

A REST server for IRMIS3

Gabriele Carcassi

Summary

- What is IRMIS?
- IRMIS3 architecture
- Getting data
- Writing data
- Implementation
- Performance

What is IRMIS?

DB and tools to keep track of accelerator parts and their relationships

The screenshot shows the IRMIS Component Browser interface. On the left is a tree view of components, including Site, Building, Room, Rack, and VME Chassis. The main area is divided into three panels: Control, Housing, and Power. The Control panel shows a Network component with an EVG100. The Housing panel shows a Site component with a Building 902, Room 13 Control room, Rack 1-1, and VME Chassis - Mupac containing an EVG100. The Power panel shows a Utility component with a Switch Gear, Circuit Breaker, AC Panel, 120VAC Power Strip/Outlet, VME Power Supply - Mupac, and VME Chassis - Mupac containing an EVG100. Below these panels is a detailed view of the EVG100 component, showing its name, description, manufacturer, form factor, and properties. A diagram at the bottom shows the EVG100 component connected to another EVG100 component via a fiber optic connection.

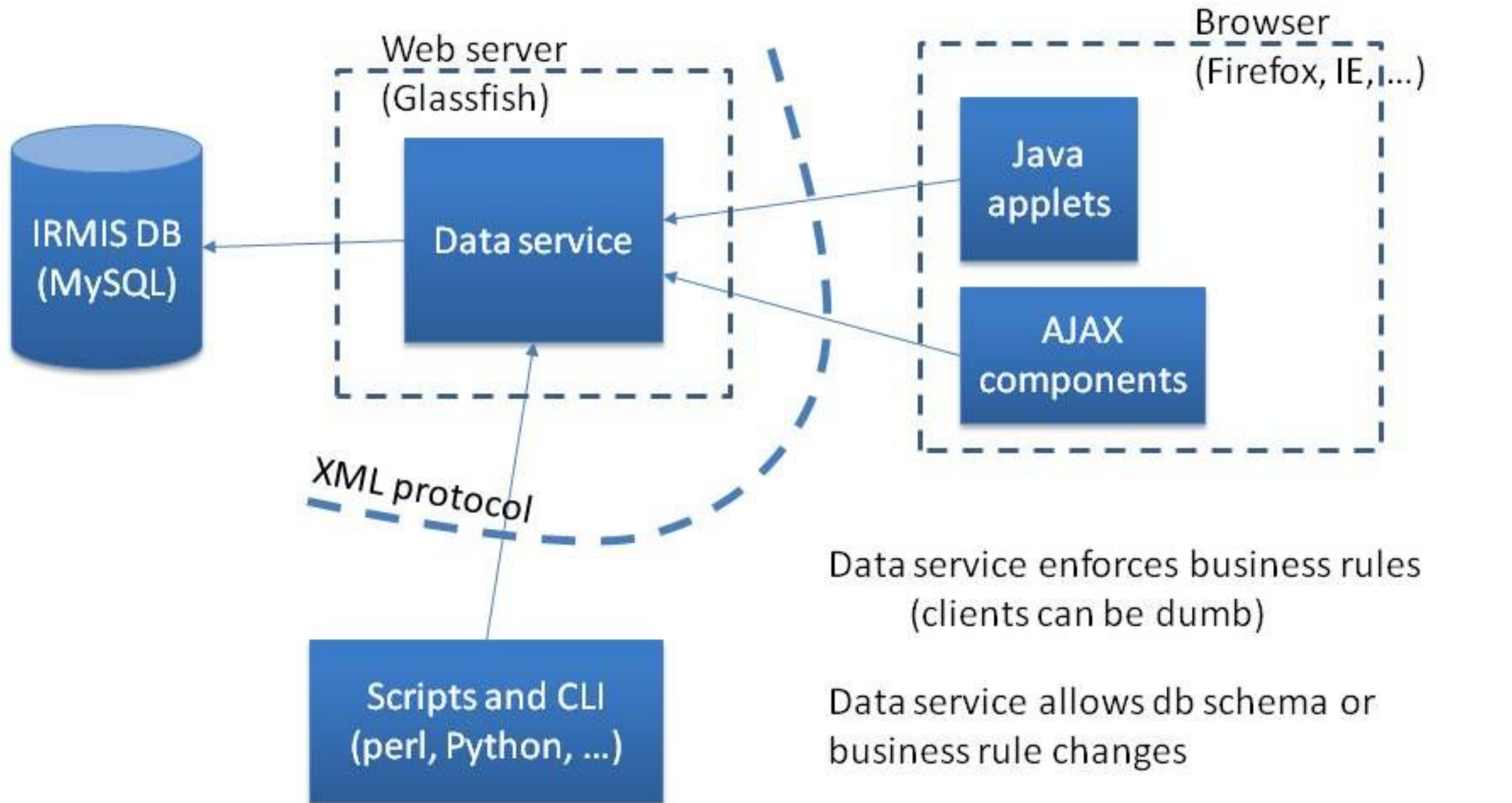
Where is it?

How is it powered?

How is it controlled?

How is it connected?

IRMIS3 architecture



Data service enforces business rules
(clients can be dumb)

Data service allows db schema or
business rule changes

Data service provides a good abstraction
layer on which to integrate

Getting data

```
<componentType id="90" description="Oscilloscope: 4 Channel @ 500MS/s; 500MHz BW" name="HP54540A">
  <manufacturer id="5" name="Agilent (HP)"/>
  <formFactor id="5" description="Freestanding"/>
  <functions>
    <function id="46" description="CCMS"/>
    <function id="8" description="Instrument"/>
  </functions>
  <requires>
    <interface id="143" description="GPIB_Slave" relType="control"/>
    <interface id="5" description="Freestanding" relType="housing"/>
    <interface id="78" description="120VAC" relType="power"/>
  </requires>
  <presents>
    <interface id="234" description="Port" relType="control"/>
  </presents>
  <ports>
    <port name="Chnl 1">
      <portType id="13" name="BNC-F" group="RF Connectors">
        <pinDesignator id="303" name="1"/>
      </portType>
      <pin usage="Chnl 1" pinDesignatorId="303">
        <signalType id="1" direction="IN"/>
      </pin>
    </port>
    <port name="Chnl 2">...</port>
    <port name="Chnl 3">...</port>
    <port name="Chnl 4">...</port>
    <port name="RS232">...</port>
    ...
  </ports>
</componentType>
```

XML returned
REST style (data service)

De-normalized data

Id references

This is an example of what the
data service might deliver

Writing data

XML describing a transaction sent through a POST

```
<transaction xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xmlns='http://xml.bnl.gov/schema/irmis'
  xsi:schemaLocation='http://xml.bnl.gov/schema/irmis irmis.xsd'>
  <create>
    <manufacturer name="Acopian"/>
  </create>
  <update>
    <componentType id="2" name="Site" description="Overall facility. Root of housing hierarchy.">
      <manufacturer id="1" name="None"/>
      <formFactor id="1" description="Virtual"/>
      <properties/>
      <functions/>
      <requires/>
      <presents/>
      <ports/>
    </componentType>
  </update>
  <delete>
    <formFactor id="4" description="IndustryPack"/>
  </delete>
  <connect>
    <cable color="blue" label="AA" portAId="123" portBId="321"/>
    <conductor pinAId="123" pinBId="321"/>
    <conductor pinAId="124" pinBId="322"/>
  </connect>
</transaction>
```

different independent “commands”

no lock “FOR UPDATE”

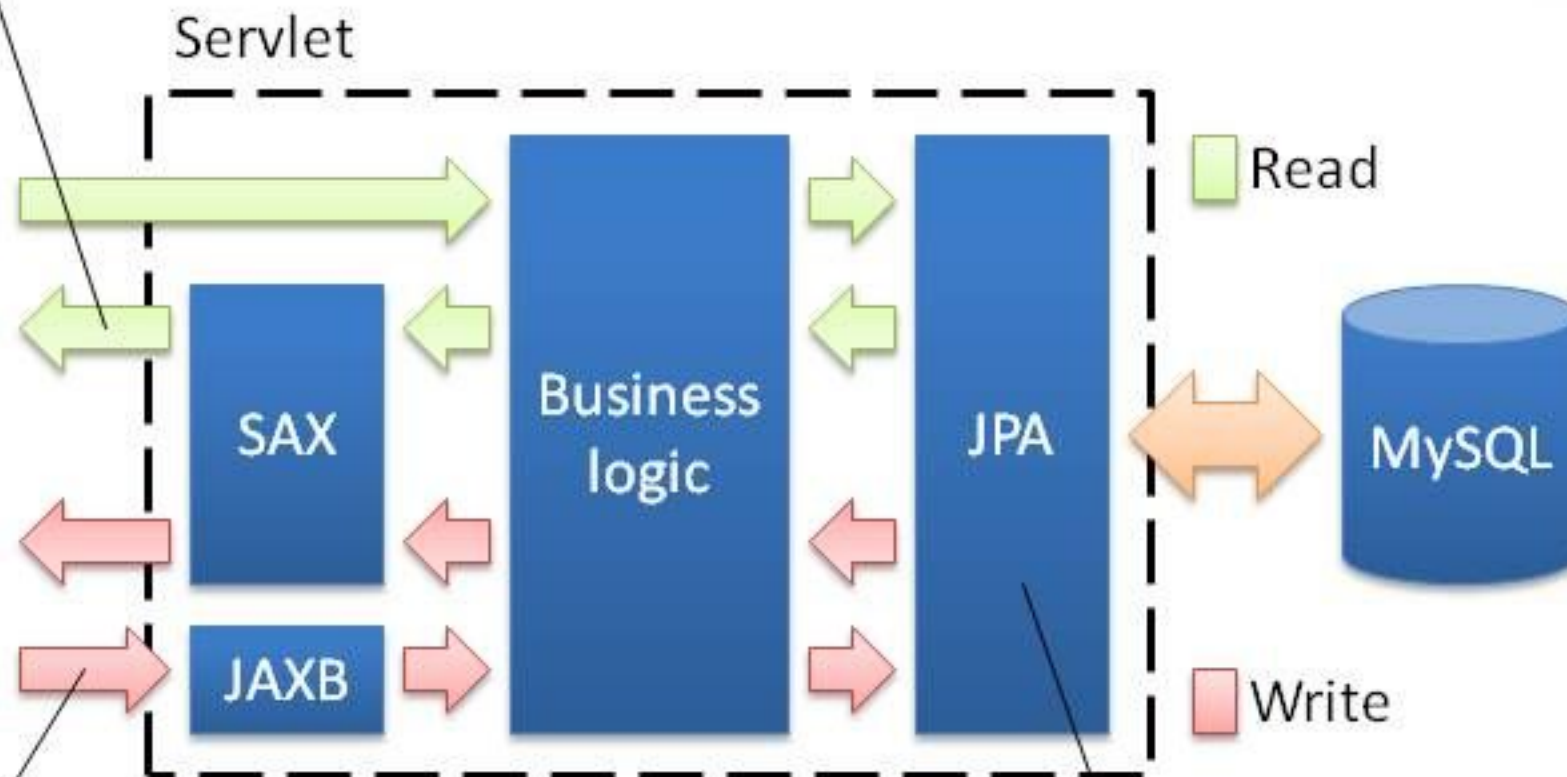
idempotent actions (error or same result)

allows remote/disconnected work

Implementation

JAVA SE 6
JAVA EE 5
GLASSFISH 2.X

Stream out XML as soon as possible
to reduce delay



JAXB to convert transaction request
to Java objects

JPA vs JDBC: simpler queries to maintain
6% performance loss

Performance

- Dell Precision M4300 laptop, Intel Core Duo T7500, 2.20 GHz - APS dataset
 - Entire component data set streamed in about 5 seconds (31338 components, 63911 relationships)
 - Most other queries under 100 ms (manufacturers 88ms, interfaces 95ms)
- Roughly half of the time is spent in DB access and half in XML generation
 - Streaming starts right after DB access
- Only tuning done is on the database query (was “good enough”)
 - Other areas: hardware/os tuning, xml generation improvements, profiling, caching, ...

Conclusion

- 12 releases, 6 database schema changes within a year
- Interests from other sites
 - NSLC (Michigan State University) contributed a php API
 - Developed a small inventory management on top
- Consider services in front of your databases
 - Easier to evolve, encapsulate business logic
- Use REST for data and SOAP/XML-RPC for RPC

