Make IoT Child’s play

Gamifying IoT with Vorto & Kura

Alexander Edelmann, Vorto Committer
Luca Dazi, Kura Committer
Make IoT Child‘s Play - Overview

IoT Fever Game with Vorto & Kura

1. TI Sensor Tag
   - Describes
   - TI Sensor Tag

2. Measured Data
   - Reads
   - Information Model Repository
     - Information Model
       -TI Sensor Tag
     - Mapping
       -(Kura BLE)
       -TI Sensor Tag

3. Mosquito Server
   - Generates
   - Kura SensorTag OSGI Service Bundle
   - IoT Gateway Stack

Vorto Code Generator

Kura
IoT Gateways

Revolution: Towards Real-time Actionable Data

Make IoT Child’s Play - Kura
Eclipse Open IoT Stack for Java

Device Abstraction
- `javax.comm`
- `javax.usb w/ udev access`
- `java HID APIs`
- `GPS Position` (requires `jdk.dio`)
- `GPIO / SPI / I2C`
- `java.bluetooth / BLE`

Basic Gateway Services
- `DB Service`
- `Clock Service`
- `Watchdog`

Network Configuration
- `Cellular, Wi-Fi, Ethernet`
- `Firewall, Port Forwarding`
- `Link Monitors`

Connectivity and Delivery
- `Cloud Services`
- `Data Services`
- `MQTT Paho`

Field Protocols
- `Modbus`
- `CANBus`
- `Custom Protocols`

Remote Management
- Remote Access
- Configuration Management

Administration GUI

OSGi Application Container (Eclipse Equinox, Concierge)

Java SE 7 / 8 (OpenJDK)

Make IoT Child’s Play - Kura
Make IoT Child’s Play - Kura

Using Services

Acquire the relevant Declarative Service

```xml
<reference name="GPIOService"
  policy="static"
  bind="setGPIOService"
  unbind="unsetGPIOService"
  cardinality="1..1"
  interface="org.eclipse.kura_gpio.GPIOService"/>
```

Define Service members

```java
public void setGPIOService(GPIOService gpioService) {
  m_GPIOService = gpioService;
}

public void unsetGPIOService(GPIOService gpioService) {
  m_GPIOService = null;
}
```

Define Service hooks methods

```java
protected void activate(ComponentContext componentContext, Map<String, Object> properties) {
  s_logger.info("Activating Shoot A Pi...");

  try {
    m_CloudClient = m_CloudService.newCloudClient(APP_ID);
    m_CloudClient.addCloudClientListener(this);
  } catch (KuraException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }

doUpdate();
}
```

Use the Service

```java
// Publish the message
try {
  int messageId = m_CloudClient.publish(topic, payload, qos, retain);
  s_logger.info("Published to {} message: {} with ID: {}", new Object[] { topic, payload, messageId });
} catch (Exception e) {
  s_logger.error("Cannot publish topic: " + topic, e);
}
```
Define a metatype for Application-specific configurable data...

```xml
{name="Test Class" description="Example test class">

<!--
<Icon resource="" size="32"/>
-->

<AD id="test.property" name="test.property" type="Integer" cardinality="0" required="true" default="18" description="Sample property for test purposes"/>
```

And manage those data with the WebUI (Or remotely...)

![Kura WebUI screenshot]

---

**Make IoT Child’s Play - Kura**

**OSGi Metatype Definition**

Define a metatype for Application-specific configurable data...

```xml
{name="Test Class" description="Example test class">

<!--
<Icon resource="" size="32"/>
-->

<AD id="test.property" name="test.property" type="Integer" cardinality="0" required="true" default="18" description="Sample property for test purposes"/>
```

And manage those data with the WebUI (Or remotely...)

![Kura WebUI screenshot]
**Information model** – Abstract representation of functionalities and properties of the TI SensorTag.

**Functionalities**
- Accelerometer Sensor
- Temperature Sensor
- Humidity Sensor
- Pressure Sensor
- etc.

**Mapping**

**TI Sensor Tag Information Model**
- Functionblock: Accelerometer Sensor
- Functionblock: Temperature Sensor
- Functionblock: Humidity Sensor
- Functionblock: Pressure Sensor
- ...
- Device specific information

Information Models are created using the **IoT Toolset**.
The IoT Tool Set – realized as Eclipse plugins and provides a textual DSL editor

Overview of Vorto Models in the local workspace

View & Edit Vorto Models in a DSL Editor

Search & Download Vorto Models from the centralized Vorto Repository.
The Vorto DSL: An easy way to describe a device

```java
namespace examples.informationmodels.sensors
version 1.0.0
displayname "TI SensorTag CC2650"
description "Information model for the TI SensorTag CC2650."
category demo
using examples.functionblockmodels.sensors.Accelerometer ; 1.0.0
using examples.functionblockmodels.sensors.TemperatureSensor ; 1.0.0
using examples.functionblockmodels.sensors.HumiditySensor ; 1.0.0
using examples.functionblockmodels.sensors.PressureSensor ; 1.0.0
using examples.functionblockmodels.microphone ; 1.0.0
using examples.functionblockmodels.magnetometer ; 1.0.0
using examples.functionblockmodels.sensors.MagneticSensor ; 1.0.0
using examples.functionblockmodels.light ; 1.0.0
using examples.functionblockmodels.gyroscope ; 1.0.0

infomodel TI_SensorTag_CC2650 {
    functionblocks {
        accelerometer as Accelerometer "Function block representing the accelerometer of the device."
        temperatureSensor as TemperatureSensor "Function block representing the temperature sensor of the device."
        humiditySensor as HumiditySensor "Function block representing the humidity sensor of the device."
        pressureSensor as PressureSensor "Function block representing the pressure sensor of the device."
        microphone as Microphone "Function block representing the digital microphone of the device."
        magnetometer as Magnetometer "Function block representing the magnetometer of the device."
        magneticSensor as MagneticSensor "Function block representing the magnetic sensor of the device."
        light as Light "Function block representing the LED light of the device."
        gyroscope as Gyroscope "Function block representing the gyroscope of the device."
    }
}
```
Make IoT Child’s Play - Vorto

Vorto Model Repository – Manage & Share Information Models in a centralized repository.

Vorto IoT Tool Set

Vorto Repository

3rd Party Tool / App

vorto.eclipse.org/repo
A single platform for different stakeholders

Great, let me generate it for my platform.

Search & Download

“Ok, I will add the sensor and share it again”

Upload Model

Looks interesting. I want to stay informed

Create Watch & Receive Change Notifications

User A

“The temperature sensor is missing!”

Comment

User B

Make IoT Child’s Play - Vorto

Great, let me generate it for my platform.

Search & Download

“Ok, I will add the sensor and share it again”

Upload Model

Looks interesting. I want to stay informed

Create Watch & Receive Change Notifications

User A

“The temperature sensor is missing!”

Comment

User B

Make IoT Child’s Play - Vorto
Information Models can be transformed in various representations. Example Code Generators are:

- **iOS** Swift
- **Kura** Java, XML
- **Bosch** Java, XML, XSD

IoT Tool Set

Information Model Repository

Code Generator Extension Point

Make IoT Child’s Play - Vorto
Code Generation: Kura TI Sensor Tag OSGI Bundle
Vorto Toolset Architecture

Source Code available under https://github.com/eclipse/vorto
Make IoT Child’s Play

Putting it all together in a demo...

1. TI Sensor Tag

2. Kura SensorTag OSGi Service Bundle

3. Mosquito Server

Measured Data

Information Model
Repository

Vorto

Information Model
TI Sensor Tag

Mapping
(Kura BLE)
TI Sensor Tag

Describes

Generates

Reads

Generates

Describes

Kura Code Generator

Mosquito Server

Measured Data

IoT Gateway Stack