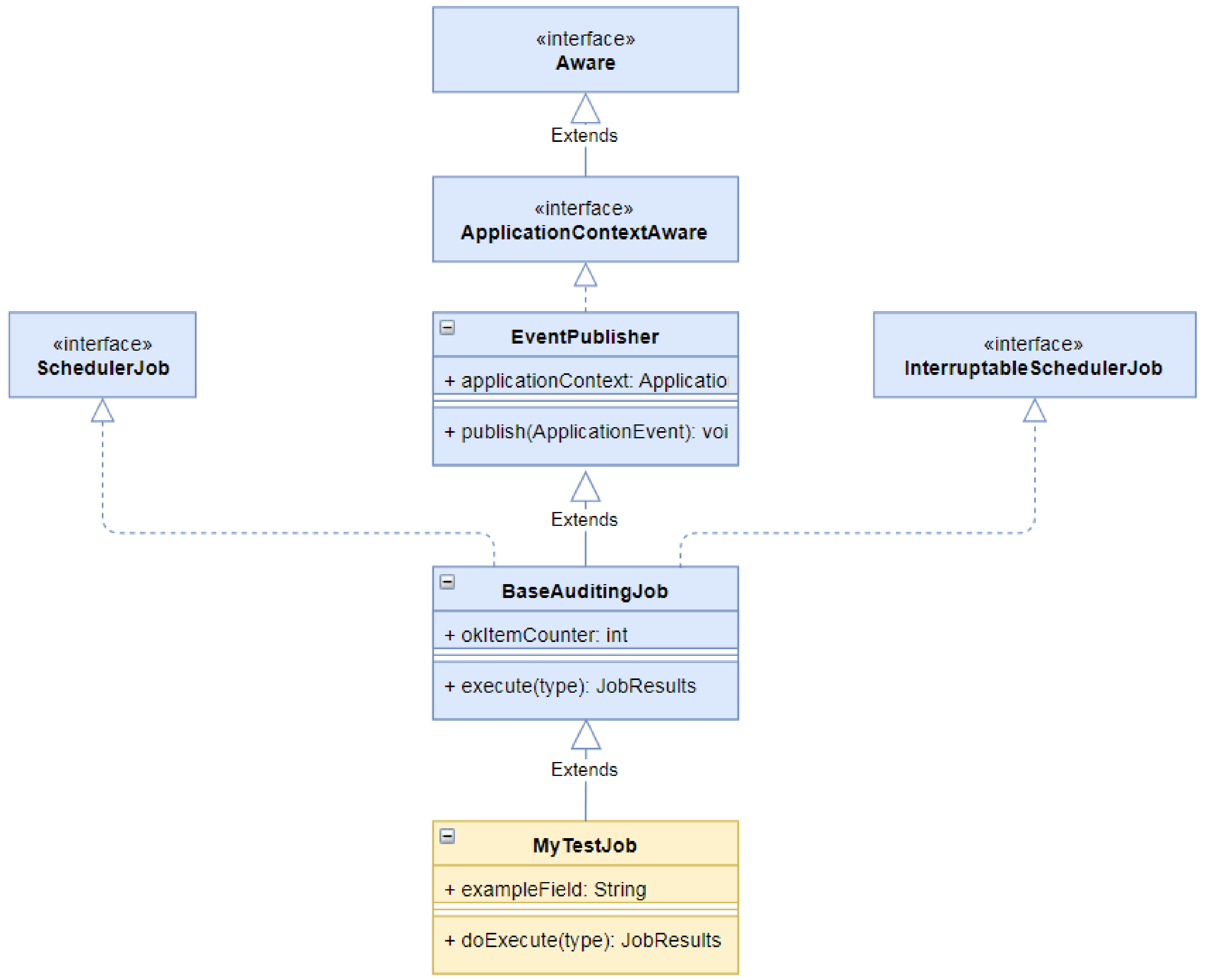
Try it out!

## 批次處理

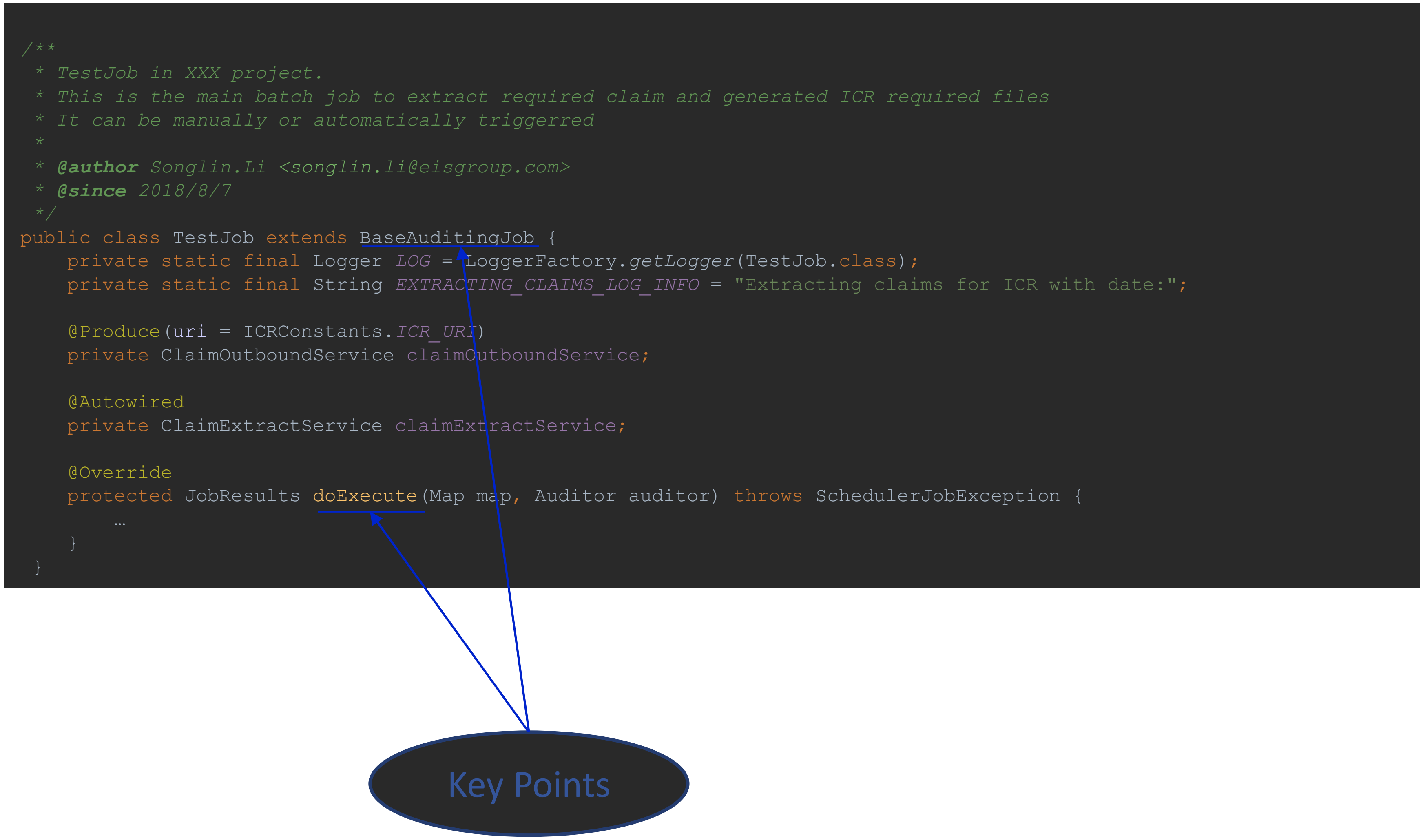
## 批次處理主要功能

- 通過文件交互數據
- 通過數據庫中間表交互數據

## 批次處理類圖


«interface»
Aware
Extends
«interface»
ApplicationContextAware
«interface»
SchedulerJob
EventPublisher
+ applicationContext: Applicatio
+ publish(ApplicationEvent): voi
«interface»
InterruptableSchedulerJob
Extends
BaseAuditingJob
+ okItemCounter: int
+ execute(type): JobResults
Extends
MyTestJob
+ exampleField: String
+ doExecute(type): JobResults

# 批次處理開發範例

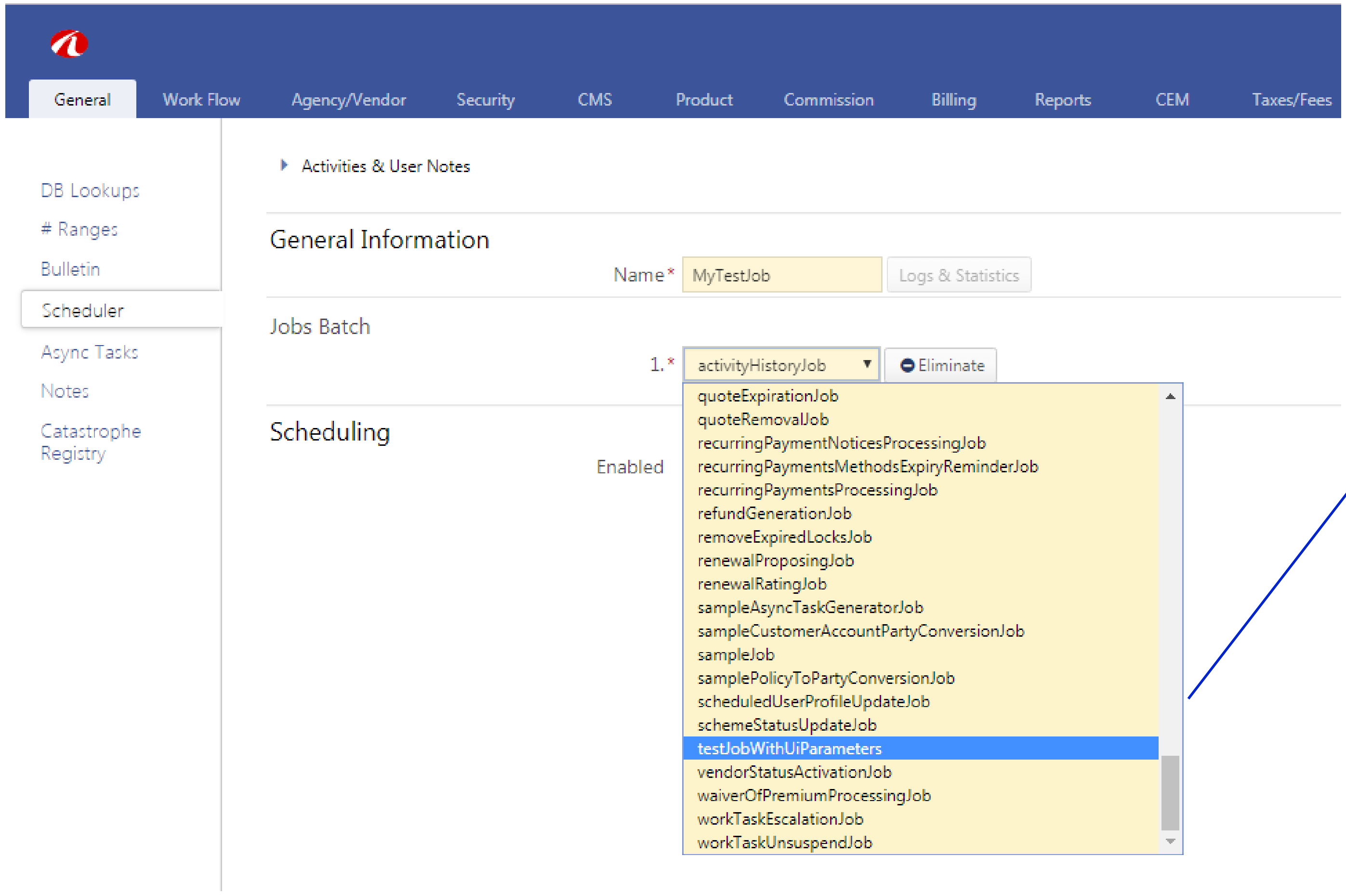


```java
/**
 * TestJob in XXX project.
 * This is the main batch job to extract required claim and generated ICR required files
 * It can be manually or automatically triggerred
 *
 * @author Songlin.Li <songlin.li@eisgroup.com>
 * @since 2018/8/7
 */
public class TestJob extends BaseAuditingJob {
    private static final Logger LOG = LoggerFactory.getLogger(TestJob.class);
    private static final String EXTRACTING_CLAIMS_LOG_INFO = "Extracting claims for ICR with date:";

    @Produce(uri = ICRConstants.ICR_URI)
    private ClaimOutboundService claimOutboundService;

    @Autowired
    private ClaimExtractService claimExtractService;

    @Override
    protected JobResults doExecute(Map map, Auditor auditor) throws SchedulerJobException {
        …
    }
}
```

# 批次處理開發範例

在spring的配置文件中配置創建的批處理bean實例

```
<bean id= “testJobWithUiParameters” class="com.exigen.test.interfaces.icr.job.TestJob" />
```



General Information

Name* MyTestJob

Jobs Batch

1.* activityHistoryJob

quoteExpirationJob
quoteRemovalJob
recurringPaymentNoticesProcessingJob
recurringPaymentsMethodsExpiryReminderJob
recurringPaymentsProcessingJob
refundGenerationJob
removeExpiredLocksJob
renewalProposingJob
renewalRatingJob
sampleAsyncTaskGeneratorJob
sampleCustomerAccountPartyConversionJob
sampleJob
samplePolicyToPartyConversionJob
scheduledUserProfileUpdateJob
schemeStatusUpdateJob
testJobWithUiParameters
vendorStatusActivationJob
waiverOfPremiumProcessingJob
workTaskEscalationJob
workTaskUnsuspendJob

Scheduling

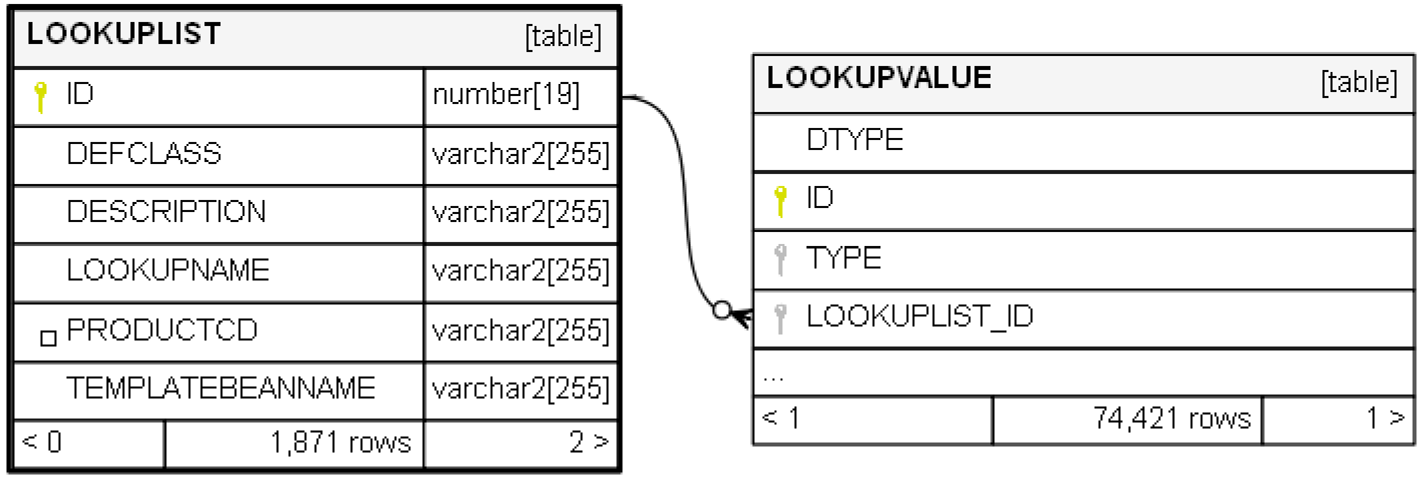
```
com.exigen.scheduler.bean.manger.JobManager

public String[] getAvailableJobNames() {
   return applicationContext.getBeanNamesForType(SchedulerJob.class);
}
```

## Lookups

## Lookups

Lookup可以想像成一種數據結構關係， 每個代碼都有對應的值。

Lookup主要操作兩個數據表格LookupList 和 LookupValue.



| LOOKUPLIST | [table] |
|---|---|
| ID | number[19] |
| DEFCLASS | varchar2[255] |
| DESCRIPTION | varchar2[255] |
| LOOKUPNAME | varchar2[255] |
| PRODUCTCD | varchar2[255] |
| TEMPLATEBEANNAME | varchar2[255] |
| < 0   1,871 rows | 2 > |

| LOOKUPVALUE | [table] |
|---|---|
| DTYPE | |
| ID | |
| TYPE | |
| LOOKUPLIST_ID | |
| ... | |
| < 1   74,421 rows | 1 > |

## Lookups

### LookupList:

```
@Entity
public class LookupList extends BaseEntity implements Serializable{

   private String lookupName;
   private String description;


   private String defClass;
   private String productCd;

   @OneToMany(cascade=CascadeType.ALL)
   @JoinColumn(name="LOOKUPLIST_ID")
   private List<CodeValueLookup> codeValueLookups;
}
```

### LookupValue:

```
@Entity
public class LookupValue extends BaseEntity{

   private String code;

   public String getCode() {
      return code;
   }
   public void setCode(String code) {
      this.code = code;
   }
}
```

## Lookups

創建新的Lookup (Liquibase XML)

1.創建LookupList

```
<changeSet  id="01" author="sli" dbms="mssql">
    <preConditions onFail="MARK_RAN" onError="CONTINUE">
        <sqlCheck expectedResult="0">
            SELECT COUNT(*) FROM LookupList where lookupName = 'InboundPaymentFilter'
        </sqlCheck>
    </preConditions>
    <comment>Feature-001: add inbound payment filter actions</comment>
    <sql>
        INSERT INTO LookupList (defClass, lookupName) VALUES ('com.exigen.interfaces.domain.TestInboundPaymentFilterLookupValue',
'InboundPaymentFilter');
    </sql>
</changeSet>
```

## Lookups

創建新的Lookup

2.創建LookupValue數據

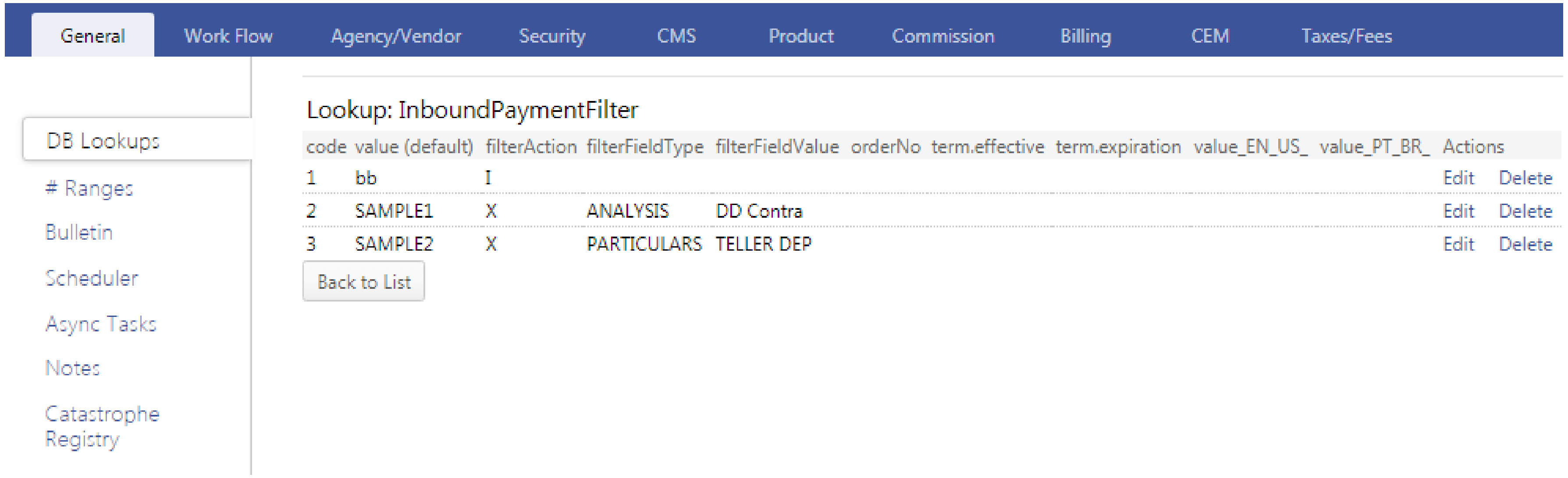
```
<changeSet id="02" author="sli" dbms="mssql" runOnChange="true">
    <comment>Feature-001 : add inbound payment filter actions</comment>
    <sql>
        ALTER TABLE LookupValue ${constraint.disable.keyword} CONSTRAINT FK_LookupValueLookupListID;
    </sql>
    <sql>
        DELETE FROM LookupValue WHERE LOOKUPLIST_ID = (SELECT id FROM LookupList WHERE lookupName = 'InboundPaymentFilter');
    </sql>
    <loadData tableName="LookupValue" file="csv/interfaces/InboundPaymentFilter_lookup.csv"/>
    <sql>
        UPDATE LookupValue
            SET LOOKUPLIST_ID = (SELECT id FROM LookupList WHERE lookupName = 'InboundPaymentFilter')
        WHERE LOOKUPLIST_ID IS NULL
    </sql>
    <sql>
        ALTER TABLE LookupValue ${constraint.enable.keyword} CONSTRAINT FK_LookupValueLookupListID
    </sql>
</changeSet>
```

csv/interfaces/InboundPaymentFilter_lookup.csv

```
DTYPE,code,displayValue,filterAction,filterFieldType,filterFieldValue,effective,expiration
TestInboundPaymentFilterLookupValue,1,bb,I,,,NULL,NULL
TestInboundPaymentFilterLookupValue,2,SAMPLE1,X,ANALYSIS,DD Contra,NULL,NULL
TestInboundPaymentFilterLookupValue,3,SAMPLE2,X,PARTICULARS,TELLER DEP,NULL,NULL
```

## Lookups

創建新的Lookup


General | Work Flow | Agency/Vendor | Security | CMS | Product | Commission | Billing | CEM | Taxes/Fees

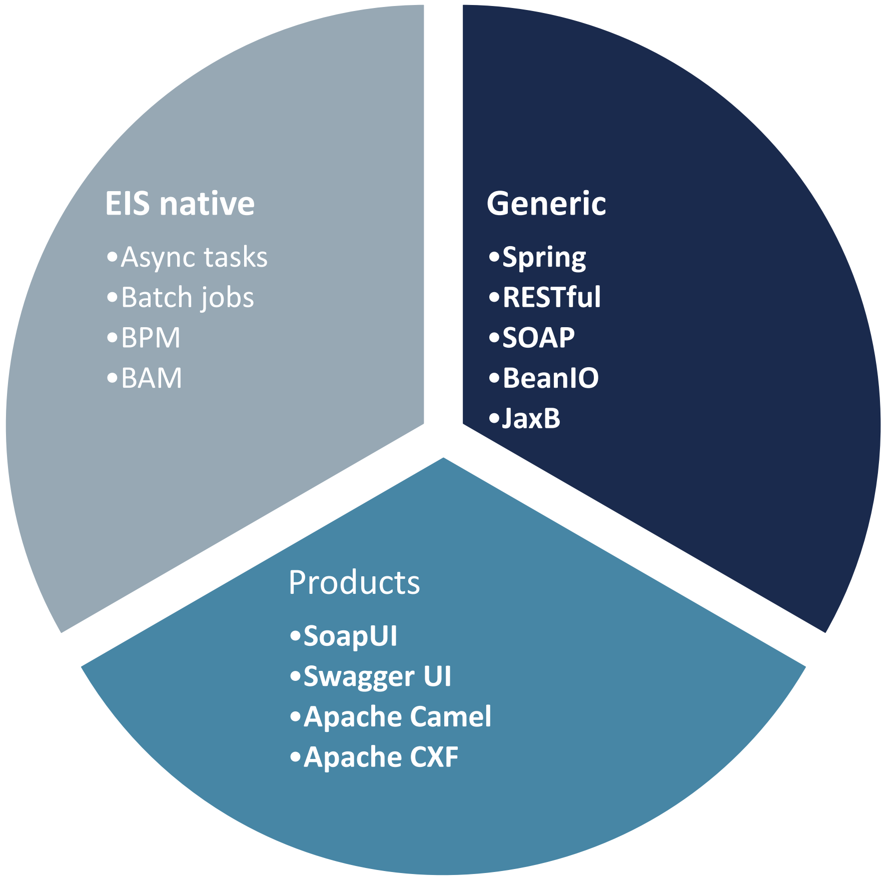
DB Lookups
# Ranges
Bulletin
Scheduler
Async Tasks
Notes
Catastrophe Registry

Lookup: InboundPaymentFilter

| code | value (default) | filterAction | filterFieldType | filterFieldValue | orderNo | term.effective | term.expiration | value_EN_US_ | value_PT_BR_ | Actions | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | bb | I | | | | | | | | Edit | Delete |
| 2 | SAMPLE1 | X | ANALYSIS | DD Contra | | | | | | Edit | Delete |
| 3 | SAMPLE2 | X | PARTICULARS | TELLER DEP | | | | | | Edit | Delete |

Back to List

# 推薦學習範圍



**EIS native**

- Async tasks
- Batch jobs
- BPM
- BAM

**Generic**

- **Spring**
- **RESTful**
- **SOAP**
- **BeanIO**
- **JaxB**

Products

- **SoapUI**
- **Swagger UI**
- **Apache Camel**
- **Apache CXF**

**Java書箱推薦：**
<<Thinking in Java>>
<<Spring in Action>>

## Thank You