## Componente Java para configuración

Dirección General de Tecnologías de la Información y Comunicaciones

Áreas de Gobierno Tecnológico de SSII y del Dato

Servicio Andaluz de Salud
Consejería de Sanidad, Presidencia y Emergencias

Contenido



Resumen


- **Versión:** v01r13
- **Fecha publicación:** 22 feb 2019
- **Entrada en vigor desde:** 22 feb 2019


blocked URL

## *Cumplimiento normativo*

Las normas expuestas son de obligado cumplimiento. La STIC podrá estudiar los casos excepcionales los cuales serán gestionados a través de los responsables del proyecto correspondiente y autorizados por el Área de Gobernanza de la STIC. Asimismo cualquier aspecto no recogido en estas normas deberá regirse en primera instancia por las guías técnicas correspondientes al esquema nacional de seguridad y esquema nacional de interoperabilidad según correspondencia y en su defecto a los marcos normativos y de desarrollo software establecidos por la Junta de Andalucía, debiendo ser puesto de manifiesto ante la STIC.

La STIC se reserva el derecho a la modificación de la norma sin previo aviso, tras lo cual, notificará del cambio a los actores implicados para su adopción inmediata según la planificación de cada proyecto.

En el caso de que algún actor considere conveniente y/o necesario el incumplimiento de alguna de las normas y /o recomendaciones, deberá aportar previamente la correspondiente justificación fehaciente documentada de la solución alternativa propuesta, así como toda aquella documentación que le sea requerida por la STIC para proceder a su validación técnica.

**Contacto Arquitectura:** l-arquitectura.stic@juntadeandalucia.es

## *Histórico de cambios*

Los cambios en la normativa vendrán acompañados de un registro de las modificaciones. De este modo se podrá realizar un seguimiento y consultar su evolución. Ordenándose de mas recientes a menos recientes, prestando especial cuidado a las cabezeras de la tablas dónde se indican las fechas de entrada en vigor y versión.

| **Versión** | v01r13 | **Fecha publicación** | 22 feb 2019 | **Fecha entrada en vigor** | 22 feb 2019 |
|---|---|---|---|---|---|
| **Alcance** | | | | | |
| • Documentación para la integración con la api de gestión de configuraciones de los aplicativos | | | | | |

## *Datos de la libreria*:

| | |
|---|---|
| **JDK** | 1.6 |
| **CDI** | 1.0/1.1 (opcional) |
| **Repositorio** | http://git.sas.junta-andalucia.es/componentes_comunes/Java/sasconfiguration |
| **Dependencia Maven** | Con CDI:<br>`<dependency> <groupId>es.ja.csalud.sas.componentescomunes.sasconfiguration</groupId> <artifactId>sasconfiguration-cdi</artifactId> <version>x.x.x</version> <scope>runtime</version> < /dependency>`<br><br>Sin CDI:<br>`<dependency> <groupId>es.ja.csalud.sas.componentescomunes.sasconfiguration</groupId> <artifactId>sasconfiguration-core</artifactId> <version>x.x.x</version> <scope>runtime</version> < /dependency>`<br><br>Para usar las anotaciones:<br>`<dependency> <groupId>es.ja.csalud.sas.componentescomunes.sasconfiguration</groupId> <artifactId>sasconfiguration-annotation</artifactId> <version>x.x.x</version> </dependency>`<br><br>Api de los servicios:<br>`<dependency> <groupId>es.ja.csalud.sas.componentescomunes.sasconfiguration</groupId> <artifactId>sasconfiguration-api</artifactId> <version>x.x.x</version> </dependency>` |

# 1. Introducción

Esta librería permitirá obtener configuración de diferentes orígenes y hacer uso de ellos de una forma simplificada en los aplicativos.

Puede encontrarse un ejemplo de integración de esta librería en el repositorio Git de esta librería.

## 2. Funcionalidades

Actualmente la librería tiene las siguiente funcionalidades:

- Consultas parametricas
    - Mediante el servicio o anotaciones
    - Origenes:
        - Fichero (json/yaml/properties)
        - Base de datos (JNDI)
        - Git (json/yaml/properties)(Beta)
    - Tipos de consulta
        - Consulta por Key
            - Puede devolver un objeto complejo mediante transformaciones
        - Consulta por prefijo.
            - Retorna un Mapa de strings
            - Tambien puede retornar un objeto complejo si existe la transformacion concreta
- Establecer parametricas
    - Permite establecer parametricas en cualquiera de los datasources, aunque solo algunos permitirian persistir estas parametricas en la fuente de datos.
- Eventos de cambios en las parametricas (Beta)
    - Listeners que reaccionan a los cambios de una parametrica
- Trasformaciones
    - Transformaciones automaticas de cadena de texto a objeto java y viceversa mediante la implementacion de una interfaz de transformacion
    - Tambien se permite la transformacion de un mapa de strings a objeto

De cara a priorizar futuras funcionalidades, si algún proveedor tiene la necesidad de alguna funcionalidad adicional se deberá de comunicar a l-arquitectura.stic.sspa@juntadeandalucia.es

## 3- Dependencias

--

## 4- Historial de Cambios

| sasconfiguration-core/cdi | sasconfiguration-api | sasconfiguration-annotation | sasconfiguration-ds-git |
|---|---|---|---|
| v2.2.1<br>• Corregido bug el jndiDataSource<br>v2.2.0<br>• Nuevo datasource Enviroment<br>v2.1.0<br>• Adaptacion para nuevo datasource (git)<br>• Transformaciones de mapa a objeto<br>v2.0.0<br>• Compatibilidad con JDK 6 o superior<br>v1.1.0.1<br>• Añadida opcion de obtener las propiedades mediante anotaciones<br>v1.0.0.1<br>• Primera version del la libreria | v1.2.0<br>• Expuestas clases abstractas<br>v1.1.0<br>• Expuestas todas las interfaces<br>v1.0.0.1<br>• Primera version del la libreria | v1.1.0<br>• Cambio de nombre de paquetes<br>v1.0.0.1<br>• Primera version del la libreria | v1.1.1<br>• Soporte diferentes tipos de ficheros<br>v1.0.0<br>• Primera version del la implementacion del datasource de Git |

## 5. Integración de la librería

Para incluir la librería en un proyecto Maven es necesario realizar los siguientes pasos:

1. Agregar el repositorio del SAS: Repositorio de artefactos
2. Especificar la dependencia en el POM del proyecto que requiere la funcionalidad.

   -Si es un proyecto con cdi:

```
...
<dependencies>
    <!--CONFIGURACION-->
    <dependency>
        <groupId>es.ja.csalud.sas.componentescomunes.sasconfiguration<
/groupId>
        <artifactId>sasconfiguration-cdi</artifactId>
        <version>x.y.z</version>
    </dependency>
...
```

   -Si es un proyecto SIN cdi:

```
   ...
<dependencies>
    <!--CONFIGURACION-->
    <dependency>
        <groupId>es.ja.csalud.sas.componentescomunes.sasconfiguration<
/groupId>
        <artifactId>sasconfiguration-core</artifactId>
        <version>x.y.z</version>
    </dependency>
...
```

   Puede ser usada mediante CDI o de forma tradicional (Se incluye una factoria para facilitar el uso).

# 6. Configuración

La configuracion se realiza mediante un fichero de configuración que por defecto esta nombrado como 'BaseConfig' ,la extensión puede ser properties,yaml,json. Se puede acceder a toda la configuración programaticamente usando el objeto de configuración (SasConfig):

### Configuración por fichero:

En el fichero de configuración se establecen los diferentes DataSources que se cargaran por defecto.

- Se leerá en primer lugar el prefijo 'sasconfiguration.datasources' que deberá de contener los diferentes datasources a cargar
    - Si se usa un fichero properties se debera de definir un indice a la hora de agrupar las propiedades de cada datasources:
      ejemplo: sasconfiguration.datasources.N.propiedad
- Una vez cargados todos los datasources se leerá la propiedad 'sasconfiguration.defaultSettleableDS'
    - Esta propieda establece por nombre el datasource donde se estabelceran los nuevos parámetros si no se especifica ninguno

DataSources incluidos:

- **`JNDIConfigDataSource`**
    - Parametros configurables:
    - jndi
        - Indica el valor del jndi al que se conectara
        - Obligatorio
    - tableName
        - Nombre de la tabla que contiene la configuracion
        - Obligatorio
    - keyColumnName
        - Nombre de la columna que contiene las claves
        - Opcional, valor por defecto 'key'
    - valueColumnName
        - Nombre que contiene los valores
        - Opcional, valor por defecto 'value'
    - select
        - consulta sql que se ejecutara para obtener las configuraciones
        - Opcional, valor por defecto 'SELECT ${keyColumnName}, ${valueColumnName} FROM ${tableName}
    - allowWrite

        - Permite establecer las propiedades en la fuente de datos
        - Opcional, valor por defecto 'false'
    - loadOnCreate
        - Indica si se deberan de cargar todas las configuraciones en cache en cuanto se instancie el objeto
        - Opcional, valor por defecot 'false'
    - expiration
        - Indica la duracion de las configuraciones obtenidas en cache, se establece en milisegundos
        - Opcional, por defecto -1L
        - expiration e interval se deben de establecer siempre conjuntamente
    - interval
        - Indica cada cuanto se revisara la caducidad de las configuraciones en cache, se establece en milisegundos
        - Opcional, por defecto -1L
        - expiration e interval se deben de establecer siempre conjuntamente
    - maxSize
        - Indica el tamaño maximo de la cache
        - Opcional, por defecto -1
        - maxSize e strategyToRemove se deben de establecer siempre conjuntamente
    - strategyToRemove
        - Indica la estrategia a la hora de eliminar elementos de la cache sus valores puede ser {NONE,OLD,LESS_USED}
        - Opcional, valor por defecto 'NONE'
        - maxSize e strategyToRemove se deben de establecer siempre conjuntamente
    - allowReadFromSource
        - Habilita la lectura de la fuente de datos
        - Opcional, valor por defecto 'false'
    - priority
        - Establece la prioridad con la que sera consultado este datasource, cuanto mayor el numero mas preferencia.
        - Opcional, valor por defecto 1000+(100*{nº de datasources ya cargados})
    - name
        - Nombre del datasource
        - Opcional, valor por defecto ${jndi}
- **FileConfigDataSource**
    - Parametros configurables:

    - fileName
        - Nombre del fichero a cargar
        - Obligatorio
    - fileFormat
        - Formato del fichero a cargar, su valor puede ser {JSON,YAML,PROPERTIES}
        - Opcional. Valor por defecto 'PROPERTIES'
    - priority
        - Establece la prioridad con la que sera consultado este datasource, cuanto mayor el numero mas preferencia.
        - Opcional, valor por defecto 1000+(100*{nº de datasources ya cargados})
    - name
        - Nombre del datasource
        - Opcional, valor por defecto ${fileName}
- **GitConfigDataSource**
    - Parametros Configurables:
    - branch
        - Rama de la que obtendra el fichero
        - Opcional, valor por defecto 'master'
    - fileName
        - Nombre del fichero a cargar
        - Obligatorio
    - fileFormat
        - Formato del fichero a cargar, su valor puede ser {JSON,YAML,PROPERTIES}
        - Opcional. Valor por defecto 'JSON'
    - passphrase
        - Contraseña de la clave privada
        - Opcional valor por defecto ''
    - privateKey
        - Clave privada
        - Obligatorio
    - publicKey
        - Clave publica
        - Obligatorio
    - uri
        - url del repositorio Git
        - Obligatorio
    - priority

        - Establece la prioridad con la que sera consultado este datasource, cuanto mayor el numero mas preferencia.
        - Opcional, valor por defecto 1000+(100*{nº de datasources ya cargados})
    - name
        - Nombre del datasource
        - Obligatorio

Ejemplo properties

```
sasConfiguration.dataDources.0.type=es.ja.csalud.sas.componentescomunes.sasconfiguration.datasource.file.
FileConfigDataSource
sasConfiguration.dataSources.0.filename=config.properties
sasConfiguration.dataSources.0.fileFormat = PROPERTIES
sasConfiguration.dataSources.0.priority = 100
sasConfiguration.dataSources.1.type=es.ja.csalud.sas.componentescomunes.sasconfiguration.datasource.jndi.
JndiConfigDataSource
sasConfiguration.dataSources.1.jndi=jdbc/JAVAEE7DEMO
sasConfiguration.dataSources.1.tableName = ARQ_CONFIG
sasConfiguration.dataSources.1.name = setDS
sasConfiguration.defaultSettleableDS = setDS
```

Ejemplo fichero yaml:

```
sasConfiguration:
  dataSources:
    - type: es.ja.csalud.sas.componentescomunes.sasconfiguration.datasource.file.FileConfigDataSource
      fileName: pConfig.properties
      fileFormat: PROPERTIES
      name: propConfig
    - type: es.ja.csalud.sas.componentescomunes.sasconfiguration.datasource.file.FileConfigDataSource
      fileName: anotherConfig.json
      fileFormat: JSON
      name: jsonConfigFile
      priority: 10
    - type: es.ja.csalud.sas.componentescomunes.sasconfiguration.datasource.jndi.JndiConfigDataSource
      jndi: jdbc/myDatasource
      tableName: ARQ_CONFIG
      name: setDS

  defaultSettleableDS: setDS
```

### Configuracion en codigo

Por código se pueden configurar todas las opciones descritas con anterioridad. Tambien se pueden añadir DataSources y TransformationsHandlers propios del usuario de la libreria.

Ejemplo:

```
@Inject
SasConfig sasConfig;
...
public void init(@Observes @Initialized(ApplicationScoped.class) Object init) {
    ...
    JNDIConfigDataSource jndiConfigDataSource = JNDIConfigDataSource.Builder()
        .withJndiDataSource("jdbc/JAVAEE7DEMO")
        .withTableName("ARQ_CONFIG")
        .withName("setDS").build();
    sasConfig.addDataSource(jndiConfigDataSource).withDefaultSettableDataSourceName("setDS");
    sasConfig.addTransformation(new CustomTransformationHandler());
    ...
```

# 7. Uso

### Consultas paramétricas

La consulta de paramétricas se puede realizar de dos maneras diferentes, desde código, haciendo uso del servicio especifico de la librería "SasConfigService". Este servicio puede ser inyectado.

Ejemplo código:

```
...
@Inject
SasConfigService sasConfigService;
...
public void do(){
    ...
    String urlRestBDU = sasConfigService.get("integracion.servicios.bdu.rest.url");
    //String urlRestBDU = sasConfigService.get("integracion.servicios.bdu.rest.url","http://bdu.es/rest")
; <-- con valor por defecto
    MacoUser usuarioGenerico = sasConfigService.get("interno.usuarioGenerico", MacoUser.class); //<--
Para poder asignar parámetros a objetos directamente, hay que preconfigurar una trasformación, mas
adelante se vera como
    ...
```

Por anotaciones es mas sencillo aunque puede que existan casos que no sea lo suficientemente flexible esta opcion.

Ejemplo Anotaciones:

```
public class PeticionesBDUsample {
    @Configuration(key="integracion.servicios.bdu.rest.url",defaultValue="http://bdu.es/rest")
    String urlRestBDU;
    @Configuration(key="interno.usuarioGenerico")
    MacoUser usuarioGenerico;
    ...
}
```

Mediante codigo tambien se puede obtener un mapa de parametricas basado en un prefijo.

Ejemplo:

```
...
@Inject
SasConfigService sasConfigService;
...
public void do(){
    ...
    Map<String,String> nodosActivos = sasConfigService.getByPrefix("interno.nodos.");
    ...
```

### Transformaciones

Para poder asignar directamente un parametro a un objecto complejo, se hace uso de una interfaz, "TransformationHandler". Esta Interfaz ha sido implementada con varias clases por defecto que se encuentran en el paquete es.ja.csalud.sas.componentescomunes.sasconfiguration.transformation.base

Las clases concretas son:

- BooleanTransformationHandler
- ByteTransformationHandler
- DoubleTransformationHandler
- FloatTransformationHandler
- IntegerTransformationHandler
- LongTransformationHandler
- ShortTransformationHandler
- StringTransformationHandler
- ListTransformationHandler
- MapTransformationHandler
- FileConfigDataSourceTransformationHandler
- JndiConfigDataSourceTransformationHandler

La interfaz puede ser implementada y añadida a la configuracion para que pueda hacer transformaciones de objetos complejos especificos del proyecto que este haciendo uso de la libreria.

Se puede extender la clase abstracta 'BaseTransformationHandler' y las implementaciones tenerlas en el paquete 'es.ja.csalud.sas.componentescomunes.sasconfiguration.transformation' para que sean cargadas automaticamente. En este caso no seria necesario hacer lo siguiente.

Para añadir un TransformationHandler ya implementado, se debe de tener acceso al objeto de configuracion (injectandolo por ejemplo al inicio de la aplicacion)

Ejemplo:

```
@Inject
SasConfig sasConfig;
...
public void init(@Observes @Initialized(ApplicationScoped.class) Object init) {
    ...
    sasConfig.addTransformation(new CustomTransformationHandler());
    ...
```