akquinet

iPOJO

A Tale about Simplicity

Simplified version, with focus on **iPOJO-related syntax and examples**

[Ada Diaconescu]

akquinet

What about me ?

- Solution Architect in the Modular and Mobile CC
- Apache Software Foundation
  - PMC Apache Felix, Apache Ace
  - Apache Felix iPOJO project leader
- OW2
  - Chameleon project leader

The Apache Software Foundation
http://www.apache.org/

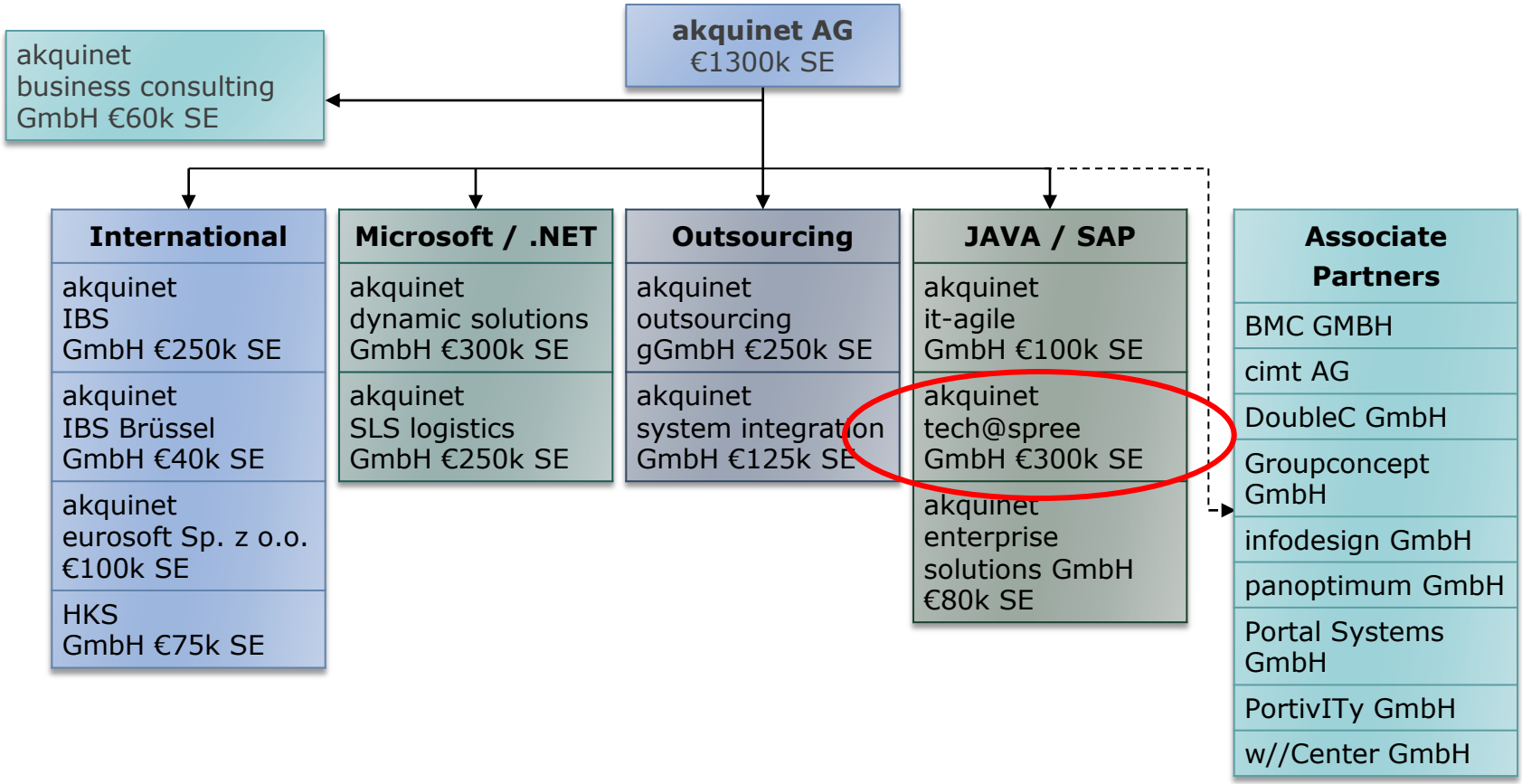
OW2 Consortium

APACHE felix

iPOJO

chameleon

akquinet


akquinet AG
€1300k SE
akquinet
business consulting
GmbH €60k SE
International
akquinet
IBS
GmbH €250k SE
akquinet
IBS Brüssel
GmbH €40k SE
akquinet
eurosoft Sp. z o.o.
€100k SE
HKS
GmbH €75k SE
Microsoft / .NET
akquinet
dynamic solutions
GmbH €300k SE
akquinet
SLS logistics
GmbH €250k SE
Outsourcing
akquinet
outsourcing
gGmbH €250k SE
akquinet
system integration
GmbH €125k SE
JAVA / SAP
akquinet
it-agile
GmbH €100k SE
akquinet
tech@spree
GmbH €300k SE
akquinet
enterprise
solutions GmbH
€80k SE
Associate Partners
BMC GMBH
cimt AG
DoubleC GmbH
Groupconcept GmbH
infodesign GmbH
panoptimum GmbH
Portal Systems GmbH
PortivITy GmbH
w//Center GmbH

akquinet

Modular and Mobile Solutions

- Competence Center focusing on
  - Modular Systems
    - Modularization expertise
    - OSGi-based
    - Sophisticated, Large scale, Distributed systems
  - Mobile Solutions
    - *In the large*
      - Mobile devices, Interactions middleware, Server-side …
      - M2M, B2B
- Open Technologies
  - OSGi (Apache Felix, Apace Ace, OW2 Chameleon, Apache Sling…)
  - Android
  - Java EE (JBOSS, OW2 JOnAS)

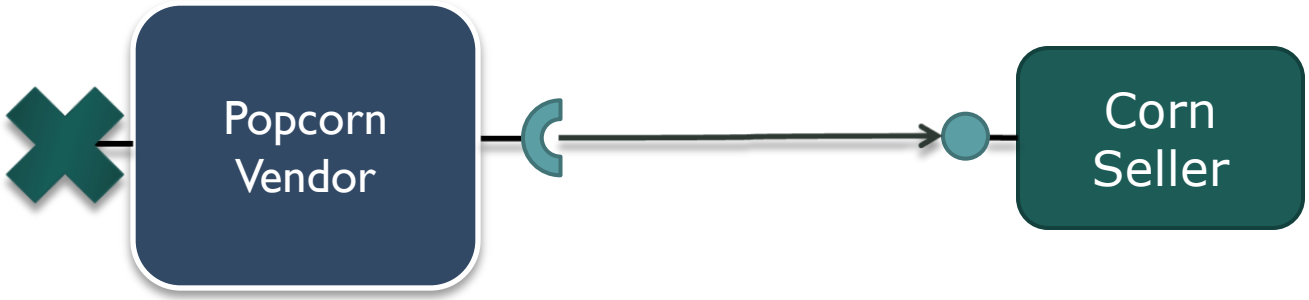
iPOJO

Simplified version, with focus on
iPOJO syntax and examples
[Ada Diaconescu]

The fastest path ?

Snark Bar - Application Example

- OSGi™ Snack Bar
  - Pop Corn Vendor



Popcorn Vendor service listener

```
public synchronized void serviceChanged(ServiceEvent event) {

        // If a  seller was registered, then see if we need one.
        // If  so, then get a reference to it.
        if (event.getType() == ServiceEvent.REGISTERED) {
                if (m_ref == null) { // get a reference to the service object.
                        m_ref = event.getServiceReference();
                        m_seller = (Vendor) m_context.getService(m_ref);
                        m_sr = m_context.register(Vendor.class.getName(), this, null);
                }
        }

        // If a seller was unregistered, then see if it was the one we were using.
        // If so, then unget the service and try to query to get another one.
        else if (event.getType() == ServiceEvent.UNREGISTERING) {
                if (event.getServiceReference() == m_ref) { // unget service object and null references.
                    m_context.ungetService(m_ref);
                    m_ref = null;
                    m_seller = null;
                    // Query to see if we can get another service.
                    ServiceReference[] refs = m_context.getServiceReferences(Vendor.class.getName(), "(type=corn)");
                    if (refs != null) {
                        m_ref = refs[0]; // Get a reference to the first service object.
                        m_seller = (Vendor) m_context.getService(m_ref);
                    } else { // No more provider.
                        m_sr.unregister();
                     }
        }
}
```

Implementation using iPOJO

```
@Component(name="popcorn-provider")
@Provides
public class PopcornVendor implements Vendor {
    @Requires
    private Reseller m_vendor;

    public  String getName() { return "Popcorn from Paris"; }
    public String buy() {
            m_vendor.buy();
            return "popcorn";
    }
}
```

---

metadata.xml file:

```
<ipojo>
        <instance component="popcorn-provider"/>
</ipojo>
```

Implementation using iPOJO annotations only (no metadata.xml file)

```
@Component(name="popcorn-provider")
@Instantiate
@Provides
public class PopcornVendor implements Vendor {
    @Requires
    private Reseller m_vendor;

    public  String getName() { return "Popcorn from Paris"; }
    public String buy() {
            m_vendor.buy();
            return "popcorn";
    }
}
```

iPOJO

- Apache Felix sub-project
- iPOJO provides
  - A very simple development model (POJO)
  - The automation of service-based interactions
    - Service Publication,
    - Service Properties,
    - Service Discovery,
    - Service Tracking,
    - Service Binding.
  - An extensible infrastructure
    - To manage other non-functional features

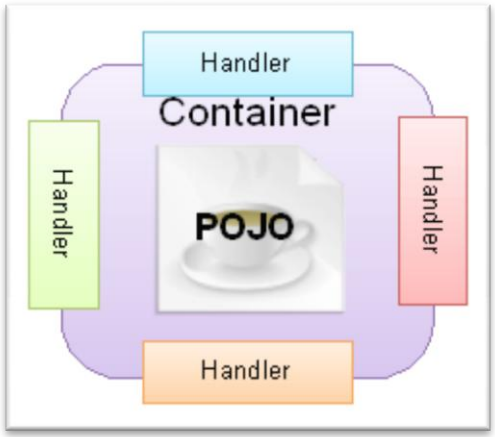
akquinet

iPOJO overview

- Differentiation between
  - Component types (<component>)
    - Describe non-functional requirements
  - Instances (<instance>)
    - You can create several instances from the same component types
    - Will provide services, require services…
- Injection mechanism
  - The Container can interact with the POJO by
    - Invoking methods on the POJO (Container => POJO)
    - Injecting value inside fields (Container => POJO)
    - Be notified of POJO changes on fields (POJO => Container)

Mechanism & Container

- Bytecode weaving
  - Aspect-like mechanisms
- Metatada format agnosticism
  - XML, Annotation, API
- Highly Extensible
  - Handlers



akquinet

Component Type

- @Component

  - Goal: Declares a component type

  - Target: class definition

    ```
    @Component
    public class HelloConsumer{…}
    ```

  - Attributes:
    - Name: factory name
    - Immediate: is the instance immediately activated (boolean)
    - Propagation: enables / disables configuration propagation
  - …

Service Providing

- @Provides

  - Goal: defines the component's provided services
    - By default: All implemented interfaces will be published

  - Target: class definition

```
@Component
@Provides
public class HelloImpl implements HelloService{…}
```

  - Attributes:
    - Specifications: set the published interfaces (class[])
    - Factory/Strategy: Service Object creation strategy
      - {SINGLETON|FACTORY|INSTANCE|…}

Service Properties

- @ServiceProperty

  - Goal: define a service property
    (published with the service)

  - Target: field

```
@ServiceProperty
private String m_message;
```

  - Attributes:
    - name: property name (optional, default=field name)
    - value: property value (optional, default=no value)
    - mandatory: is the property mandatory? (optional, default=false)

akquinet

Requiring a service

- iPOJO Service requirement
  - Optional or Mandatory
  - Scalar or Aggregate
  - Filter - defines an LDAP filter (optional)
  - Binding policy - static, dynamic, dynamic priority
- Involves service lookup and service binding
- Two approaches:
  - Field injection: service reference injected automatically
  - Method invocation: the developer manages the dynamics

Requiring a service – field injection

- @Requires
  - Goal: defines an iPOJO Service dependency
  - Target: field (field injection)
  - Attributes:
    - Filter: defines an LDAP filter (optional)
    - Optional: is the dependency optional? (default = "false")
    - Policy: binding policy – static, dynamic, dynamic-priority
    - From: defines a specific provider
    - …

Requiring a service – field injection

- Hello service – field injection exemple
- Annotations metadata

```
@Component
@Instantiate
public class HelloConsumer {
    @Requires
    private Hello m_hello;

    public doSomething() {
        System.out.println(m_hello.getMesage());
    }
}
```

Requiring a service – field injection

- Hello service – field injection exemple
- XML metadata

```
<component classname="...HelloConsumer">
  <requires field="m_hello"/> ...
</component>
```

Field injection of Services

- Synchronization management

```
@Requires
Foo m_foo;

public void statelessAccess() {//synchronization managed by iPOJO
        m_foo.doA()
        m_foo.doB()
        m_foo.doC()
}
```

akquinet

Requiring a service - method injection

- HelloConsumer - method invocation exemple - Annotations

```
@Component
public class HelloConsumer {
   private Hello m_hello;

   @Bind
   public void bindHello(Hello h) { m_hello = h; }

   @Unbind
   public void unbindHello() { m_hello = null; }

   public doSomething() {
         System.out.println(m_hello.getMesage());
   }
}
```

Requiring a service - method injection

- HelloConsumer - method invocation exemple – XML metadata

```
<component classname="...HelloConsumer">
   <requires>
         <callback type="bind"
                  method="bindHello">
         <callback type="unbind"
                  method="unbindHello">
   </requires> ...
</component>
```

Field injection with Aggregation

```
@Component
public class HelloConsumer {
   @Requires
   private Hello m_hellos[];

   // Array => Aggregate
   public doSomething() {
         for(int I = 0; I < m_hellos.length; i++) {
         System.out.println(
                        m_hellos[i].getMessage());
         }
   }
}
```

Method invocation with Aggregation

```
public class HelloConsumer {
   private List m_hellos = new ArrayList();

   @Bind(aggregate=true)
   private void bindHello(Hello h) {       m_hellos.add(h); }

   @Unbind
   private void unbindHello(Hello h) {    m_hellos.remove(h); }

   public synchronized doSomething() { //
         for(Hello h : m_hellos) {
                  System.out.println(
                           h.getMessage());
         }
   }
} }
```

Optional requirement with field injection

```
@Component
public class HelloConsumer {

@Requires(optional=true)
private Hello m_hello;

public doSomething() {
   System.out.println(m_hello.getMesage());
}
}
```

- *Nullable* object is injected when no service is available
  - Implements the interface and does nothing

Optional requirement with method invocation

```
@Component
public class HelloConsumer {

   private Hello m_hello;

   @Bind(optional=true)
   public void bindHello(Hello h) { m_hello = h; }

   @Unbind
   public void unbindHello() { m_hello = null; }

   public doSomething() {
      if(m_hello != null) { // Must be checked
          System.out.println(m_hello.getMesage()); }
   }
}
```

Filtered requirements

- **Field injection**

```
@Requires(filter="(language=fr)")
private String DictionaryService dict;
```

- **Method invocation**

```
@Bind(filter="(langage=en)")
public void bindHDictionary(DictionaryService svc) { ... }

@Unbind
public void unbindDictionary() {...}
```

akquinet

Binding policies

- Static policy

```
@Requires(policy="static")
private Hello[] m_hellos;
```

- Dynamic-priority policy

```
@Requires(policy="dynamic-priority")
private Hello[] m_hellos;
```

akquinet

Lifecycle

- iPOJO Instance state
  - INVALID | VALID

```
@Validate
protected void activate() {
  SwingUtils.invokeAndWait(new Runnable() {
    public void run() {  setVisible(true); }
  });
}

@Invalidate
protected void deactivate() {
   SwingUtils.invokeLater(new Runnable() {
       public void run() { setVisible(false); dispose(); }
   });
}
```

akquinet

Configuration

- Component type declare Property
  - @Property
  - Method: setX(value)
  - Field
- Component Instance receive configurations
  - Instance configuration (XML)
  - 'Factory' configuration (API)
  - Configuration Admin
- @Instantiate does *not* allow configurations

Instance Configuration

- @Component => Component type
- <Instance> => Instance
  - metadata.xml
    - Not necessary in the same bundle
  - Supports configuration

```
<instance component=”foo”/>

<instance component="buns_and_wieners">
    <property name="buns" value="9"/>
    <property name="wieners" value="8"/>
</instance>
```

- Cfg files
  - Java properties file

Conclusion

How are you ?

akquinet

Modularity, OSGi, SOCM

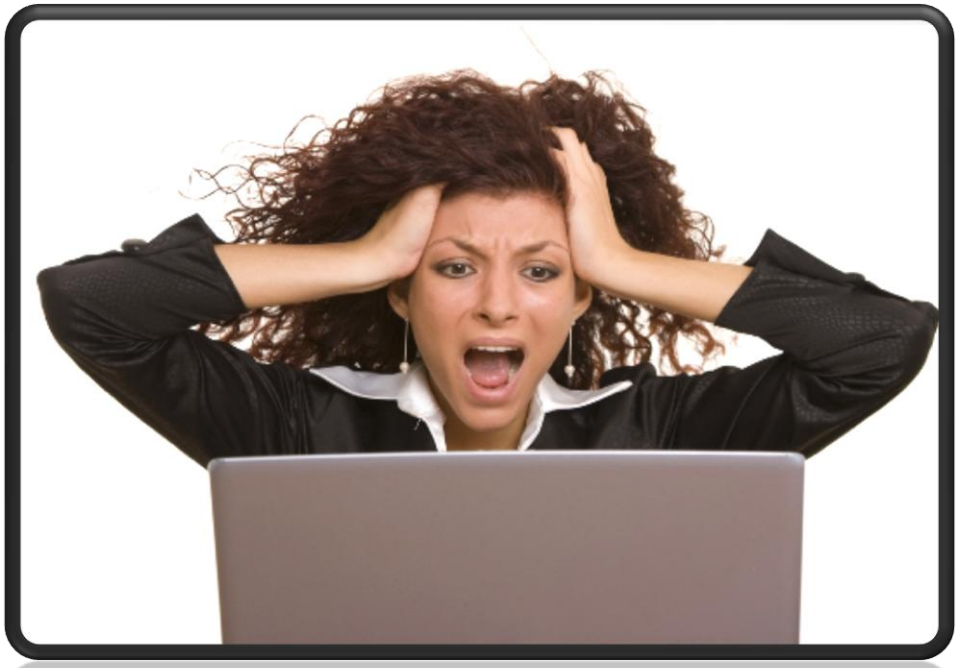
- The objectives
  - Dynamic, modular applications
- The trail
  - OSGi
- The easiest way
  - SOCM

iPOJO

- Apache Felix sub-project
  - 5+ years of development
  - Used in numerous projects
    - Application server
    - Gateway
    - Mobile phone
- It’s the most advanced component model for OSGi

Questions ?

COMASIC:
ada.diaconescu_at_telecom-paristech.fr



Karl Pauls
karl.pauls@akquinet.de
Bülowstraße 66, 10783 Berlin
+49 151 226 49 845

Dr. Clement Escoffier
clement.escoffier@akquinet.de
Bülowstraße 66, 10783 Berlin
+49 175 2467717