特別研究報告書

IoT 環境における状況依存型サービス連携の実現

指導教員　　石田 亨 教授

京都大学工学部情報学科

渡辺 隆弘

平成 28 年 2 月 6 日

IoT 環境における状況依存型サービス連携の実現

渡辺 隆弘

内容梗概

アブストラクト研究の背景と、概要

研究の貢献

1. Web とセンサーを繋ぐ画一化されたプラットフォームが存在しない
2. Web サービスの利用にその都度リクエストを送信しなければならない
3. サービス選択が手動

Realization of situated service composition in IoT environment

Takahiro Watanabe

Abstract

abstract

IoT 環境における状況依存型サービス連携の実現

目次

第1章　はじめに

はじめに

第2章　関連研究

この章では本研究に利用している各用語についての説明と，課題点について述べる．

2.1　IoT

IoT とは,

2.2　IoS

IoS とは,

2.3　CEP

CEP(Complex Event Processing，または複合イベント処理) とは，刻々と生成されるデータをリアルタイムに処理するための方式である．事前に定義したルールに，リアルタイムにデータを挿入し，そのルールに応じて即座に処理を行う．これまでのビッグデータ分析の方法は，データをデータベースに蓄積し，任意のタイミングで参照し，分析するという手法であったために，情報の処理に時間がかかるという問題点があった．CEP は対象のデータを直近の範囲に絞り，メモリ上に読みこんで処理を行うため処理を高速化でき，”直近の数秒以内に ”などの条件に沿ってデータを処理することが可能となる．本研究では，この CEP をストリーム形式であるセンサーデータに対し応用することを考える．

2.4　サービス連携

サービス連携とは，IoS 基盤に集積された各原子サービスを組み合わせ，ユーザの要求を満たす高い品質 (QoS，または Quality of Service) の複合サービスを構成する技術である．従来，複合サービスを構成するためには，ユーザが自らの要求を満足するような原子サービスを選択する方法が取られていた．また，複合サービスの自動構築を行う方法として，人工知能のプランニング技術を用いてワークフローを自動生成する研究が主流であった．しかし，IoS 環境において

は，同種の原子サービスが複数登録されるために，ワークフローを生成することよりむしろ，ワークフローに当てはめる原子サービスの選択が自動化できる必要がある．

第3章　提案手法

本章では，センサーのサービス化を行うための手法と，センサーから取得したデータによって，複合サービスのサービス選択，サービス実行を自動で行うための手法を提案する．

3.1　センサーのサービス化手法

本節では，センサーのサービス化手法を提案する．現状は，前述した通りセンサーの仕様が画一化されていないために，センサーを利用するシステムを実装する際，センサーの種類によって異なる実装が必要であるという問題点が存在する．この問題点を本提案は解決する．
データ定義を OpenIoT のセンサー定義に基づいて画一化する．OpenIoT のセンサー定義の例は以下である．データの値，取得時間や，温度，湿度，照度といったデータタイプを示す propertyType などが存在する．

センサー定義例

```
//Observation

private String id;
private Date times;
private String sensorId;
private String featureOfInterest="";
private ArrayList<ObservedProperty> readings;
private String metaGraph;
private String dataGraph;



//ObservedProperty

private static final long serialVersionUID = 1L;
private Object value;
private Date times;
private String propertyType;
private String unit;
private String observationId;
```

センサーの開発者は，センサーから値を取得した際に，Observation を作成し，各変数に取得した値を格納するようにサービスを構成する．ユーザはこのサービスの仕様に従ってシステムを実装することで，ユーザからは種々のセンサー間の違いは隠蔽され，画一化されたセンサーサービスとしてデータを利用することができる．例えば，センサーから温度 20 ℃，湿度 50%のデータを取得した際には以下のように Observation を生成する．

```
1 Observation o = new Observation();     //Observationオブジェクトの作成
2 ArrayList<ObservedProperty> readings = new ArrayList<ObservedProperty>();
      //ObservedPropertyのリストの作成
3 ObservedProperty tempProperty = new ObservedProperty();     //
      ObservedPropertyオブジェクトの作成
4 ObservedProperty humdProperty = new ObservedProperty();
5 tempProperty.setPropertyType("http://openiot.eu/ontology/ns/AirTemperature");
```

```
        //propertyTypeの設定
6  humdProperty.setPropertyType("http://openiot.eu/ontology/ns/AtmosphereHumidity"
       );
7  tempProperty.setValue(20);      //valueに値を格納
8  humdProperty.setValue(50);
9  readings.add(tempProperty);      //ObservedPropertyのリストに追加
10 readings.add(humdProperty);
11 o.setReadings(readings);;      //Observationに作成したリストを格納
```

3.2 状況依存型サービス選択手法

本節では，センサーの値によって複合サービス中の原子サービスを選択する手法を提案する．

3.2.1 データの取得

前節に基づいて作成されたセンサーサービスオブジェクトがサーバーへ送信される．サーバーはデータを受け取った時点で，このオブジェクトを CEP エンジンに挿入する．

3.2.2 原子サービスの選択

原子サービスの選択においては，ECA ルールを応用することを考える．ECA ルールとは，〜〜するものであり，

E ：Event

C ：Condition

A ：Action

の 3 つの状態が定義される．イベントが発生した際，その状況に応じてアクションを実行する，というルールの実行を行う．

本研究では，ECA ルールを CEP エンジンで実現する．つまり，ECA ルールを以下のように適用する．

E ：センサーからのデータの取得

C ：センサーから取得した値

A ：選択するサービスとサービスへの入力の生成，サービスの実行

以上から，センサーからデータを取得した際，サーバーから CEP エンジンにデータを挿入し，事前に定義されたルールに基づいて，選択するサービスとサービスへの入力の生成とサービスの実行を行うという一連の処理が実行される．また，本研究では CEP エンジンにおいて適用するルールは事前に定義されているものとし，状況に応じてどのような処理を実行すべきかというルールの構成の

点についての議論は行わない．

3.2.3 貢献

この手法により以下の 2 点の問題点が解決される．

1. 複合サービス内の原子サービスの選択
   これまで，複合サービス内の原子サービスの選択は，ユーザによって指定する方向で行われてきた．例えば，言語グリッドの翻訳サービスを利用する際，翻訳エンジンとして yahoo 翻訳をつまり，原子サービスの選択にユーザの知識や経験が要求されるため，以下のような問題点が生じる．
   - ユーザが初めて複合サービスを利用する際にどのような原子サービスを利用すれば適当かが分からない
   - ユーザのサービスに対しての知識が不足しているために，ユーザのサービス選択がユーザの要求に関わらず固定化されてしまい，ユーザの要求を満たすよりよい原子サービスの組み合わせがあるにもかかわらず，より質の低いサービス選択を行ってしまう

   一方，本提案では，専門家が一度ルールを作成すれば，センサーの値によって分岐するルールに従って原子サービスの選択を行うことができ，サービス連携においてユーザのサービスに対しての知識や経験に関わらず一定の質の高いサービス合成が可能となる．
2. 複合サービスのリアルタイム実行
   複合サービスは，複数の Web サービスを組み合わせたものであるため，実行の仕様は Web サービスに基づく．Web サービスはリクエストに応じてレスポンスを返す形式であるため，Web サービスを利用するためには，ユーザは Web サービスにリクエストを送信する必要がある．本提案では，センサーの値をイベントとして CEP エンジンに挿入し，リアルタイムで処理，アクションとして複合サービスへの入力生成とサービス実行を行うことによって，ユーザがサービスのリクエストを送信することなく，リアルタイムかつ自動的なサービス実行が可能となる．

第 4 章　提案アーキテクチャ

本章では，前章に説明した提案手法に基づいて，IoT 環境下で複合サービスの選択，実行を行うアーキテクチャの提案を行う．

4.1

第5章　実装

本章では，前章に提案したアーキテクチャの実装について説明し，動作確認と評価について述べる．最後に実装の結果に対して考察を行う．

5.1　シチュエーション

体育館を利用するユーザに，温度，湿度などの情報から運動への助言を音声で与えるシステムを実装することを考える．ユーザは様々な言語圏のユーザが想定されるため，それぞれのユーザが利用する言語に基づいてアナウンスを行う必要がある．

5.2　仕様

言語は Java を用いて実装した．以下に各モジュールの詳細を述べる．

5.2.1　センサーデバイス

体育館に設置することを想定するセンサーデバイスは，(株) オムロンの環境センサー”URL”とする．このセンサーによって取得できるデータタイプの中から，今回は温度データと湿度データを利用する．

5.2.2　ユーザデバイス

ユーザが所持している端末を，iOS を搭載した端末とした．この端末はセンサーデバイスからデータを取得し，Observation を形成してサーバーへ送信するデバイスとして働く．センサーデバイスと本デバイス間の通信は BLE(Bluetooth Low Energy) を使用する．BLE は省電力の無線通信技術であり.....
構成要素は以下．

- WaikikiSensor
  取得したデータを端末上に表示する．
- Myviewcontrollor
  データを取得し，センサーデータオブジェクトを構成してサーバに送信する．オブジェクトの構成法は 3.1 節で説明した方法に基づく．
  1. Observation オブジェクトを生成する．
  2. ObservedProperty として tempProperty，humdProperty を作成する．それぞれ，温度のデータ，湿度のデータを格納するオブジェクトである

3. ObservedProperty それぞれに，データタイプを示す PropertyType とデータの値を格納する．
4. tempProperty と humdProperty を Observation オブジェクトに格納する．

5.2.3 サーバー

サーバーと周辺のモジュールについて説明する．
以下のモジュールからなる．

- サーバー
  サーバーは ObservationReceiver クラスとして実装される．デバイスから Observation オブジェクトが送信された際に，CEP エンジンにでデータをイベントとして挿入する．
- CEP エンジン
  CEP エンジンとして，Java で実装されたイベントエンジンである Drools を利用する．ルールとして”badminton.drl”を実装した．ルールの概要を以下に挙げる．表
- 複合サービス
  複合サービスとして，言語グリッドを利用する．言語グリッドは，登録された言語サービスを自由に組み合わせて新しい言語サービスを生み出す複合サービスである．本研究では，言語グリッドに登録されているサービスの中から，形態素解析サービスと辞書翻訳サービス，音声合成サービスを利用する．仕様は ”URL ”
  - 形態素解析
  - 辞書翻訳
  - 音声合成

5.3　動作確認
5.4　考察
第6章　終わりに

謝辞

本研究を行うにあたり, 貴重な資料をご提供いただきました株式会社オムロン様に深く感謝申し上げます. そして本研究を行うにあたり, 熱心なご指導, ご助言を賜りました石田亨教授に厚く御礼申し上げます. また, 日頃より数々のご助言をいただきました中口孝雄特定研究員, 林冬惠助教をはじめ, 石田・松原研究室の皆様方に心より感謝いたします.

付録

実装のソースコードを添付する.

A.1 デバイスでセンサーデータを取得し、サーバーへ送信するモジュールのソースコード

A.1.1 MyviewController.java

```
1  package org.langrid.waikiki.sensor;
2
3  import java.net.MalformedURLException;
4  import java.net.URL;
5  import java.util.ArrayList;
6
7  import org.langrid.waikiki.sensor.omron.EnvSensor;
8  import org.langrid.waikiki.sensor.omron.EnvSensorScanner;
9  import org.langrid.waikikiws.service.ObservationReceiverImpl;
10 import org.langrid.waikikiws.service.api.ObservationReceiver;
11 import org.openiot.lsm.beans.Observation;
12 import org.openiot.lsm.beans.ObservedProperty;
13 import org.robovm.apple.coregraphics.CGRect;
14 import org.robovm.apple.foundation.NSBundle;
15 import org.robovm.apple.foundation.NSURL;
16 import org.robovm.apple.uikit.UIColor;
17 import org.robovm.apple.uikit.UIView;
18 import org.robovm.apple.uikit.UIViewController;
19 import org.robovm.apple.webkit.WKScriptMessage;
20 import org.robovm.apple.webkit.WKScriptMessageHandlerAdapter;
21 import org.robovm.apple.webkit.WKUserContentController;
22 import org.robovm.apple.webkit.WKWebView;
23 import org.robovm.apple.webkit.WKWebViewConfiguration;
24
25
26 import jp.go.nict.langrid.client.jsonrpc.JsonRpcClientFactory;
27 import net.arnx.jsonic.JSON;
28
29 public class MyViewController extends UIViewController {
30     public MyViewController() {
31         // Get the view of this view controller.
32         UIView view = getView();
33
34         // Setup background.
35         view.setBackgroundColor(UIColor.white());
36
37         WKUserContentController controller = new WKUserContentController();
38         controller.addScriptMessageHandler(new WKScriptMessageHandlerAdapter()
               {
39             @Override
40             public void didReceiveScriptMessage(WKUserContentController c,
                   WKScriptMessage message) {
```

```
41                  System.out.println("message: " + message.getBody());
42                  if(message.getName().equals("handler")){
43                      if(message.getBody().toString().equals("startScan"))
44                          startScan();
45                      if(message.getBody().toString().equals("stopScan"))
46                          stopScan();
47                  }
48              }
49          }, "handler");
50          WKWebViewConfiguration config = new WKWebViewConfiguration();
51          config.setUserContentController(controller);
52          CGRect frame = view.getFrame();
53          wv = new WKWebView(
54                  new CGRect(frame.getMinX(), frame.getMinY() + 16,
55                          frame.getWidth(), frame.getHeight() - 16),
56                  config);
57          view.addSubview(wv);
58          NSURL bu = NSBundle.getMainBundle().getBundleURL();
59          wv.loadFileURL(new NSURL(bu.toString() + "index.html"), bu);
60
61          try {
62              client = new JsonRpcClientFactory().create(
63                      ObservationReceiver.class,
64  //大学                    new URL("http://10.229.250.104:8080/waikikiws/services/
        ObservationReceiver")
65                      new URL("http://192.168.11.2:8080/waikikiws/services/
                            ObservationReceiver")  //家
66                      );
67          } catch (MalformedURLException e) {
68              e.printStackTrace();
69          }
70      }
71
72      private void startScan(){
73          scanner.startScan(s -> {
74              System.out.println(JSON.encode(s).toString());
75              wv.evaluateJavaScript("found(" + JSON.encode(s) + ");", null);
76              // 送信
77              client.notify(createObservation(s)); //s = {"brightness":-112,....}
78          });
79      }
80
81      private void stopScan(){
82          scanner.stopScan();
83      }
84  //s.get~で要素の値を取り出してObservationを生成
85      private Observation createObservation(EnvSensor s){
86          String TEMPERATURE = "http://openiot.eu/ontology/ns/AirTemperature";
87          String HUMIDITY = "http://openiot.eu/ontology/ns/AtmosphereHumidity";
88
89          Observation o = new Observation();
90          ArrayList<ObservedProperty> readings = new ArrayList<ObservedProperty
                >();
```

```
91          ObservedProperty tempProperty = new ObservedProperty();
92          ObservedProperty humdProperty = new ObservedProperty();
93
94          double temperature = s.getTemperature()/100;
95          double humidity = s.getHumidity()/100;
96
97          tempProperty.setPropertyType(TEMPERATURE);
98          humdProperty.setPropertyType(HUMIDITY);
99          tempProperty.setValue(temperature);
100         humdProperty.setValue(humidity);
101         readings.add(tempProperty);
102         readings.add(humdProperty);
103         o.setReadings(readings);
104         return o;
105     }
106
107     private ObservationReceiver client;
108     private EnvSensorScanner scanner = new EnvSensorScanner();
109     private final WKWebView wv;
110 }
```

A.1.2 WaikikiSensor.java

```
1  package org.langrid.waikiki.sensor;
2
3  import org.robovm.apple.foundation.NSAutoreleasePool;
4  import org.robovm.apple.uikit.UIApplication;
5  import org.robovm.apple.uikit.UIApplicationDelegateAdapter;
6  import org.robovm.apple.uikit.UIApplicationLaunchOptions;
7  import org.robovm.apple.uikit.UIScreen;
8  import org.robovm.apple.uikit.UIWindow;
9
10 public class WaikikiSensor extends UIApplicationDelegateAdapter {
11     private UIWindow window;
12     private MyViewController rootViewController;
13
14     @Override
15     public boolean didFinishLaunching(UIApplication application,
           UIApplicationLaunchOptions launchOptions) {
16         // Set up the view controller.
17         rootViewController = new MyViewController();
18
19         // Create a new window at screen size.
20         window = new UIWindow(UIScreen.getMainScreen().getBounds());
21         // Set the view controller as the root controller for the window.
22         window.setRootViewController(rootViewController);
23         // Make the window visible.
24         window.makeKeyAndVisible();
25
26         return true;
27     }
28
29     public static void main(String[] args) {
```

```
30          try (NSAutoreleasePool pool = new NSAutoreleasePool()) {
31              UIApplication.main(args, null, WaikikiSensor.class);
32          }
33      }
34  }
```

A.2 受信したデータをルールエンジンに挿入し、状況に応じた出力を得るモジュールのソースコード

A.2.1 ObservationReceiverImpl.java

```
1  package org.langrid.waikikiws.service;
2
3  import java.util.ArrayList;
4
5  import org.langrid.waikikiws.DroolsManager;
6  import org.langrid.waikikiws.service.api.ObservationReceiver;
7  import org.langrid.waikikiws.service.api.ObservationReceiverDebug;
8  import org.openiot.lsm.beans.Observation;
9  import org.openiot.lsm.beans.ObservedProperty;
10 import org.langrid.waikikiws.VoiceText;
11 import org.langrid.waikikiws.service.TargetLanguage;
12
13 public class ObservationReceiverImpl
14 implements ObservationReceiver, ObservationReceiverDebug{
15     @Override
16     public void notify(Observation o){
17         System.out.println("------------------");
18         System.out.println(o.getReadings().get(0).getValue());
19         System.out.println("------------------");        //ここまでは通ってる
20         DroolsManager.getSession().insert(o); //
               Observationをルールエンジンへ挿入する
21     }
22
23     public static String TEMPERATURE = "http://openiot.eu/ontology/ns/
           AirTemperature";
24     public static String HUMIDITY = "http://openiot.eu/ontology/ns/
           AtmosphereHumidity";
25     @Override
26     public void dummyNotify(double temperature, double humidity,String
           tlanguage) {
27
28         Observation o = new Observation();
29         ArrayList<ObservedProperty> readings = new ArrayList<ObservedProperty
               >();
30         ObservedProperty tempProperty = new ObservedProperty();
31         ObservedProperty humdProperty = new ObservedProperty();
32         tempProperty.setPropertyType(TEMPERATURE);
33         humdProperty.setPropertyType(HUMIDITY);
34         tempProperty.setValue(temperature);
35         humdProperty.setValue(humidity);
```

```
36          readings.add(tempProperty);
37          readings.add(humdProperty);
38          o.setReadings(readings);;
39          DroolsManager.getSession().insert(o);
40          DroolsManager.getSession().insert(new TargetLanguage(tlanguage));
41      }
42
43  }
```

A.2.2 Translator.java

```
 1  package org.langrid.waikikiws.service;
 2
 3  import java.io.IOException;
 4  import java.io.InputStream;
 5  import java.io.InputStreamReader;
 6  import java.io.Reader;
 7  import java.net.MalformedURLException;
 8  import java.net.URL;
 9  import java.util.Arrays;
10  import java.util.List;
11  import java.util.Properties;
12
13  import org.langrid.waikikiws.Bindings;
14  import org.langrid.waikikiws.service.api.TranslatorService;
15
16  import jp.go.nict.langrid.client.RequestAttributes;
17  import jp.go.nict.langrid.client.soap.SoapClientFactory;
18  import jp.go.nict.langrid.commons.cs.binding.BindingNode;
19  import jp.go.nict.langrid.service_1_2.AccessLimitExceededException;
20  import jp.go.nict.langrid.service_1_2.InvalidParameterException;
21  import jp.go.nict.langrid.service_1_2.NoAccessPermissionException;
22  import jp.go.nict.langrid.service_1_2.NoValidEndpointsException;
23  import jp.go.nict.langrid.service_1_2.ProcessFailedException;
24  import jp.go.nict.langrid.service_1_2.ServerBusyException;
25  import jp.go.nict.langrid.service_1_2.ServiceNotActiveException;
26  import jp.go.nict.langrid.service_1_2.ServiceNotFoundException;
27  import jp.go.nict.langrid.service_1_2.translation.TranslationService;
28
29  public class Translator implements TranslatorService{
30      public Translator() throws IOException {
31          Properties p = new Properties();
32          try(InputStream is = getClass().getResourceAsStream("/langrid.
                properties");
33                  Reader r = new InputStreamReader(is, "UTF-8")){
34              p.load(r);
35          }
36          this.url = p.getProperty("url");
37          this.userId = p.getProperty("userId");
38          this.password = p.getProperty("password");
39      }
40      @Override
```

```
41      public String translate(String sourceLang, String targetLang, String source
            ) {
42          List<BindingNode> bindings = Arrays.asList(
43                  new BindingNode("MorphologicalAnalysisPL", "Mecab"),
44                  new BindingNode("TranslationPL", "KyotoUJServer")
45                  );
46  //      List<BindingNode> bindings = Bindings.getBindings();
47          try {
48              TranslationService trans = new SoapClientFactory().create(
49                      TranslationService.class,
50                      new URL(url + "
                            TranslationCombinedWithBilingualDictionaryWithLongestMatchSearch
                            "),
51                      userId, password
52                      );
53              for(BindingNode n : bindings){
54                  ((RequestAttributes)trans).getTreeBindings().add(n);
55              }
56              return trans.translate(sourceLang, targetLang, source);
57          } catch (MalformedURLException | AccessLimitExceededException |
                InvalidParameterException | NoAccessPermissionException |
                ProcessFailedException | NoValidEndpointsException |
                ServerBusyException | ServiceNotActiveException |
                ServiceNotFoundException e) {
58              throw new RuntimeException(e);
59          }
60      }
61
62      private String url;
63      private String userId;
64      private String password;
65  }
```

A.2.3 Binding.java

```
1   package org.langrid.waikikiws;
2
3   import java.util.Arrays;
4   import java.util.List;
5
6   import jp.go.nict.langrid.commons.cs.binding.BindingNode;
7
8   public class Bindings {
9       public static List<BindingNode> getBindings() {
10          return bindings;
11      }
12      public static void binding1(){
13          bindings = Arrays.asList(
14                  new BindingNode("MorphologicalAnalysisPL", "Mecab"),
15                  new BindingNode("TranslationPL", "KyotoUJServer")
16                  );
17          //System.out.println("binding1");
18      }
```

```
19      public static void binding2(){
20          bindings = Arrays.asList(
21                  new BindingNode("MorphologicalAnalysisPL", "Mecab"),
22                  new BindingNode("BilingualDictionaryWithLongestMatchSearchPL",
                        "KyotoTourismDictionaryDb"),
23                  new BindingNode("TranslationPL", "KyotoUJServer")
24                  );
25          //System.out.println("binding2");
26      }
27      public static void setBindings(List<BindingNode> bindings) {
28          Bindings.bindings = bindings;
29      }
30      private static List<BindingNode> bindings = Arrays.asList(
31              new BindingNode("MorphologicalAnalysisPL", "Mecab"),
32              new BindingNode("TranslationPL", "KyotoUJServer")
33              );
34  }
```

A.2.4 DroolsManager.java

```
1   package org.langrid.waikikiws;
2
3   import java.io.IOException;
4
5   import org.kie.api.runtime.KieSession;
6
7   public class DroolsManager {
8       public static synchronized KieSession getSession(){
9           if(session == null){
10              try {
11                  session = DroolsUtil.createStreamSessionFromResource("/
                        badminton.drl");
12              } catch (IOException e) {
13                  throw new RuntimeException(e);
14              }
15              Thread t = new Thread(() -> {
16                  session.fireUntilHalt();
17              });
18              t.setDaemon(true);
19              t.start();
20              org.kie.api.runtime.rule.FactHandle.State.class.getName();
21          }
22          return session;
23      }
24
25      private static KieSession session;
26  }
```

A.2.5 DroolsUtil.java

```
1   package org.langrid.waikikiws;
2
3   import java.io.IOException;
```

```
 4 import java.io.InputStream;
 5
 6 import org.kie.api.KieBase;
 7 import org.kie.api.KieBaseConfiguration;
 8 import org.kie.api.KieServices;
 9 import org.kie.api.builder.KieBuilder;
10 import org.kie.api.builder.KieFileSystem;
11 import org.kie.api.builder.Message;
12 import org.kie.api.builder.Results;
13 import org.kie.api.conf.EventProcessingOption;
14 import org.kie.api.runtime.KieContainer;
15 import org.kie.api.runtime.KieSession;
16
17 public class DroolsUtil {
18     public static KieSession createSessionFromResource(Package pkg, String
          rulePath) throws IOException{
19         return createSessionFromResource(
20                 "/" + pkg.getName().replaceAll("\\.", "/") + "/" + rulePath);
21     }
22
23     public static KieSession createSessionFromResource(String rulePath)
24     throws IOException{
25         KieServices kieServices = KieServices.Factory.get();
26         KieFileSystem kfs = kieServices.newKieFileSystem();
27         try(InputStream is = DroolsUtil.class.getResourceAsStream(rulePath)){
28             // for each DRL file, referenced by a plain old path name:
29             kfs.write("src/main/resources" + rulePath,
30                     kieServices.getResources().newInputStreamResource(is));
31             KieBuilder kieBuilder = kieServices.newKieBuilder( kfs ).buildAll
                   ();
32             Results results = kieBuilder.getResults();
33             if( results.hasMessages( Message.Level.ERROR ) ){
34                 System.out.println( results.getMessages() );
35                 throw new RuntimeException("### errors ###" );
36             }
37             KieContainer kieContainer = kieServices.newKieContainer(
38                     kieServices.getRepository().getDefaultReleaseId() );
39             KieBase kieBase = kieContainer.getKieBase();
40             return kieBase.newKieSession();
41         }
42     }
43
44     public static KieSession createStreamSessionFromResource(Package pkg,
          String rulePath) throws IOException{
45         return createStreamSessionFromResource(
46                 "/" + pkg.getName().replaceAll("\\.", "/") + "/" + rulePath);
47     }
48     public static KieSession createStreamSessionFromResource(String rulePath)
49     throws IOException{
50         KieServices kieServices = KieServices.Factory.get();
51         KieFileSystem kfs = kieServices.newKieFileSystem();
52         try(InputStream is = DroolsUtil.class.getResourceAsStream(rulePath)){
53             // for each DRL file, referenced by a plain old path name:
```

```
54              kfs.write("src/main/resources" + rulePath,
55                      kieServices.getResources().newInputStreamResource(is));
56              KieBuilder kieBuilder = kieServices.newKieBuilder( kfs ).buildAll
                    ();
57              Results results = kieBuilder.getResults();
58              if( results.hasMessages( Message.Level.ERROR ) ){
59                  System.out.println( results.getMessages() );
60                  throw new RuntimeException("### errors ###" );
61              }
62              KieContainer kieContainer = kieServices.newKieContainer(
63                      kieServices.getRepository().getDefaultReleaseId() );
64              KieBaseConfiguration config = KieServices.Factory.get().
                    newKieBaseConfiguration();
65              config.setOption( EventProcessingOption.STREAM );
66              KieBase kieBase = kieContainer.newKieBase(config);
67              return kieBase.newKieSession();
68          }
69      }
70  }
```

A.2.6 TargetLanguage

```
1  package org.langrid.waikikiws.service;
2
3
4  public class TargetLanguage {
5      public TargetLanguage(){
6      }
7
8      public TargetLanguage(String targetlang) {
9          super();
10         this.targetlanguage = targetlang;
11     }
12
13     public String getTargetlang() {
14         return targetlanguage;
15     }
16
17     public void setTargetlang(String targetlang){
18         this.targetlanguage = targetlang;
19     }
20
21     private String targetlanguage;
22
23 }
```

A.2.7 VoiceText.java

```
1  package org.langrid.waikikiws;
2
3  import java.net.URL;
4  import jp.go.nict.langrid.client.soap.SoapClientFactory;
5  import jp.go.nict.langrid.service_1_2.speech.Speech;
```

```
6  import jp.go.nict.langrid.service_1_2.speech.TextToSpeechService;
7  import javax.sound.sampled.*;
8
9
10 import java.io.*;
11
12 public class VoiceText {
13     public void voicetext(String text,String lang) throws Exception {
14
15     // TODO 自動生成されたメソッド・スタブ
16         TextToSpeechService c =
17                 new SoapClientFactory().create(
18                         TextToSpeechService.class,
19                         new URL("http://langrid.org/service_manager/invoker/
                               kyoto1.langrid:VoiceText"),
20                         "ishida.kyoto-u", "tWJaakYm");
21         Speech s = c.speak(lang, text, "woman","audio/x-wav");
22
23         byte[] buf = s.getAudio();
24         ByteArrayInputStream stream = new ByteArrayInputStream(buf);
25         AudioInputStream ais = AudioSystem.getAudioInputStream(stream);
26         byte[] data = new byte [ais.available()];
27         ais.read(data);
28         ais.close();
29         AudioFormat af = ais.getFormat();
30         DataLine.Info info = new DataLine.Info(SourceDataLine.class, af);
31         SourceDataLine line = (SourceDataLine)AudioSystem.getLine(info);
32         line.open();
33         line.start();
34         line.write(buf,0,buf.length);
35         line.drain();
36         line.close();
37     }
38 }
```

A.2.8 TransTextToSpeech.java

```
1  package org.langrid.waikikiws;
2
3  import org.langrid.waikikiws.VoiceText;
4  import org.langrid.waikikiws.service.Translator;
5
6  public class TransTextToSpeech {
7
8      public void transtexttospeech(String text,int i) throws Exception{
9          Translator trans = new Translator();
10         VoiceText tts = new VoiceText();
11         String lang;
12         if (i == 0) {
13             lang = "en";
14         }
15         else{
16             lang = "zh-CN";
```

```
17          }
18          String transtext = trans.translate("ja",lang,text);
19          tts.voicetext(transtext,lang);
20      }
21  }
```

A.2.9 badminton.drl

```
1  import org.openiot.lsm.beans.Observation;
2  import org.openiot.lsm.beans.ObservedProperty;
3  import org.langrid.waikikiws.TransTextToSpeech;
4  import org.langrid.waikikiws.service.TargetLanguage;
5  import java.util.ArrayList;
6
7
8  function void TTTS(String text,int t){
9    TransTextToSpeech ttts = new TransTextToSpeech();
10   ttts.transtexttospeech(text,t);
11 }
12
13 /*
14 rule "test"
15 when
16   $o: Observation()
17   $op1: ObservedProperty(
18     propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
19     )
20     from $o.readings
21   $op2: ObservedProperty(
22     propertyType == "http://openiot.eu/ontology/ns/AtmosphereHumidity"
23     )
24     from $o.readings
25   $tl: TargetLanguage(
26 //     targetlanguage == "en"
27     )
28 then
29 //  System.out.println("ok-------");
30 //  System.out.println($tl.getTargetlang());
31 //  TransTextToSpeech ttts = new TransTextToSpeech();
32 //  ttts.transtexttospeech("おはようございます",Integer.parseInt($op2.getValue
       ().toString()) $tl.getTargetlang());
33 end
34 */
35
36 rule "WBGTcalc1"
37 when
38   $o: Observation()
39   $op1: ObservedProperty(
40     propertyType == "http://openiot.eu/ontology/ns/AirTemperature"
41     )
42     from $o.readings
43   $op2: ObservedProperty(
44     value > 80 &&
```

```
45      propertyType == "http://openiot.eu/ontology/ns/AtmosphereHumidity"
46      )
47      from $o.readings
48    $t: TargetLanguage()
49  then
50      double tmp = Double.parseDouble($op2.getValue().toString());
51      double hmd = Double.parseDouble($op2.getValue().toString());
52      double WBGT = tmp + (hmd - 80) / 5;
53      Observation o = new Observation();
54      ArrayList<ObservedProperty> readings = new ArrayList<ObservedProperty>();
55      ObservedProperty wbgtProperty = new ObservedProperty();
56      ObservedProperty tlangProperty = new ObservedProperty();
57      wbgtProperty.setPropertyType("http://ishida.kyoto-u/watanabe/
            WetBulbGlobTemperature");
58      tlangProperty.setPropertyType("http://ishida.kyoto-u/watanabe/
            TargetTransLanguage");
59      wbgtProperty.setValue(WBGT);
60      String st =  $t.getTargetlang();
61      if(st.equals("en")){
62          tlangProperty.setValue(0);
63      }else if (st.equals("zh-CN")) {
64          tlangProperty.setValue(1);
65      }
66      readings.add(wbgtProperty);
67      o.setReadings(readings);
68      insert(o);
69  end
70
71  rule "WBGTcalc2"
72  when
73    $o: Observation()
74    $op1: ObservedProperty(
75      propertyType == "http://openiot.eu/ontology/ns/AirTemperature"
76      )
77      from $o.readings
78    $op2: ObservedProperty(
79      value <= 80 &&
80      propertyType == "http://openiot.eu/ontology/ns/AtmosphereHumidity"
81      )
82      from $o.readings
83    $t: TargetLanguage()
84  then
85      double tmp = Double.parseDouble($op2.getValue().toString());
86      double hmd = Double.parseDouble($op2.getValue().toString());
87      double WBGT = tmp - (80 - hmd) / 5;
88      Observation o = new Observation();
89      ArrayList<ObservedProperty> readings = new ArrayList<ObservedProperty>();
90      ObservedProperty wbgtProperty = new ObservedProperty();
91      ObservedProperty tlangProperty = new ObservedProperty();
92      wbgtProperty.setPropertyType("http://ishida.kyoto-u/watanabe/
            WetBulbGlobTemperature");
93      tlangProperty.setPropertyType("http://ishida.kyoto-u/watanabe/
            TargetTransLanguage");
```

```
94      wbgtProperty.setValue(WBGT);
95      String st =  $t.getTargetlang();
96      if(st.equals("en")){
97          tlangProperty.setValue(0);
98      }else if (st.equals("zh-CN")) {
99          tlangProperty.setValue(1);
100     }
101     readings.add(wbgtProperty);
102     readings.add(tlangProperty);
103     o.setReadings(readings);
104     insert(o);
105 end
106
107 /*
108 rule "test2"
109 when
110   $o: Observation()
111   $op1: ObservedProperty(
112     propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
113     )
114     from $o.readings
115   $op2: ObservedProperty(
116     value <= 80 &&
117     propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
118     )
119     from $o.readings
120 then
121     TTTS("こんばんは",Integer.parseInt($op1.getValue().toString()));
122 end
123 */
124
125 rule "phase5"  //WBGT>=31
126 when
127   $o: Observation()
128   $op1: ObservedProperty(
129     value >= 31 &&
130     propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
131     )
132     from $o.readings
133   $op2: ObservedProperty(
134     propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
135     )
136     from $o.readings
137 then
138     System.out.println("運動を中止しましょう.");
139     TTTS("運動を中止しましょう.",Integer.parseInt($op2.getValue().toString()));
140 end
141
142 rule "phase4"  //28<=WBGT<31
143 when
144   $o: Observation()
145   $op1: ObservedProperty(
146     value < 31 && value >= 28 &&
```

```
147       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
148       )
149       from $o.readings
150     $op2: ObservedProperty(
151       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
152       )
153       from $o.readings
154   then
155       System.out.println("激しい運動は避け、積極的に休息と水分補給を行いましょう.");
156       TTTS("激しい運動は避け、積極的に休息と水分補給を行いましょう
              .",Integer.parseInt($op2.getValue().toString()));
157   end
158
159   rule "phase3"   //25<=WBGT<28
160   when
161     $o: Observation()
162     $op1: ObservedProperty(
163       value < 28 && value >= 25 &&
164       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
165       )
166       from $o.readings
167     $op2: ObservedProperty(
168       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
169       )
170       from $o.readings
171   then
172       System.out.println("激しい運動を行う際は,30分おきくらいに休息をとりましょう.");
173       TTTS("激しい運動を行う際は,30分おきくらいに休息をとりましょう.",Integer.parseInt
              ($op2.getValue().toString()));
174   end
175
176   rule "phase2"   //21<=WBGT<25
177   when
178     $o: Observation()
179     $op1: ObservedProperty(
180       value < 25 && value >= 21 &&
181       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
182       )
183       from $o.readings
184     $op2: ObservedProperty(
185       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
186       )
187       from $o.readings
188   then
189       System.out.println("水分補給には十分気をつけましょう.");
190       TTTS("水分補給には十分気をつけましょう
              .",Integer.parseInt($op2.getValue().toString()));
191   end
192
193   rule "phase1"    //WBGT<21
194   when
195     $o: Observation()
196     $op1: ObservedProperty(
```

```
197       value < 21 &&
198       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
199       )
200       from $o.readings
201     $op2: ObservedProperty(
202       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
203       )
204       from $o.readings
205   then
206       System.out.println("熱中症の危険は少ないですが,適宜水分補給をしましょう.");
207       TTTS("熱中症の危険は少ないですが,適宜水分補給をしましょう.",Integer.parseInt($op2
              .getValue().toString()));
208   end
209
210   rule "floor"    //湿度で床が滑る
211   when
212     $o: Observation()
213     $h: ObservedProperty(
214       value >= 90 &&
215       propertyType == "http://openiot.eu/ontology/ns/AtmosphereHumidity"
216       )
217       from $o.readings
218     $op2: ObservedProperty(
219       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
220       )
221       from $o.readings
222   then
223       System.out.println("湿度が高く床が滑りやすくなっています.気をつけましょう.");
224       TTTS("湿度が高く床が滑りやすくなっています
              .気をつけましょう.",Integer.parseInt($op2.getValue().toString()));
225   end
226
227   rule "shuttle1"    //1番シャトル
228   when
229     $o: Observation()
230     $op1: ObservedProperty(
231       value >= 33 &&
232       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
233       )
234       from $o.readings
235     $op2: ObservedProperty(
236       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
237       )
238       from $o.readings
239   then
240       System.out.println("1番のシャトルを使いましょう.");
241       TTTS("1番のシャトルを使いましょう.",Integer.parseInt($op2.getValue().
              toString()));
242   end
243
244   rule "shuttle2"    //2番シャトル
245   when
246     $o: Observation()
```

```
247     $op1: ObservedProperty(
248       value < 33 && value >= 27 &&
249       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
250       )
251       from $o.readings
252     $op2: ObservedProperty(
253       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
254       )
255       from $o.readings
256   then
257       System.out.println("2番のシャトルを使いましょう.");
258       TTTS("2番のシャトルを使いましょう.",Integer.parseInt($op2.getValue().
              toString()));
259   end
260
261   rule "shuttle3"    //3番シャトル
262   when
263     $o: Observation()
264     $op1: ObservedProperty(
265       value < 27 && value >= 22 &&
266       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
267       )
268       from $o.readings
269     $op2: ObservedProperty(
270       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
271       )
272       from $o.readings
273   then
274       System.out.println("3番のシャトルを使いましょう.");
275       TTTS("3番のシャトルを使いましょう.",Integer.parseInt($op2.getValue().
              toString()));
276   end
277
278   rule "shuttle4"    //4番シャトル
279   when
280     $o: Observation()
281     $op1: ObservedProperty(
282       value < 22 && value >= 17 &&
283       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
284       )
285       from $o.readings
286     $op2: ObservedProperty(
287       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
288       )
289       from $o.readings
290   then
291       System.out.println("4番のシャトルを使いましょう.");
292       TTTS("4番のシャトルを使いましょう.",Integer.parseInt($op2.getValue().
              toString()));
293   end
294
295   rule "shuttle5"    //5番シャトル
296   when
```

```
297     $o: Observation()
298     $op1: ObservedProperty(
299       value < 17 && value >= 12 &&
300       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
301       )
302       from $o.readings
303     $op2: ObservedProperty(
304       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
305       )
306       from $o.readings
307   then
308       System.out.println("5番のシャトルを使いましょう.");
309       TTTS("5番のシャトルを使いましょう.",Integer.parseInt($op2.getValue().
              toString()));
310   end
311
312   rule "shuttle6"    //6番シャトル
313   when
314     $o: Observation()
315     $op1: ObservedProperty(
316       value < 12 &&
317       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
318       )
319       from $o.readings
320     $op2: ObservedProperty(
321       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
322       )
323       from $o.readings
324   then
325       System.out.println("6番のシャトルを使いましょう.");
326       TTTS("6番のシャトルを使いましょう.",Integer.parseInt($op2.getValue().
              toString()));
327   end
328
329   rule "stringsh"    //温度が高いときのストリング
330   when
331     $o: Observation()
332     $op1: ObservedProperty(
333       value > 25 &&
334       propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
335       )
336       from $o.readings
337     $op2: ObservedProperty(
338       propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
339       )
340       from $o.readings
341   then
342       System.out.println("適正温度のときより+1ポンドのガットが適切です.");
343       TTTS("適正温度のときより+1ポンドのガットが適切です.",Integer.parseInt($op2.
              getValue().toString()));
344   end
345
346   rule "stringsn"    //適正温度のときのストリング
```

```
347 when
348   $o: Observation()
349   $op1: ObservedProperty(
350     value <= 25 && value >= 15 &&
351     propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
352     )
353     from $o.readings
354   $op2: ObservedProperty(
355     propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
356     )
357     from $o.readings
358 then
359     System.out.println("適正");
360     TTTS("ガットの適正温度です.",Integer.parseInt($op2.getValue().toString()));
361 end
362
363 rule "stringsl"    //温度が低いときのストリング
364 when
365   $o: Observation()
366   $op1: ObservedProperty(
367     value < 15 &&
368     propertyType == "http://ishida.kyoto-u/watanabe/WetBulbGlobTemperature"
369     )
370     from $o.readings
371   $op2: ObservedProperty(
372     propertyType == "http://ishida.kyoto-u/watanabe/TargetTransLanguage"
373     )
374     from $o.readings
375 then
376     System.out.println("適正温度のときより-1ポンドのガットが適切です.");
377     TTTS("適正温度のときより-1ポンドのガットが適切です.",Integer.parseInt($op2.
          getValue().toString()));
378 end
```

A.3 オムロンのセンサー定義

A.3.1 EnvSensor.java

```
1  package org.langrid.waikiki.sensor.omron;
2
3  public class EnvSensor {
4      public EnvSensor() {
5      }
6      //EnvSensor 11個の変数
7      public EnvSensor(
8              String uuid,
9              int lineNo, int temperature, int humidity, int brightness,
10             int uvIndex, int pressure, int noise,
11             int discomfortIndex, int heatstrokeIndex, int cellVoltage) {
12         this.uuid = uuid;
13         this.lineNo = lineNo;
14         this.temperature = temperature;
```

```
15          this.humidity = humidity;
16          this.brightness = brightness;
17          this.uvIndex = uvIndex;
18          this.pressure = pressure;
19          this.noise = noise;
20          this.discomfortIndex = discomfortIndex;
21          this.heatstrokeIndex = heatstrokeIndex;
22          this.cellVoltage = cellVoltage;
23      }
24  //行番号が1,その他が2バイトずつの19バイト
25      public static EnvSensor create(String uuid, byte[] bytes){
26          if(bytes.length != 19) throw new RuntimeException("The length of bytes
                must be 19");
27          return new EnvSensor(
28                  uuid,
29                  (int)bytes[0],
30                  bytes[1] + (bytes[2] << 8),
31                  bytes[3] + (bytes[4] << 8),
32                  bytes[5] + (bytes[6] << 8),
33                  bytes[7] + (bytes[8] << 8),
34                  bytes[9] + (bytes[10] << 8),
35                  bytes[11] + (bytes[12] << 8),
36                  bytes[13] + (bytes[14] << 8),
37                  bytes[15] + (bytes[16] << 8),
38                  bytes[17] + (bytes[18] << 8));
39      }
40
41      public String getUuid() {
42          return uuid;
43      }
44      public void setUuid(String uuid) {
45          this.uuid = uuid;
46      }
47      public int getLineNo() {
48          return lineNo;
49      }
50      public void setLineNo(int lineNo) {
51          this.lineNo = lineNo;
52      }
53      public int getTemperature() {
54          return temperature;
55      }
56      public void setTemperature(int temperature) {
57          this.temperature = temperature;
58      }
59      public int getHumidity() {
60          return humidity;
61      }
62      public void setHumidity(int humidity) {
63          this.humidity = humidity;
64      }
65      public int getBrightness() {
66          return brightness;
```

```
67      }
68      public void setBrightness(int brightness) {
69          this.brightness = brightness;
70      }
71      public int getUvIndex() {
72          return uvIndex;
73      }
74      public void setUvIndex(int uvIndex) {
75          this.uvIndex = uvIndex;
76      }
77      public int getPressure() {
78          return pressure;
79      }
80      public void setPressure(int pressure) {
81          this.pressure = pressure;
82      }
83      public int getNoise() {
84          return noise;
85      }
86      public void setNoise(int noise) {
87          this.noise = noise;
88      }
89      public int getDiscomfortIndex() {
90          return discomfortIndex;
91      }
92      public void setDiscomfortIndex(int discomfortIndex) {
93          this.discomfortIndex = discomfortIndex;
94      }
95      public int getHeatstrokeIndex() {
96          return heatstrokeIndex;
97      }
98      public void setHeatstrokeIndex(int heatstrokeIndex) {
99          this.heatstrokeIndex = heatstrokeIndex;
100     }
101     public int getCellVoltage() {
102         return cellVoltage;
103     }
104     public void setCellVoltage(int cellVoltage) {
105         this.cellVoltage = cellVoltage;
106     }
107
108     private String uuid;
109     private int lineNo; // ("行番号: " + bytes[0]);
110     private int temperature; // ("温度: " + (bytes[1] + (bytes[2] << 8)));
111     private int humidity; // ("相対湿度: " + (bytes[3] + (bytes[4] << 8)));
112     private int brightness; // ("照度: " + (bytes[5] + (bytes[6] << 8)));
113     private int uvIndex; // ("UVI: " + (bytes[7] + (bytes[8] << 8)));
114     private int pressure; // ("気圧: " + (bytes[9] + (bytes[10] << 8)));
115     private int noise; // ("騒音: " + (bytes[11] + (bytes[12] << 8)));
116     private int discomfortIndex; // ("不快指数: " + (bytes[13] + (bytes[14] <<
            8)));
117     private int heatstrokeIndex; // ("熱中症危険度: " + (bytes[15] + (bytes[16]
            << 8)));
```

```
118     private int cellVoltage; // ("電池電圧: " + (bytes[17] + (bytes[18] <<
            8)));
119 }
```

A.3.2 EnvSensorListener.java

```
1  package org.langrid.waikiki.sensor.omron;
2
3  public interface EnvSensorListener {
4      void onFound(EnvSensor sensor);
5  }
```

A.3.3 EnvSensorScanner.java

```
1  package org.langrid.waikiki.sensor.omron;
2
3  import java.util.LinkedHashSet;
4  import java.util.Set;
5
6  import org.robovm.apple.corebluetooth.CBAdvertisementData;
7  import org.robovm.apple.corebluetooth.CBCentralManager;
8  import org.robovm.apple.corebluetooth.CBCentralManagerDelegateAdapter;
9  import org.robovm.apple.corebluetooth.CBCharacteristic;
10 import org.robovm.apple.corebluetooth.CBPeripheral;
11 import org.robovm.apple.corebluetooth.CBPeripheralDelegateAdapter;
12 import org.robovm.apple.corebluetooth.CBService;
13 import org.robovm.apple.foundation.NSData;
14 import org.robovm.apple.foundation.NSError;
15 import org.robovm.apple.foundation.NSNumber;
16
17 import jp.go.nict.langrid.client.jsonrpc.JsonRpcClientFactory;
18 import jp.go.nict.langrid.repackaged.net.arnx.jsonic.JSON;
19
20 public class EnvSensorScanner {
21     private Set<CBPeripheral> peripherals;
22     private CBCentralManager central;
23     public void startScan(EnvSensorListener listener){
24         this.peripherals = new LinkedHashSet<>();
25         this.central = new CBCentralManager(
26                 new CBCentralManagerDelegateAdapter(){
27                     @Override
28                     public void didUpdateState(CBCentralManager central) {
29                         switch(central.getState().toString()){
30                             case "PoweredOn":
31                                 central.scanForPeripherals(null, null);
32                                 break;
33                         }
34                         super.didUpdateState(central);
35                     }
36                     @Override
37                     public void didDiscoverPeripheral(CBCentralManager central,
                            CBPeripheral peripheral, CBAdvertisementData
                            advertisementData, NSNumber rssi) {
```

```
38                      if(peripherals.contains(peripheral)) return;
39                      peripherals.add(peripheral);
40                      System.out.println(peripheral);
41                      if("EnvSensor-BL01".equals(peripheral.getName())){
42                          central.connectPeripheral(peripheral, null);
43                      }
44                  }
45                  @Override
46                  public void didConnectPeripheral(CBCentralManager central,
                        CBPeripheral peripheral) {
47                      peripheral.setDelegate(new CBPeripheralDelegateAdapter
                            (){
48                          @Override
49                          public void didDiscoverServices(CBPeripheral
                                peripheral, NSError error) {
50                              for(CBService s : peripheral.getServices()){
51                                  if(s.getUUID().toString().equals("0C4C3000
                                        -7700-46F4-AA96-D5E974E32A54")){
52                                      peripheral.discoverCharacteristics(null
                                            , s);
53                                  }
54                              }
55                          }
56                          @Override
57                          public void didDiscoverCharacteristics(CBPeripheral
                                 peripheral, CBService service,
58                                  NSError error) {
59                              for(CBCharacteristic c : service.
                                    getCharacteristics()){
60                                  if(c.getUUID().toString().equals("0C4C3001
                                        -7700-46F4-AA96-D5E974E32A54")){
61                                      peripheral.readValue(c);
62                                  }
63                              }
64                          }
65                          @Override
66                          public void didUpdateValue(CBPeripheral peripheral,
                                 CBCharacteristic characteristic,
67                                  NSError error) {
68                              NSData data = characteristic.getValue();
69                              listener.onFound(EnvSensor.create(peripheral.
                                    getIdentifier().toString(), data.getBytes
                                    ()));
70                          }
71                      });
72                      peripheral.discoverServices(null);
73                  }
74                  @Override
75                  public void didFailToConnectPeripheral(CBCentralManager
                        central, CBPeripheral peripheral,
76                          NSError error) {
77                      System.out.println("failed to connected to " +
                            peripheral);
```

```
78                      }
79                  }, null
80                  );
81      }
82      public void stopScan(){
83          central.stopScan();
84      }
85  }
```

A.4 OpenIoTのデータ定義

A.4.1 Observation.java

```
1   package org.openiot.lsm.beans;
2   /**
3   *     Copyright (c) 2011-2014, OpenIoT
4   *
5   *     This file is part of OpenIoT.
6   *
7   *     OpenIoT is free software: you can redistribute it and/or modify
8   *     it under the terms of the GNU Lesser General Public License as published
        by
9   *     the Free Software Foundation, version 3 of the License.
10  *
11  *     OpenIoT is distributed in the hope that it will be useful,
12  *     but WITHOUT ANY WARRANTY; without even the implied warranty of
13  *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
14  *     GNU Lesser General Public License for more details.
15  *
16  *     You should have received a copy of the GNU Lesser General Public License
17  *     along with OpenIoT.  If not, see <http://www.gnu.org/licenses/>.
18  *
19  *      Contact: OpenIoT mailto: info@openiot.eu
20  */
21  import java.util.ArrayList;
22  import java.util.Date;
23
24  /**
25   *
26   * @author Hoan Nguyen Mau Quoc
27   *
28   */
29  public class Observation implements java.io.Serializable {
30      private String id;
31      private Date times;
32      private String sensorId;
33      private String featureOfInterest="";
34      private ArrayList<ObservedProperty> readings;
35      private String metaGraph;
36      private String dataGraph;
37
38      public Observation(){
```

```
39          id = ""+System.nanoTime();
40          readings = new ArrayList<ObservedProperty>();
41      }
42
43      public String getId() {
44          return id;
45      }
46      public void setId(String id) {
47          this.id = id;
48      }
49      public Date getTimes() {
50          return times;
51      }
52      public void setTimes(Date times) {
53          this.times = times;
54      }
55      public String getSensor() {
56          return sensorId;
57      }
58      public void setSensor(String sensorId) {
59          this.sensorId = sensorId;
60      }
61      public String getFeatureOfInterest() {
62          return featureOfInterest;
63      }
64      public void setFeatureOfInterest(String featureOfInterest) {
65          this.featureOfInterest = featureOfInterest;
66      }
67      public ArrayList<ObservedProperty> getReadings() {
68          return readings;
69      }
70      public void setReadings(ArrayList<ObservedProperty> readings) {
71          this.readings = readings;
72      }
73
74      public void addReading(ObservedProperty reading){
75          readings.add(reading);
76      }
77
78      public void removeReading(ObservedProperty reading){
79          readings.remove(reading);
80      }
81
82      public String getMetaGraph() {
83          return metaGraph;
84      }
85
86      public void setMetaGraph(String metaGraph) {
87          this.metaGraph = metaGraph;
88      }
89
90      public String getDataGraph() {
91          return dataGraph;
```

```
92      }
93
94      public void setDataGraph(String dataGraph) {
95          this.dataGraph = dataGraph;
96      }
97
98  }
```

A.4.2 ObserbatonProperty.java

```
1   package org.openiot.lsm.beans;
2   /**
3   *     Copyright (c) 2011-2014, OpenIoT
4   *
5   *     This file is part of OpenIoT.
6   *
7   *     OpenIoT is free software: you can redistribute it and/or modify
8   *     it under the terms of the GNU Lesser General Public License as published
        by
9   *     the Free Software Foundation, version 3 of the License.
10  *
11  *     OpenIoT is distributed in the hope that it will be useful,
12  *     but WITHOUT ANY WARRANTY; without even the implied warranty of
13  *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
14  *     GNU Lesser General Public License for more details.
15  *
16  *     You should have received a copy of the GNU Lesser General Public License
17  *     along with OpenIoT.  If not, see <http://www.gnu.org/licenses/>.
18  *
19  *      Contact: OpenIoT mailto: info@openiot.eu
20  */
21  import java.util.ArrayList;
22  import java.util.Date;
23
24  /**
25   *
26   * @author Hoan Nguyen Mau Quoc
27   *
28   */
29  public class Observation implements java.io.Serializable {
30      private String id;
31      private Date times;
32      private String sensorId;
33      private String featureOfInterest="";
34      private ArrayList<ObservedProperty> readings;
35      private String metaGraph;
36      private String dataGraph;
37
38      public Observation(){
39          id = ""+System.nanoTime();
40          readings = new ArrayList<ObservedProperty>();
41      }
42
```

```
43      public String getId() {
44          return id;
45      }
46      public void setId(String id) {
47          this.id = id;
48      }
49      public Date getTimes() {
50          return times;
51      }
52      public void setTimes(Date times) {
53          this.times = times;
54      }
55      public String getSensor() {
56          return sensorId;
57      }
58      public void setSensor(String sensorId) {
59          this.sensorId = sensorId;
60      }
61      public String getFeatureOfInterest() {
62          return featureOfInterest;
63      }
64      public void setFeatureOfInterest(String featureOfInterest) {
65          this.featureOfInterest = featureOfInterest;
66      }
67      public ArrayList<ObservedProperty> getReadings() {
68          return readings;
69      }
70      public void setReadings(ArrayList<ObservedProperty> readings) {
71          this.readings = readings;
72      }
73
74      public void addReading(ObservedProperty reading){
75          readings.add(reading);
76      }
77
78      public void removeReading(ObservedProperty reading){
79          readings.remove(reading);
80      }
81
82      public String getMetaGraph() {
83          return metaGraph;
84      }
85
86      public void setMetaGraph(String metaGraph) {
87          this.metaGraph = metaGraph;
88      }
89
90      public String getDataGraph() {
91          return dataGraph;
92      }
93
94      public void setDataGraph(String dataGraph) {
95          this.dataGraph = dataGraph;
```

```
96          }
97
98  }
```