Migración SICOF ERP - Proceso: Guía de Configuración de Desarrollo

La siguiente sección define las configuraciones que se deben tener presente en el desarrollo de los microservicios de la fábrica de desarrollo.

Consideraciones Previas

- Los microservicios deben aplicar las configuraciones definidas en la mayoria de los casos.
- Si un microservicio requiere configuraciones especiales, estas deben ser validadas con los lideres de desarrollo (Pablo Quintana, Daberson Henao, Carlos Torres, Gersain Castañeda).

Configuraciones del Microservicio

Todo microservicio debe tener su configuración centralizada en el repositorio git definido en la fabrica, sin embargo existen situaciones donde un microservicio puede requerir configuraciones particulares. A continuación se definen los escenarios donde aplicarán cada tipo de configuración:

Configuración Centralizada

- Configuración de Registro y Descubrimiento (Cloud Config / Eureka)
- Configuración de Routing (Zuul)
- Base de Datos Centralizada

Configuración Local

- Lógica del Negocio

Archivos de Configuración

Se define los siguientes tipos de archivos de configuración

- Configuración Centralizada: Archivos Yaml
- Configuración Local: Archivos Properties

Nombre del Microservicio y configuración POM (Maven)

- Todo microservicio que represente lógica del negocio debe iniciar con la palabra Microservicio y terminar con la palabra ADA y debe usar nomeclatura Camel Case[1] sin signos de puntuación. **Ejemplo: MicroservicioPruebaModelADA**
- El **groupId** del proyecto debe ser el nombre del paquete principal. **Ejemplo:**

<groupId>co.ada.test.prueba</groupId>

- El **artifactId** y **name** deben ser iguales al nombre del proyecto. **Ejemplo: <artifactId>MicroservicioPruebaModelADA</artifactId>** y **<name>MicroservicioPruebaModel</name>**
- Todo microservicio debe contener una descripción. **Ejemplo: <description>Microservico de prueba para conexiones multiples</description>**
- Todo microservicio debe definir empaquetado tipo jar. **Ejemplo: <packaging>jar</packaging>**
- Todo desarrollador que actualice el microservicio debe registrarse en el POM en la etiqueta Developer.

Ejemplo Sección Nombre:

```
<groupId>co.ada.test.prueba</groupId>
<artifactId>MicroservicioPruebaModelADA</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>MicroservicioPruebaModelADA</name>
<description>Microservico de prueba para conexiones multiples</description>
<packaging>jar</packaging>
```

Ejemplo Sección Developers:

```
        <developers>
        <developer>
            <id>carlos.torres</id>
            <name>Carlos Torres</name>
            <email>carlos.torres@ada.co</email>
            <organization>ADA S.A.</organization>
            <organizationUrl>www.ada.co</organizationUrl>
            <roles>
                <role>architect</role>
                <role>developer</role>
            </roles>
            <timezone>America/Bogota</timezone>
        </developer>
    </developers>
```

Reglas de Nombre del Microservicio y Archivo de Configuración

- El nombre de la aplicación del microservicio debe ser definido de acuerdo al nombre exacto del paquete principal **Ejemplo**: *co.ada.sicof.terceros*
- El nombre del archivo de configuración centralizada debe ser **bootstrap.yml**
- El nombre del archivo de configuracion local debe ser **application.properties**
- Deben existir configuraciones por cada ambiente de despliegue en el repositorio de configuración y cada archivo de configuración debe indicar el perfil los cuales son dev: desarrollo - test: QA - prod: producción **Ejemplo:** Microservicio: Terceros, Paquete Principal: *co.ada.sicof.terceros*, Nombre del Microservicio spring.application.name=co.ada.sicof.terceros, perfil de ambiente de despliegue de desarrollo: dev, Nombre del archivo de configuración: **co.ada.sicof.terceros-dev.yml**

- La creación de los archivos de configuración centralizada deben ser solicitados al administrador del repositorio de configuración (Pablo Quintana / Carlos Torres).

Dominio de Clases de Entidades Comunes (Models)

En la arquitectura propuesta es muy común que existan microservicios que proveen información y otros que la procesan, por esta razon hace necesario el conocimiento de las estructuras en las cuales se transportan los datos ya que en muchos escenarios persisten en el sistema. Teniendo presente esta situación recurrente se crea un domino de clases y entidades comunes (proyecto spring) donde se van a registrar todas la entidades que puedan ser requeridas por microservicios. A ontinuación se definen las reglas a considerar para incluir una entidad.

- Cada clase/entidad no debe tener lógica de negocio.
- Cada clase/entidad debe describir la estructura en su totalidad
- Cada clase/entidad debe estar agrupada según la estructura de definición de su parque principal teniendo presente que el nombre base empezará con la estrucutra base del domino de clases y entidades comunes Ejemplo: Para el microservicio **co.ada.sicof.contabilidad.tercero** su agrupación de entidades en el proyecto de dominio de clases de entidades comunes debe ser **co.ada.models.sicof.contabilidad.tercero**
- Se puede inlcuir definiciones resumidas de una entidad.
- La inclusión de esas entidades debe ser validada y aprobada por el administrador del dominio de clases y entidades comunes.

Como utilizar el proyecto Models

De acuerdo al ambiente de despliegue el proyecto domino de clases y entidades comunes estará en uno de los siguientes repositorios (versión compilada .jar):

- Desarrollo (dev): http://adacsc.co:1443/svn/repository/ADA/SICOFERP/fuentes/branches/development/core/ModelsClassesADA/target/CommonsClassesADA-0.0.1-SNAPSHOT.jar
- Pruebas/calidad (qa):
- Producción (prod):

Dependencia Maven

Esta es la dependencia que debe agregar en el POM del microservicio.

```
<dependency>
  <groupId>co.ada.models</groupId>
  <artifactId>ModelsClassesADA</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</dependency>
```

Es responsabilidad del desarrollador utilizar la versión más reciente las cual será publica en la sección correspondiente de la wiki.

Para utilizar el dominio de clases y entidades comunes en los micoservicios siga los siguientes pasos:

1. Identifique el ambiente dev, qa o prod a utilizar
2. Identifique la versión actual (muy importante para evitar versiones obsoletas)
3. Importe la dependencia en Maven
4. Agregue la dependencia @EntityScan con el array de los paquetes o clases que necesita en la clase principal del microservicio.

←Volver atrás

1)

https://es.wikipedia.org/wiki/Camel_case