EUSKO JAURLARITZA
GOBIERNO VASCO

Programa Saiotek 2006

SMARTLAB

Entorno de Trabajo Inteligente Colaborativo y Programable

Manual del programador del bundle VideoIP

Tecnológico Fundación Deusto
Teknologikoa Deustu Fundazioa

HISTORIAL DE CAMBIOS

Versión	Descripción	Autor	Fecha	Comentarios
V0.1	Versión inicial	Unai Aguilera	23/01/2008	

TABLA DE CONTENIDOS

1 INTRODUCCIÓN

En éste documento se recoge el manual del programador del bundle de VideoIP explicando cuál es la estructura del mismo y cuales son los pasos necesarios para extender la funcionalidad del mismo.

El bundle de VideoIP es el encargado de llevar a cabo la gestión de las cámaras de videoip. Un mismo bundle puede llevar a cabo el control y gestión de varias cámaras. Debido a que cada una de las cámaras existentes en el mercado tiene unas características distintas se ha realizado un diseño que permite abstraer el uso de las cámaras de las implementaciones específicas de cada modelo. Todas las cámaras deben implementar la interfaz *ICameraIP* que especifica la funcionalidad que deben proporcionar.

2 CLASES IMPORTANTES

es.deusto.tecnologico.zaingune.videoip.ICameraIP

Esta clase define la interfaz que debe ser implementada por las diferentes cámaras ip para proporcionar la funcionalidad correspondiente. Los métodos que define la interfaz son las siguientes:

- move(Direction dir): mueve la cámara en una dirección determinada. Las direcciones en las que se puede mover la cámara son: LEFT, RIGHT, DOWN, UP.
- enableAutoPatrol(): activa el modo de patrulla de la cámara.
- enableAutoPan(): activa el modo de auto-panning de la cámara.
- stopAutoMotion(): detiene el movimiento automático de la cámara, tanto el autopatrol como el autopanning.
- resetPosition(): sitúa a la cámara en la posición original.
- enableMotionDetection(boolean enable): active la detección de movimiento de la cámara haciendo que esta notifique al bundle los eventos de detección de movimiento que se produzca.
- setEnabled(boolean enable): activa o desactiva la cámara, incluyendo la detección de movimiento y la grabación de imágenes si está soportada por la cámara.
- getImage(): permite obtener la imagen actual capturada por la cámara.
- loadConfig(): este método es el encargado de llevar a cabo la configuración de la cámara.

Actualmente sólo existe una implementación de esta interfaz correspondiente con el modelo de cámara DLink 5300G.

es.deusto.tecnologico.zaingune.videoip.VideoIPController

Esta interfaz define la funcionalidad del controlador de video ip que se encarga de la gestión

de las diferentes cámaras. Su función es permitir el acceso a las cámaras registradas en el sistema. Los métodos más importantes que proporciona son los siguientes:

- addCamera(ICameraIP camera): añade una cámara a la colección y la registra en OSGi como un servicio.
- removeCamera(ICameraIP camera): elimina la cámara correspondiente de OSGi y de la colección de cámaras gestionadas.
- getCamera(String name): permite obtener una cámara a partir de su nombre.
- getAllCameras(): obtiene la colección de todas las cámaras registradas en el bundle.
- getNumCameras(): obtiene el número de cámaras registradas en el bundle.

3 EXTENSIÓN DEL BUNDLE

El bundle puede ser extendido añadiendo nueva funcionalidad mediante la integración de nuevas cámaras IP. Los pasos para llevar a cabo el bundle añadiendo nuevas cámaras son los siguientes:

1. Implementar la interfaz ICamaraIP para el nuevo modelo de cámara.
2. Modificar el fichero de configuración para añadir los datos de las nuevas cámaras.

El formato del fichero de configuración es el siguiente:

```
<?xml version="1.0"?>
<videoipconfig>
      <camera implClass="es.deusto.tecnologico.zaingune.videoip.impl.dlink5300g.DLink5300G"
            name="Smartlab B Camera"
            username="ftpuser" ip="20.0.0.242" pass="ftpuser123"/>
</videoipconfig>
```

El significado de los atributos del fichero es el siguiente:

- **implClass:** es la clase que implementa la funcionalidad de la cámara.

El resto de los datos es propio de la cámara que se esté instanciando. En el caso del modelo DLink 5300G los datos que deben ser indicados en el fichero para su correcto funcionamiento son los siguientes:

- **name:** es el nombre de la cámara.
- **ip:** dirección ip de la cámara. Esta información es necesaria para llevar al control de la misma de forma remota.
- **username y password:** es el usuario y password que utilizará la cámara para conectarse al servidor FTP utilizado en la detección de movimiento. Este servidor FTP es necesario debido a las características de este modelo de cámara en concreto.

4 INTEGRACIÓN EN LA PLATAFORMA SMARTLAB

El bundle de Video IP desarrollado dentro del proyecto Zaingune ha sido extendido para añadir capacidades de descripción semántica que pueden ser registrados en el razonador semántico.

Se ha llevado a cabo una modificación de algunas clases del bundle original extendiéndolas para añadir la nueva funcionalidad.

La clase principal del bundle ha sido creada extendiendo la funcionalidad proporcionada por la clase principal del bundle de VideoIP desarrollado para Zaingune. Sin embargo, se ha modificado para añadir la funcionalidad necesaria para realizar el registro y la carga de la configuración de las cámaras web.

El fichero de configuración de las cámaras (cameras.xml) ha sido modificado para añadir información sobre la localización de las cámaras quedando el formato del mismo de la siguiente manera:

```
<?xml version="1.0"?>
<videoipconfig>
      <camera implClass="es.deusto.tecnologico.smartlab.videoip.impl.dlink5300g.SDLink5300G"
                  name="Smartlab B Camera"
                  username="ftpuser" ip="20.0.0.242" pass="ftpuser123"
                  posX="15"
                  posY="20"/>
</videoipconfig>
```

La implementación existente de la cámara DLink5300G ha sido extendida implementando la interfaz *ISmartlabService* que proporciona los métodos necesarios para que el razonador pueda acceder a la información semántica de la misma.

El método getIndividual() devuelve la descripción del dispositivo de la cámara conteniendo información sobre su localización y nombre, siendo una instancia de la clase cámara definida en la ontología Smartlab. La plantilla que posteriormente es utilizada para rellenar los datos de la cámara es la siguiente:

<? xml version="1.0"?>

```
<rdf:RDF
    xmlns="http://www.morelab.deusto.es/smartlab.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.morelab.deusto.es/smartlab.owl">

  <!--Camera INDIVIDUAL-->
  <SpatialPoint rdf:ID="InsertPointIndividualName">
    <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >insertPointName</name>
    <x rdf:datatype="http://www.w3.org/2001/XMLSchema#float">55555.0</x>
    <y rdf:datatype="http://www.w3.org/2001/XMLSchema#float">666666.0</y>
    <containsDeviceItem>
      <RoboticCamera rdf:ID="InsertBinaryLightIndividualName">
        <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >InsertName</name>
        <id xml:lang="es">insertID</id>
        <deviceType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >hybrid</deviceType>
        <placedIn rdf:resource="#InsertPointIndividualName"/>
      </RoboticCamera>
    </containsDeviceItem>
  </SpatialPoint>
</rdf:RDF>
```

Por otro lado, la invocación del método *getOntology()* devolverá al razonador la ontología que se utilizará para representar el estado de la cámara cada vez que ha existido una detección de movimiento. La ontología que la cámara añade a la ontología Smartlab es la siguiente:

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.morelab.deusto.es/smartlab.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.morelab.deusto.es/smartlab.owl">

  <!--CAMERA STATUS ONTOLOGY-->
  <owl:Class rdf:ID="CameraStatus">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="motiondetected"/>
        </owl:onProperty>
```

```
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#ValueItem"/>
  </owl:Class>
  <owl:DatatypeProperty rdf:about="#motiondetected">
    <rdfs:domain rdf:resource="#CameraStatus"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  </owl:DatatypeProperty>

</rdf:RDF>
```

Cada vez que se produzca un evento se generará un concepto que será introducido en la ontología indicando el momento en el que se ha producido y el tipo del mismo. La plantilla creada que será completada con los datos correspondientes cada vez que se produzca un evento es la siguiente:

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.morelab.deusto.es/smartlab.owl#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.morelab.deusto.es/smartlab.owl">

  <!--CAMERA STATUS INDIVIDUAL-->
  <CameraStatus rdf:ID="InsertCameraStatusIndividualName">
    <targets rdf:resource="#InsertDeviceIndividualName"/>
    <motiondetected rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</motiondetected>
    <time>
      <Instant rdf:ID="InsertCameraStatusInstantName">
        <value rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
        >1950-11-11T11:11:11</value>
      </Instant>
    </time>
  </CameraStatus>
</rdf:RDF>
```