WildCAT

A Generic Framework for Context-Aware Applications

http://wildcat.ow2.org

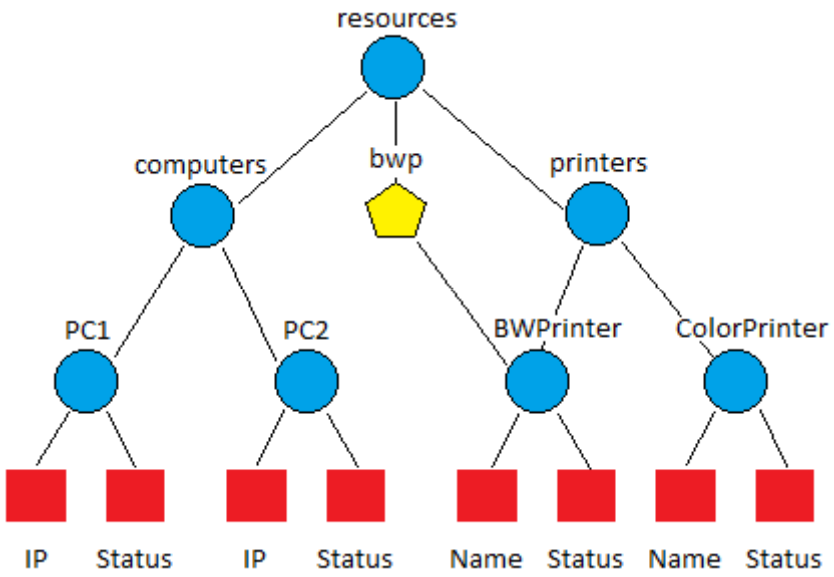
WildCAT in a nutshell

- Generic framework for context-aware applications
  - Sensor-based monitoring (generic POJOs)
  - Data aggregation
- Hierarchical data representation
  - Tree-oriented (Unix file system analogy)
  - Dynamically updated (at runtime)
- Data inspection
  - Synchronous (pull)
  - Asynchronous (push) –> event-based notification
- Distribution support (RMI and/or JMS)
- Esper backend: open source Complex Event Processing (CEP) engine
  - SQL-like query language for event processing
  - Event pattern matching, sliding window, etc.

Modeling Data

- WildCAT context: 2 types of nodes
  - Resources (basic or symbolic links)
  - Attributes (hold values)
- Addressing resources
  - Path: "self://resources/computers/PC1#IP"
  - Symbolic link: "self://resources/bwp#Status "

  = "self://resources/printers/BWPrinter#Status"
- Composite attributes
  - + * / % || &&
  - Query attributes



Inspecting Data (Pull mode)

- Generic API for data inspection
  - Context API: the only entry point
  - CRUD operations on resources/attributes
- Context creation

  Context ctx = ContextFactory.getDefaultFactory().createContext();
- Getting & Setting attributes

  ctx.createAttribute("self://constants#hello", "Hello");

  System.out.println("self://constants#hello = " + ctx.getValue("self://constants#hello"));
- Listing hierarchy content

  System.out.println("self:// :: "+ ctx.list("self://"));
- Symbolic links

  ctx.createSymbolicLink("self://demo/soft/link/toConstant", "self://constants");

Inspecting Data (Push mode)

- Event-based notification: 2 types of events
  - WHierarchyEvent : triggered by resources (hierarchy modifications)
  - WAttributeEvent : triggered by attributes (value changes)
- Action-based event handling:
  - WAction : action to be performed when an event is triggered
  - Event/Action programming model

1 Define your action

```
public class MyAction extends WAction {
 //constructor
 public void onEvent() {
   System.err.println("My action  performed");
 }
}
```

2 Register it to the event

```
String query = "select * from WEvent";

WAction action = new MyAction("myAction","simple
action to perform");

ctx.registerActions(query, action);
```

Inspecting Data (Push mode)

- Event Query Language (EQL)
  - Esper backend: open source Complex Event Processing (CEP) engine
  - SQL-like query language (SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY)
  - Sliding windows
  - Pattern matching

- Example

  SELECT avg(value.load)? FROM WAttributeEvent(source="self://proc/cpu#info").win:length(5sec)

- Query Attributes
  - Holds a query result
  - Dynamically updated (at runtime)

EQL samples

- Select 'second' from the resource-attribute 'self://date#time'

  select value('second')? from WAttributeEvent(source = 'self://date#time')

- Select all 'A' events with the 'second' value < 30

  *select * from pattern[every A=WAttributeEvent(source = 'self://date#time', value('second')? < 30)]*

- Select the max value of 'second' in a time window of 5 seconds

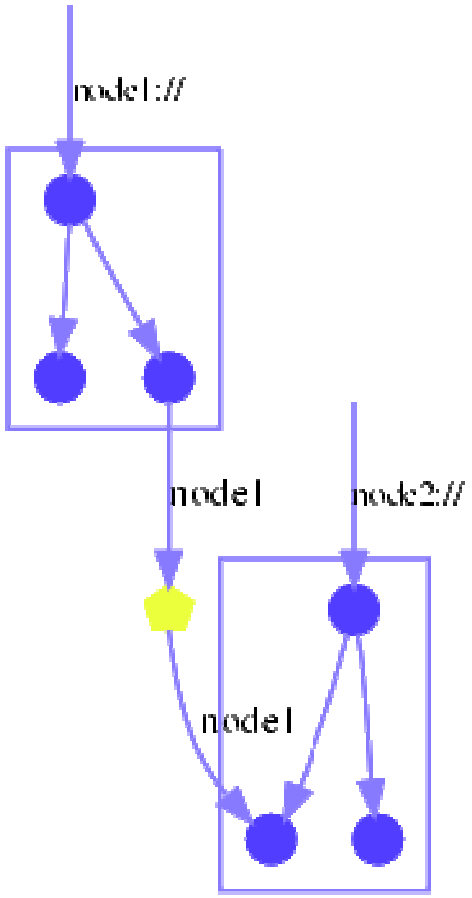
  select max(value('second')?) from WAttributeEvent(source = 'self://date#time').win:time(5)

- Select all clients with balance < 0 and then followed by balance < -500 after 5 seconds

  *select * from pattern[every A=WAttributeEvent(source = 'self://bank/account/*', value('balance')? < 0) -> every B=WAttributeEvent(source = 'self://bank/account/*', value('balance')? < -500)].win:length(5) where A.value('owner')?=B.value('owner')?*

Distribution support

- Distributed contexts
  - Connected using symbolic links
  - Inter-context communication
- Communication modes
  - RMI , JMS, RMI+JMS
  - Registry, Dispatchers



WildCAT Sensors

- Built-in Sensors
  - Java world: Runtime, Date/Time, System properties, JMX, etc.
  - Linux only: Kernel version, CPU properties, Memory/CPU load, etc.
- User defined Sensors
  - POJOAttribute: class to extend to define a new wildcat sensor
  - getValue/setValue: methods to override for new sensor behavior

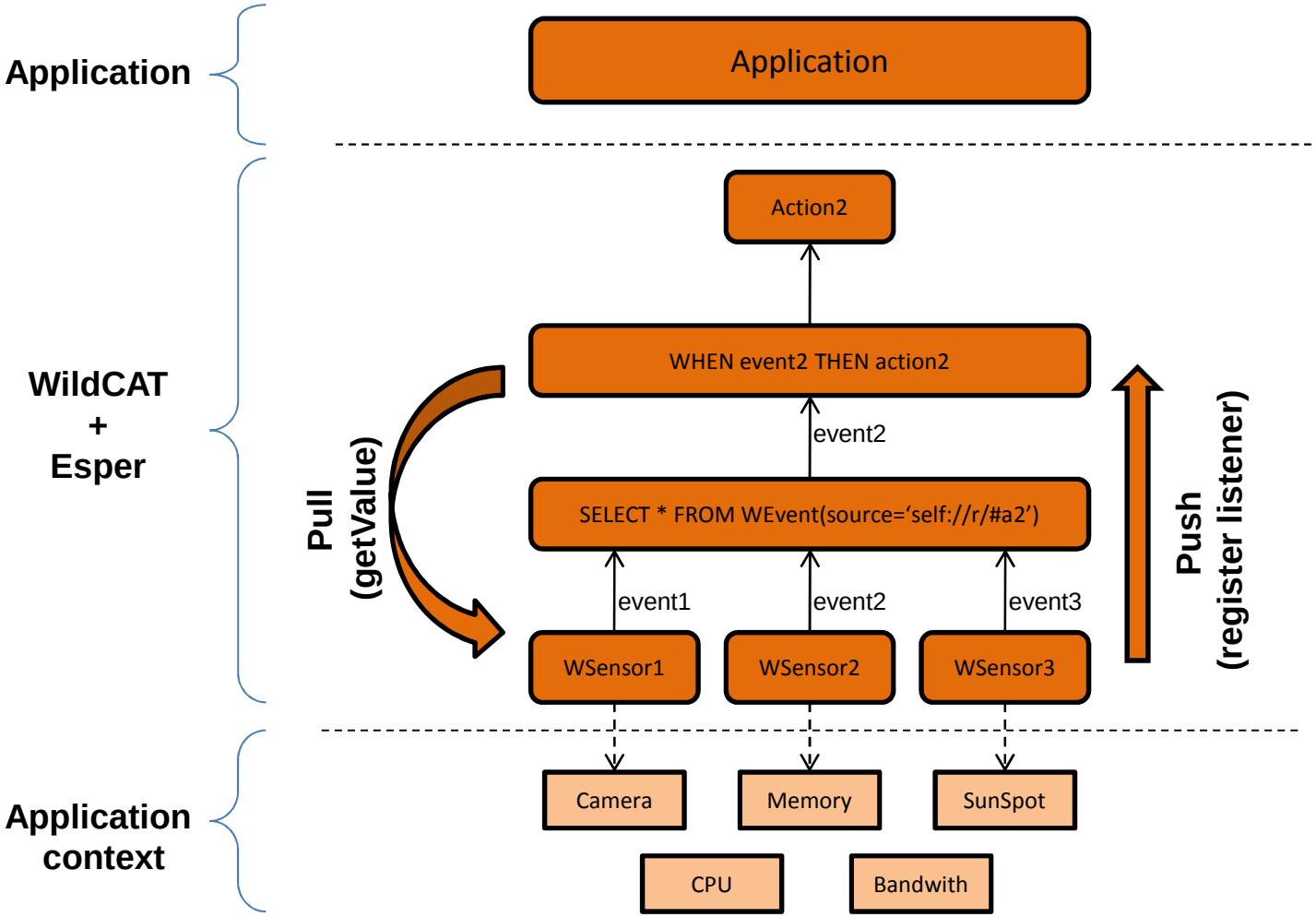
1 Define your sensor

```
public class MySensor extends POJOAttribute {
 //state & constructor
 @Override
 public void setValue(Object value) {//method body}

@Override
 public Object getValue(){//method body}
}
```

2 Attach it to an attribute

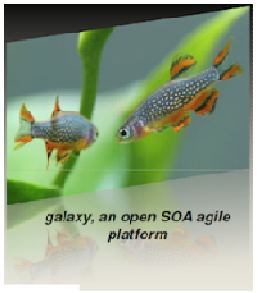
```
MySensor sensor = new MySensor();

ctx.attachAttribute("self://resource#attribute", sensor);
```

WildCAT: The big picture



WildCAT in action

- ADT Galaxy project (*http://galaxy.inria.fr*), 2008-2010
  - An open SOA platform
  - QoS monitoring for SOA/SCA application
  - BPEL execution monitoring for business processes



- OW2 JASMINe project (*http://jasmine.ow2.org*), since 2006
  - A smart tool for SOA platform management
  - JASMINe Monitoring - WildStat module (include WildCAT)

- ANR Selfware project, 2005-2008
  - A software infrastructure for building self-distributed applications
  - Smart probes to notify relevant events correlation

WildCAT card

- General stats :
  - Small footprint : 112 Ko + Esper 2.7 Mo
  - Small API : 12 interfaces, 50 Classes
  - Current stable version : 2.3.0
  - Current Esper version : 3.4.0
  - License : GPL v2
- Team
  - Thomas Ledoux (project leader)
  - Pierre Charles David (V1 initial API and Implementation)
  - Nicolas Loriant (V2 initial API and Implementation)
  - Loris Bouzonnet (Contributor)
  - Mahmoud Ben Hassine (Contributor)
  - Olivier Barais (Contributor)
- Home page & Contact
  - http://wildcat.ow2.org
  - wildcat@ow2.org