Ontology related tasks in Dog2.0

Dario Bonino
Politecnico di Torino, e-Lite research group
http://elite.polito.it
OUTLINE

- Dog2.0
  - Device categories
  - Device models
- Automatic Generation
- Queries
**DOG2.0**

- Device instances and classes in DogOnt can be associated to 2 different components of the Dog2.0 architecture
  - DeviceCategory
    - Defines how a Driver service and a Device service can cooperate (by defining a Java Interface)
  - DeviceModel
    - A representation of a physical device that can be attached to a Driver service
public interface Lamp extends Lighting {
    public static int MATCH_TYPE=100;
    public static int MATCH_SUB_TYPE=50;
    public static int MATCH_MANUFACTURER=0;
    public static String ON_OFF_STATE = "OnOffState";
    public void off();
    public void on();
}
**Device Category (Hi-Fi)**

```java
package it.polito.elite.domotics.model.devicecategory;

/**
 * HiFiCategory - automatically generated by DogOnt2Dog
 */
public interface HiFi extends Entertainment {
    public static int MATCH_TYPE = 100;
    public static int MATCH_SUB_TYPE = 50;
    public static int MATCH_MANUFACTURER = 0;

    public static String STAND_BY_ON_OFF_STATE = "StandByOnOffState";
    public static String SOURCE_SELECTION_STATE = "SourceSelectionState";
    public static String PLAY_STATE = "PlayState";
    public static String TUNER_STATE = "TunerState";
    public static String VOLUME_LEVEL_STATE = "VolumeLevelState";

    public void stop();
    public void stepUp();
    public void rew();
    public void set(Object value);
    public void setVolume(Integer volume);
    public void next();

    public void goToTrack(Integer trackNumber);
}
```
public class DogLamp extends DogDevice implements Lamp {
    public DogLamp(BundleContext context, Properties properties) {
        super(context);
        if (properties == null)
            properties = new Properties();
        properties.put(DEVICE_CATEGORY, Lamp.class.getCanonicalName());
        this.setDeviceProperties(properties);
        this.states.put(Lamp.ON_OFF_STATE,
            new DogStatusVariable(
                properties.getProperty(DogDeviceConstants.DEVICEURI),
                new StatusVariable(Lamp.ON_OFF_STATE, StatusVariable.CM_DER, "-"),
                "current state of Lamp", true));
        this.registerDevice(Lamp.class.getName());
        this.registerStatusVariable();
    }
    @Override
    public void off() {
        ((Lamp) this.driver).off();
    }
    @Override
    public void on() {
        ((Lamp) this.driver).on();
    }
}
AUTOMATIC GENERATION

- Template based
  - Common structure of device categories and device models
- Fill “placeholders” (possibly empty) with information extracted from DogOnt
  - SPARQL
- Works on the schema only
**AUTOMATIC GENERATION (RECIPE)**

### Device Categories

1. GetAllDevicesWithAncestors
2. Foreach device
   1. getDeviceCommands
   2. getDeviceStates
   3. Write the device category code
3. Compile all
4. Jar all with manifest (bundleize)
5. Use in Dog2.0

### Device Models

1. GetAllDevicesWithAncestors
2. Foreach device
   1. getDeviceCommands
   2. getDeviceStates
   3. Write the device model code
3. Compile all
4. Jar all with manifest (bundleize)
5. Use in Dog2.0
Get all devices with ancestors

```sql
SELECT DISTINCT ?x WHERE {
  ?x rdfs:subClassOf dogont:Controllable FILTER (?x!=owl:Nothing)
} ORDER BY ?x
```

- **Recurse over ?x and call**
  - OntClass deviceClass = this.modelLoader.getPlainOntModel().getOntClass(deviceURI);
  - ExtendedIterator iter = deviceClass.listSuperClasses();
QUERIES (2) - COMMANDS

Non-Parametric Commands

```
SELECT DISTINCT ?commandValue WHERE {
  dogont:"+deviceClass+" rdfs:subClassOf
  [rdf:type owl:Restriction; owl:onProperty
dogont:hasFunctionality; owl:someValuesFrom
  ?functionality] . ?functionality
  rdfs:subClassOf dogont:ControlFunctionality.
  ?functionality rdfs:subClassOf [rdf:type
  owl:Restriction; owl:onProperty
dogont:hasCommand; owl:someValuesFrom
  ?command] . ?command rdfs:subClassOf
dogont:NonParametricCommand . ?command
  rdfs:subClassOf [rdf:type owl:Restriction;
  owl:onProperty dogont:realCommandName;
  owl:hasValue ?commandValue]
} ORDER BY
?commandValue
```

Parametric Commands

```
SELECT DISTINCT ?commandValue ?commandParamValue WHERE {
  dogont:"+deviceClass+" rdfs:subClassOf [rdf:type
  owl:Restriction; owl:onProperty dogont:hasFunctionality;
  owl:someValuesFrom ?functionality] . ?functionality
  rdfs:subClassOf dogont:ControlFunctionality .
  ?functionality rdfs:subClassOf [rdf:type owl:Restriction;
  owl:onProperty dogont:hasCommand; owl:someValuesFrom
  ?command] . ?command rdfs:subClassOf
dogont:ParametricCommand . ?command rdfs:subClassOf
  [rdf:type owl:Restriction; owl:onProperty
dogont:realCommandName; owl:hasValue ?commandValue] .
  ?command rdfs:subClassOf [rdf:type owl:Restriction;
  owl:onProperty dogont:commandParamName; owl:hasValue
  ?commandParamValue]
} ORDER BY ?commandValue
```
SELECT ?state WHERE{
  dogont:"+deviceClass+
  rdfs:subClassOf [rdf:type
  owl:Restriction;
  owl:onProperty
  dogont:hasState;
  owl:someValuesFrom ?state] } 
ORDER BY ?state
Some Figures

- Total execution time <10s
  - ~7 seconds to load and reason the ontology
- Amount of compiled classes
  - 228 classes
    - 114 device categories
    - 114 device models
RULES GENERATION (DOG1.0)

Query

```
?class ?cv WHERE { ?x a
dogont:Controllable . ?y a
dogont:Controllable . ?x
dogont:controlledObject ?y . ?x
dogont:hasFunctionality ?f . ?f
dogont:hasNotification ?n . ?n
dogont:notificationValue ?v .
?f2 dogont:hasFunctionality ?f2.
?f2 dogont:hasCommand ?c .
?c rdf:type ?class .
?class rdfs:subClassOf
dogont:DiscreteCommand .
?class rdfs:subClassOf [rdf:type
owl:Restriction; owl:onProperty
dogont:realCommandName;
owl:hasValue ?cv]}
```

Rule

```
when
dev:Device( id ==
"http://elite.polito.it/ontologies/simplehome.owl#"?x")
cmd:StateCommand(state =="?v")
status:DeviceStatus( status == cmd, device == dev)
rulesCore:RulesCore()
message: DogMessage(dataType ==
DogBundle.DataTypeEnum.DEVICE_STATUS,
data == status)
then
Device newDev = new
Device("http://elite.polito.it/ontologies/DOGHouseModel.owl#"+?y+";
DeviceStatus newStatus = new DeviceStatus(newDev, new
StateCommand("+?cv+", StatusTypeEnum.DISCRETE));
DogMessage myCmd = new
DogMessage(MessageTypeEnum.CMD,new
GregorianCalendar(),
message.getDataType(), newStatus);
rulesCore.takeActions(myCmd);
```
QUESTIONS?

Dario Bonino
dario.bonino@polito.it