Tot plugin example

Control de versiones

Versión	Fecha de modificación	Responsable	Aprobador	Resumen de cambios
1.0	28/10/2022	Anjana Producto	Anjana Producto	Creación del documento

Introducción

Este documento es un kit de desarrollo de plugins.

Estructura del proyecto

Modelo de objeto de proyecto

Se trata del fichero pom.xml donde se tiene información sobre el proyecto y los detalles de configuración utilizados para construir el proyecto como por ejemplo la versión del proyecto, la descripción, las dependencias etc.
Ejemplo: tot-plugin-example/pom.xml
Ref. Doc.: Apache Maven

Archivo de configuración

La forma que tenemos de configurar es mediante archivo de configuración YAML.
Ejemplo: tot-plugin-example/src/main/resources/application.yaml
Ref. Doc.: Spring.io

Clase principal (mainClass)

La clase con el método main que necesitamos para iniciar la aplicación. Se anota con @SpringBootApplication y contiene la instrucción de arranque de Spring: SpringApplication.run(...), ya incluye los paquetes básicos para que la aplicación arranque e interactúe con tot.
Ejemplo:
tot-plugin-example/src/main/java/com/anjana/tot/plugin/example/TotPluginExampleApplication.java

Clase Configuration

Para cargar el conjunto de propiedades relacionadas desde el fichero de configuración, se crea una clase de bean con la anotación @Configuration. Ver sección Tot-plugin-lib
Ejemplo:
tot-plugin-example/src/main/java/com/anjana/tot/plugin/example/configuration/ExampleConfiguration.java

Controlador

Una vez tenemos esto procedemos a crear la clase Controlador anotado con @RestController para definir las operaciones que se llevarán a cabo. Ver sección Tot-plugin-lib
Ejemplo:
tot-plugin-example/src/main/java/com/anjana/tot/plugin/example/controller/OperationsController.java

Operaciones que podría llevar a cabo un plugin:

➔ Gestión de estructuras:
- Crear dataset: El punto de entrada de esta funcionalidad se encuentra en StructureManagementControllerInterface (Ver sección Tot-plugin-lib). Este endpoint se encarga de crear un objeto (ya sea tabla, fichero o cualquier cosa que se quiera) utilizando el metadata de un dataset en anjana. Recibe la metadata del dataset y sus datasetfields. Esta es la acción que se realiza cuando se crea un dataset gobernado o bien se modifica uno de no gobernado a gobernado.

➔ Gestión de permisos:
- Crear grupo: Para ello se crea el método createGroup cuyo endpoint se encuentra en IdentityManagementControllerInterfaceV2 (Ver sección Tot-plugin-lib). La funcionalidad de este endpoint es la de crear un grupo y asignar permisos. Para ello se crea un rol en base de datos y se otorga permiso a ese rol para algunas tablas. Los DTOs que utiliza dicho endpoint son DSAOperationResultDTO, DSAFullInfoDTO (Ver sección Tot-lib). Esta es la acción que el plugin realiza cuando se crea o versiona un DSA con datasets gobernados.

- Eliminar grupo: Para ello se crea el método deleteGroup cuyo endpoint se encuentra en IdentityManagementControllerInterfaceV2 (Ver sección Tot-plugin-lib). La funcionalidad de este endpoint es la de eliminar un grupo y desasignar permisos. Para ello se elimina el rol que tiene los permisos que quieran revocar. Los DTOs que utiliza dicho endpoint son DSAOperationResultDTO, DSAFullInfoDTO (Ver sección Tot-lib). Esta es la acción que el plugin realiza cuando se expira un DSA con datasets gobernados.

- Eliminar objeto de un grupo: Para ello se crea el método removeObject cuyo endpoint se encuentra en IdentityManagementControllerInterfaceV2 (Ver sección Tot-plugin-lib). La funcionalidad de este endpoint es la de revocar los permisos que tiene un rol sobre algunas tablas. Los DTOs que utiliza dicho endpoint son ObjectOperationResultDTO, ObjectOperationDTO (Ver sección Tot-lib). Esta es la acción que el plugin realiza cuando se expira un dataset gobernado.

- Asignar usuario: Para ello se crea el método addUser cuyo endpoint se encuentra en IdentityManagementControllerInterfaceV2 (Ver sección Tot-plugin-lib). La funcionalidad de este endpoint es la de modificar usuarios en las políticas de una tabla. Los DTOs que utiliza dicho endpoint son DSAOperationResultDTO, DSAFullInfoDTO (Ver sección Tot-lib). Esta es la acción que el plugin realiza cuando un usuario se adhiere a un DSA con datasets gobernados.

- Desasignar usuario: Para ello se crea el método removeUser cuyo endpoint se encuentra en IdentityManagementControllerInterfaceV2 (Ver sección Tot-plugin-lib). La funcionalidad de este endpoint es la de modificar usuarios en las políticas de una tabla. Los DTOs que utiliza dicho endpoint son DSAOperationResultDTO, DSAFullInfoDTO (Ver sección

Tot-lib). Esta es la acción que el plugin realiza cuando un usuario se desadhiere a un DSA con datasets gobernados.

➔ Muestreo de datos: Para ello se crea el método sample cuyo endpoint se encuentra en SampleDataControllerInterface (Ver sección Tot-plugin-lib). La funcionalidad de este endpoint es la de obtener un muestreo de los datos. Los DTOs que utiliza dicho endpoint son SampleDataDTO, MetadataInfoDTO (Ver sección Tot-lib). Esta es la acción que el plugin realiza cuando un usuario entra a la pestaña de muestreo de datos en un dataset con dicho flag a true en un dataset gobernado.

➔ Extracción metadatos: En esta sección distinguimos entre dos métodos: metadataList y metadataExtract.

- metadataList: Su endpoint se encuentra en ExtractMetadataControllerInterface (Ver sección Tot-plugin-lib). Su función es la de obtener un listado general de objetos extraídos de una tecnología. Los DTOs que utiliza dicho endpoint son ResourceStructureDTO, ResourceDTO (Ver sección Tot-lib).
- metadataExtract: Su endpoint se encuentra en ExtractMetadataControllerInterface (Ver sección Tot-plugin-lib). Su función es la extracción de un objeto específico de una tecnología. Los DTOs que utiliza dicho endpoint son MetadataImportedDTO, ResourceMetadataRequestDTO (Ver sección Tot-lib).

Se recomienda desarrollar la sección de extracción de metadatos de forma asíncrona ya que se puede generar cierto retraso por parte de las tecnologías en devolver una respuesta.

Servicios

La capa de servicios consiste de clases anotadas con @Service.Se usa esta anotación para marcar las clases como proveedores de servicios ya que son las que brindan funcionalidades. Para cada funcionalidad se implementa una clase Service específica.

Ejemplo:

- Sampleo de datos `tot-plugin-example/src/main/java/com/anjana/tot/plugin/example/service/SampleDataService.java`
- Extracción de metadata `tot-plugin-example/src/main/java/com/anjana/tot/plugin/example/service/ExtractMetadataService.java`

Visión general de la distribución de directorios, ficheros y clases:

```
- tot-plugin-example
  - src
    - main
      - java/com/anjana/tot/plugin/example
          - configuration
```

```
            ExampleConfiguration.java
        - controller
            OperationsController.java
        - service
            ExtractMetadataService.java
            PermissionService.java
            SampleDataService.java
        TotPluginExampleApplication.java
    - resources
        application.yaml
  - test/resources
pom.xml
```

Librerías

Disponemos de tres librerías que brindan implementación de algunas funcionalidades. Estas son: tot-lib, tot-plugin-lib y customers-lib.
A continuación, se explican algunas funcionalidades de cada una de ellas que pueden resultar de interés para el desarrollo del plugin.

Tot-lib

Uno de los aspectos más importantes de esta librería es el amplio repertorio de DTOs que ofrece para el mapeo de datos.
Se encuentran en: tot-lib/src/main/java/com/anjana/tot/core/model
Otras clases de interés: TotConstants.java , TotUtils.java en tot-lib/src/main/java/com/anjana/tot/core/util

Customers-lib

Al igual que la librería tot-lib esta librería ofrece varios DTOs que pueden ser de utilidad en customers-lib/src/main/java/com/anjana/core/customers/model además de varias clases que contienen enumeradores para tener valores tipados en customers-lib/src/main/java/com/anjana/core/customers/enums .

Tot-plugin-lib

Uno de los aspectos más importantes de esta librería son las interfaces con los endpoints necesarios para el funcionamiento del plugin que implementará el controlador de operaciones del plugin, estas se ubican en tot-plugin-lib/src/main/java/com/anjana/tot/plugin/core/interfaces .
Otra clase de interés es la de PluginConfiguration.java ubicada en src/main/java/com/anjana/tot/plugin/core/configuration/PluginConfiguration.java .

Esta contiene las propiedades más comunes de un plugin que son la ubicación (location), el servidor (server), listado de aris (aris) y el prefijo (groupPrefix). De esta debería extender la clase de configuración del plugin a la que se le podrán añadir el resto de propiedades que se quieran recuperar del fichero de configuración.
Esta librería contiene además otras clases de interés como puede ser la clase TotPluginException.java en tot-plugin-lib/src/main/java/com/anjana/tot/plugin/core/exception para capturar las excepciones que puede lanzar el plugin.
Contiene utilidades básicas para poder leer archivos avro, parquet, excel y csv (que son los que usan los plugins propios de anjana para extraer información de esos archivos y en el muestreo de datos). Si se quieren utilizar se deberá incluir el paquete com.anjana.tot.plugin.core.container como otro paquete a escanear en la clase principal del plugin.

Cómo crear un plugin

Este esqueleto solo contiene lo necesario (incluyendo configuración de ejemplo) para levantar el proyecto y que se registre en tot.
Para hacer un plugin totalmente funcional son necesarios varios pasos.

Dependencias

Las dependencias incluidas en el ejemplo solo sirven para levantar un spring boot y tener acceso a la utilidades que el resto de plugins tienen. Se deberán incluir aquellas que sean necesarias para la integración que se desee desarrollar. (Aviso: Cambiar la versión de spring por necesidades de la integración podría afectar a la interacción con tot y con anjana).

Unas dependencias que no se han incluido son las dependencias de tratamiento de ficheros avro, parquet, excel y csv que utilizan todos los plugin nativos de anjana. En caso de que se quieran utilizar se deben incluir las siguientes dependencias (además de incluir la paqueteria com.anjana.tot.plugin.core.container en la clase principal del proyecto para que inicie los componentes de lectura de ficheros).

```
<!-- apache avro content -->
<dependency>
        <groupId>org.apache.avro</groupId>
        <artifactId>avro</artifactId>
        <version>1.10.1</version>
</dependency>

<!-- apache parquet content -->
<dependency>
        <groupId>org.apache.parquet</groupId>
        <artifactId>parquet-common</artifactId>
        <version>1.11.1</version>
</dependency>
```

```
<dependency>
        <groupId>org.apache.parquet</groupId>
        <artifactId>parquet-hadoop</artifactId>
        <version>1.11.1</version>
</dependency>
<dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>3.3.0</version>
</dependency>

<!-- Excel content -->
<dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi</artifactId>
        <version>5.0.0</version>
</dependency>
<dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi-ooxml</artifactId>
        <version>5.0.0</version>
</dependency>

<!-- CVS Content -->
<dependency>
        <groupId>commons-validator</groupId>
        <artifactId>commons-validator</artifactId>
        <version>1.7</version>
</dependency>
```

Configuración

La configuración incluida es únicamente para la interacción correcta con tot, y se deben adaptar sus valores a los que se necesiten.

```
totplugin:
 location: http://localhost:15002/plugin/example/api/v1
 server:
   url: http://localhost:15000/tot/
```

```
  keep-alive-seconds: 60
aris:
  - ari: "anja:totplugin:extract:/infra/tech/zone/"
  - ari: "anja:totplugin:sample:/infra/tech/zone/"
  - ari: "anja:totplugin:im:/infra/tech/zone/"
encryption:
  privateKey: privatekey
  tot:
    publicKey: publickey
```

- location: Cual es la url de acceso al plugin, necesaria para que tot pueda redirigir llamadas al plugin desarrollado.
- server.url: Url donde está tot desplegado, necesaria para poder registrar, desregistrarse y mantenerse en los registros de tot.
- server.keep-alive-seconds: Cada cuánto tiempo va a volver a registrarse en tot para seguir registrado.
- aris: Poner las tripletas (infraestructura, tecnología y zona) que el plugin gobierna y qué funcionalidades ofrece (extract: extracción, sample: muestreo de datos, im: gobierno activo).
- encryption.privateKey y encryption.tot.publicKey: Deben ser las misma configuradas que se configuren en tot para que su securización funcione correctamente.

Servicios

Los servicios incluidos vienen totalmente vacíos, aquí es donde se desarrollará toda la lógica que se necesite en la integración. Los nombres de los métodos y estructura se pueden cambiar libremente, lo único es que se deben llamar desde el controller que es, al fin y al cabo, el punto de entrada en el plugin.