EUSKO JAURLARITZA
GOBIERNO VASCO

Programa Saiotek 2006

SMARTLAB

Entorno de Trabajo Inteligente Colaborativo y Programable

Manual del programador del bundle EIB/KNX

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 EIB/KNX explicando cuál es la estructura del mismo y cuales son los pasos necesarios para extenderlo.
Se explican a continuación las clases más importantes de la implementación y cómo pueden extenderse para añadir funcionalidad.

2 RESUMEN

El cliente EIB/KNX ha sido implementado utilizado una librería llamada Calimero (http://calimero.sourceforge.net/) que permite llevar a cabo la conexión con el servidor Tweety y el control de los distintos dispositivos EIB, así como la escucha de los eventos que se producen en el bus domótico.

Para facilitar el uso de la librería Calimero y su utilización cómo bundle dentro de la plataforma OSGi, se han construido una serie de clase que permiten abstraer el funcionamiento de la misma y de los dispositivos EIB subyacentes. Por cada uno de los tipos de dispositivos existentes en el bus domótico se ha creado una interfaz que define las funciones que se pueden realizar con cada uno de los dispositivos. Cada una de estas interfaces ha sido implementada con la funcionalidad correspondiente para acceder a los dispositivos EIB del tipo correcto enviando y recibiendo los datos en el formato utilizado por la librería Calimero y convirtiéndolos al formato correcto utilizado por la interfaz y que permite abstraer a los usuarios de los tipos de datos utilizados.

Posteriormente se explicará cada una de estas interfaces indicando los métodos que proporcionan.

3 CLASES IMPORTANTES

es.deusto.tecnologico.zaingune.eib.EIBController

Esta clase proporciona la funcionalidad de conexión y desconexión con el servidor EIB. Los métodos más importantes de la misma son los siguientes:

- connect(String address, int port): este método permite llevar a cabo la conexión con el servidor EIB/KNX Tweety. Para realizar la conexión necesita la dirección IP y el puerto donde se encuentra el servidor.

- disconnect(): este método permite realizar la desconexión del servidor Tweety EIB/KNX.

- loadEIBDeviceList(String filename, EIBDataReader reader): permite cargar una lista de configuración de los dispositivos a partir de un fichero utilizando un reader adecuado. EIBDataReader se trata de una interfaz Java que debe ser implementada por aquellas clase que tenga la funcionalidad de carga de un fichero concreto.

- getDevice(String devicename): permite obtener un dispositivo a partir de su nombre.

- addChangeListener(ValueChangedListener changeListener): permite añadir un escuchador de eventos a todos los dispositivos controlados por el bus EIB/KNX y cargados a través del método loadEIBDeviceList.

- removeChangeListener(ValueChangedListener changeListener): permite eliminar un escuchador de todos los dispositivos controlados a través del EIBController.

- initialize(): este método debe ser llamado para iniciar correctamente los dispositivos después de su carga.

es.deusto.tecnologico.zaingune.eib.osgi.impl.EIBControllerOSGIImpl

Esta clase implementa la interfaz EIBController y los métodos necesarios para que la clase pueda ser utilizada como un bundle OSGi. Esta clase implementa el patrón declarative service de OSGi, porque lo toda la funcionalidad de inicio y parada del bundle y por lo tanto del cliente EIB/KNX se realiza a través de los siguientes métodos:

- activate(ComponentContext context): este método realiza la inicialización del bundle y lleva a cabo la carga de la configuración del cliente EIB/KNX. La configuración es cargada del fichero data/eibconfig.xml. Además en este método se lleva a cabo el registro de los servicios de dispositivo. Cada uno de los dispositivos cargados a través del fichero de configuración es representado por una interfaz que facilita la tarea de acceso a la funcionalidad del dispositivo.

- deactivate(ComponentContext context): lleva a cabo la desactivación del bundle desregistrando los servicios OSGi para cada uno de los dispositivos proporcionados por el bus domótico según el fichero de configuración cargado por el bundle.

- registerServices(BundleContext bundleContext): realiza la carga de los servicios OSGi que representan cada uno de los dispositivos EIB.

4 INTERFACES DE DISPOSITIVO

Las interfaces de dispositivo son implementadas por clases que encapsulan la comunicación a través de la librería Calimero para abstraer al usuario de las particularidades de comunicación de la misma. Todas estas interfaces extienden la interfaz EIBService, que proporciona una serie de métodos comúnes y necesarios para todos los servicios de dispositivo.

es.deusto.tecnologico.zaingune.eib.services.EIBService

Esta interfaz define métodos genéricos para todos los servicios de dispositivo EIB que van a registrarse en el OSGi. Los métodos de la interfaz son los siguientes:

- getDeviceName(): proporciona el nombre del dispositivo.
- registerListener(EIBServiceListener listener): registra un EIBServiceListener para escuchar los eventos que se produzcan en el dispositivo.
- unregisterListener(EIBServiceListener listener): permite deregistrar un EIBServiceListener del dispositivo representado por el servicio OSGi correspondiente.
- getLastTimeStamp(): obtiene el timestamp del último evento producido por el dispositivo.

Debido a que el bus EIB no proporciona información dinámicamente sobre los dispositivos que lo constituyen esta información debe se especificado a través de un fichero durante la carga del bundle. Esta información se encuentra en el fichero data/devices.xml. El formato del fichero es el siguiente:

```
<eibsmartlab>
    <devices>
      <device interface="es.deusto.tecnologico.zaingune.eib.services.IThermometer"
                  class="es.deusto.tecnologico.zaingune.eib.devices.Thermometer">
            <eibdevice>Temperatura Actual</eibdevice>
      </device>
</devices>
<eibsmartlab>
```

Si se quiere añadir un nuevo tipo de dispositivo se deberá crear una interfaz que lo represente y que posteriormente será registrada en OSGI como un servicio e implementarla utilizando los comando de comunicación proporcionados por el bundle y que encapsulan la librería Calimero. Posteriormente se deberá modificar el fichero data/devices.xml para añadir los nuevos dispositivos y asociarlos con las interfaces e implementaciones correspondientes.

Los dispositivos actualmente implementados son los siguientes:

1. IBinaryLight: los servicios de este tipo representan luces de tipo interruptor, es decir, luces que pueden estar solamente en dos estados: encendido o apagado. Los servicios de este tipo cuentan con métodos para cambiar el estado de la luz (encendido o apagado) y para leer el estado actual de la misma.

2. IDimmableLight: pertenecen a este tipo de servicios aquellas luces que pueden variar su intensidad. Por lo tanto, existen métodos para leer y cambiar la luminosidad de la luz, especificada como un valor decimal entre 0 y 100. Se ha considerado que este tipo de servicios son un subtipo de los servicios de tipo BinaryLight. Por esta razón, este tipo de servicios proporcionan también la funcionalidad de apagado y encendido a través de los métodos correspondientes.

3. IThermometer: representa un servicio de temperatura correspondiente con un sensor en el bus. En este caso, al tratarse de un sensor, solamente se proporciona un método para dar la el valor actual de la temperatura mediante un valor decimal en grados centígrados.

4. IAlarm: esta interfaz se corresponde con aquellos dispositivos que proporcionan una alarma en el entorno, por ejemplo, sensores de inundación que puedan ser instalados en el bus domótico. Este tipo de dispositivos proporciona métodos para preguntar el último estado conocido del dispositivo de alarma. Sin embargo, debido a que resultaría ineficiente preguntar constantemente por el estado de una alarma para comprobar si esta se ha disparado, la utilidad principal de los dispositivos de este tipo viene gracias a la existencia del sistema de eventos que será explicado posteriormente.

5. IOpeningSensor: este interfaz representa aquellos dispositivos EIB que proporcionan información relativa a la apertura de algún elemento, por ejemplo una puerta o una

ventana. Debido a que se trata de sensores de apertura sólo es posible leer el estado actual, siendo imposible escribir un nuevo valor sobre el dispositivo. El dispositivo puede estar en dos estados (abierto o cerrado) correspondientes con el estado de la ventana o puerta correspondiente.

6. IHeating: esta interfaz representa a los dispositivos para el control de la calefacción del bus EIB. Proporciona un método para obtener el valor actual de la calefacción y otro para cambiar el valor del mismo. Los valores que puede tomar como parámetro estos métodos son valores comprendidos entre 0 – 100.

5 INTEGRACIÓN DENTRO DEL PROYECTO SMARTLAB

Para llevar a cabo la integración de la funcionalidad anteriormente comentada dentro de la plataforma Smartlab añadiendo información semántica y permitiendo su funcionamiento junto con el razonador semántico ha sido necesario implementar una serie de interfaces y extender la funcionalidad desarrollada anteriormente para el proyecto *Zaingune*.

El bundle desarrollado dentro del proyecto *Zaingune* proporciona toda la funcionalidad necesaria para el manejo del bus EIB/KNX pero no introduce descripciones semánticas de los dispositivos. El nuevo bundle utiliza como librería el bundle desarrollado dentro del proyecto Zaingune extendiendo las clases para añadir la funcionalidad.

Entre las modificaciones que se han realizado del bundle realizado para el proyecto Zaingune para posibilitar su uso dentro de la plataforma Smartlab se encuentra:

1. Se ha modificado la clase principal del bundle reinplementando el método encargado del registro de los servicios en OSGi. Esto ha sido necesario debido a que los servicios deben ser registrados además a través de una nueva interfaz llamada *ISmartlabService* que será la que el razonador semántico utilizará para acceder a las descripciones semánticas de los dispostivos ofrecidos por el bundle EIB/KNX.

2. Se ha modificado la carga de la configuración para añadir nueva información. El formato del fichero de configuración ha sido extendido para contener información sobre la localización de los distintos dispositivos de esa forma se permite el razonamiento sobre la posición de los mismos.

```
<?xml version="1.0"?>
<eibsmartlab>
	<devices>
		<device interface="es.deusto.tecnologico.zaingune.eib.services.IThermometer"
			class="es.deusto.tecnologico.smartlab.eib.devices.SThermometer"
			posX="15"
			posY="20">
			<eibdevice>Temperatura Actual</eibdevice>
		</device>
	</devices>
</eibsmartlab>
```

Sin embargo la modificación más importante introducida a la hora de desarrollar el nuveo

bundle se corresponde con implementación de las clases de dispositivo que ahora deben proporcionar información semántica para que pueda ser incluida en el razonador. Los dispositivos que proporcionan información semántica deben implementar la interfaz *ISmartlabService,* siendo esta interfaz la utilizada por el razonador para acceder a la información del dispositivo.

A continuación se explica como se ha llevado a cabo la descripción semántica, mediante la implementación de la interfaz comentada anteriormente, de alguno de los dispositivos pertenecientes al bus EIB/KNX.

DimmableLight

Este dispositivo representa las luces que pueden ser tanto encendidas y apagadas como atenuadas. Además este tipo de dispositivo es un actuador por lo que puede responder a determinados eventos como el encendido y apagado de las mismas que se generen en el razonador debido a la ejecución de determinadas reglas.

Se han desarrollado una serie de plantillas OWL para describir al dispositivo que posteriormente son rellenados en tiempo de ejecución con los datos correspondientes al dispositivo antes de ser introducidos en el razonador.

El dispositivo debe proporcionar información semántica sobre si mismo al razonador. Esta información incluye el tipo de dispositivo, nombre, localización que será utilizada posteriormente por el razonador para inferir nuevo conocimiento sobre los dispositivos. El razonador invocará el método *getIndividual()* para obtener esta información.

```
<?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">

  <!--DimmableLight 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>
      <DimmableLight rdf:ID="InsertDimmableLightIndividualName">
```

```
        <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"/>
      </DimmableLight>
    </containsDeviceItem>
  </SpatialPoint>

</rdf:RDF>
```

Por otro lado, el método *getOntology()* es utilizado por el razonador semántico para obtener la ontología sobre el estado del dispositivo. En este caso que sirve de ejemplo indicará si la luz está encendida o apagada y cuál es su luminosidad.

```
<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">

  <!--DimmableLight 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>
      <DimmableLight rdf:ID="InsertDimmableLightIndividualName">
        <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"/>
      </DimmableLight>
    </containsDeviceItem>
  </SpatialPoint>

</rdf:RDF>
```

Cada vez que se produzca un evento en el dispositivo se creará una instancia de la clase

anteriormente descrita y se insertará en el razonador. Para crear una instancia se utiliza la siguiente plantilla que se rellanará con los datos correspondiente al dispositivo en se momento.

```
<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">

  <!--DimmableLight 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>
      <DimmableLight rdf:ID="InsertDimmableLightIndividualName">
        <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"/>
      </DimmableLight>
    </containsDeviceItem>
  </SpatialPoint>
</rdf:RDF>
```

Además en este caso el dispositivo registra un regla que gestionará el control de las luces en función de la existencia o no de personas en la zona en la que se encuentra. La regla que inserta la luz en el motor es la siguiente:

```
[EVENT-switchOnLightsColocatedWithPerson:
      (?light rdf:type <http://www.morelab.deusto.es/smartlab.owl#LightItem>)
      (?light <http://www.morelab.deusto.es/smartlab.owl#colocatedWith> ?person),
      (?person rdf:type <http://www.morelab.deusto.es/smartlab.owl#PersonItem>),
      (?light <http://www.morelab.deusto.es/smartlab.owl#placedIn> ?placedIn),
      makeTemp(?lightOnEvent)
      ->
      (?lightOnEvent rdf:type <http://www.morelab.deusto.es/smartlab.owl#LightOnEvent>),
      (?lightOnEvent <http://www.morelab.deusto.es/smartlab.owl#subject> ?light),
      (?lightOnEvent <http://www.morelab.deusto.es/smartlab.owl#place> ?placedIn)]
```