What can enterprise mash-up technologies do for your?

Practical learning by looking at WebSphere sMash

Johan Eltes
johan.eltes@callistaenterprise.se
www.callistaenterprise.se

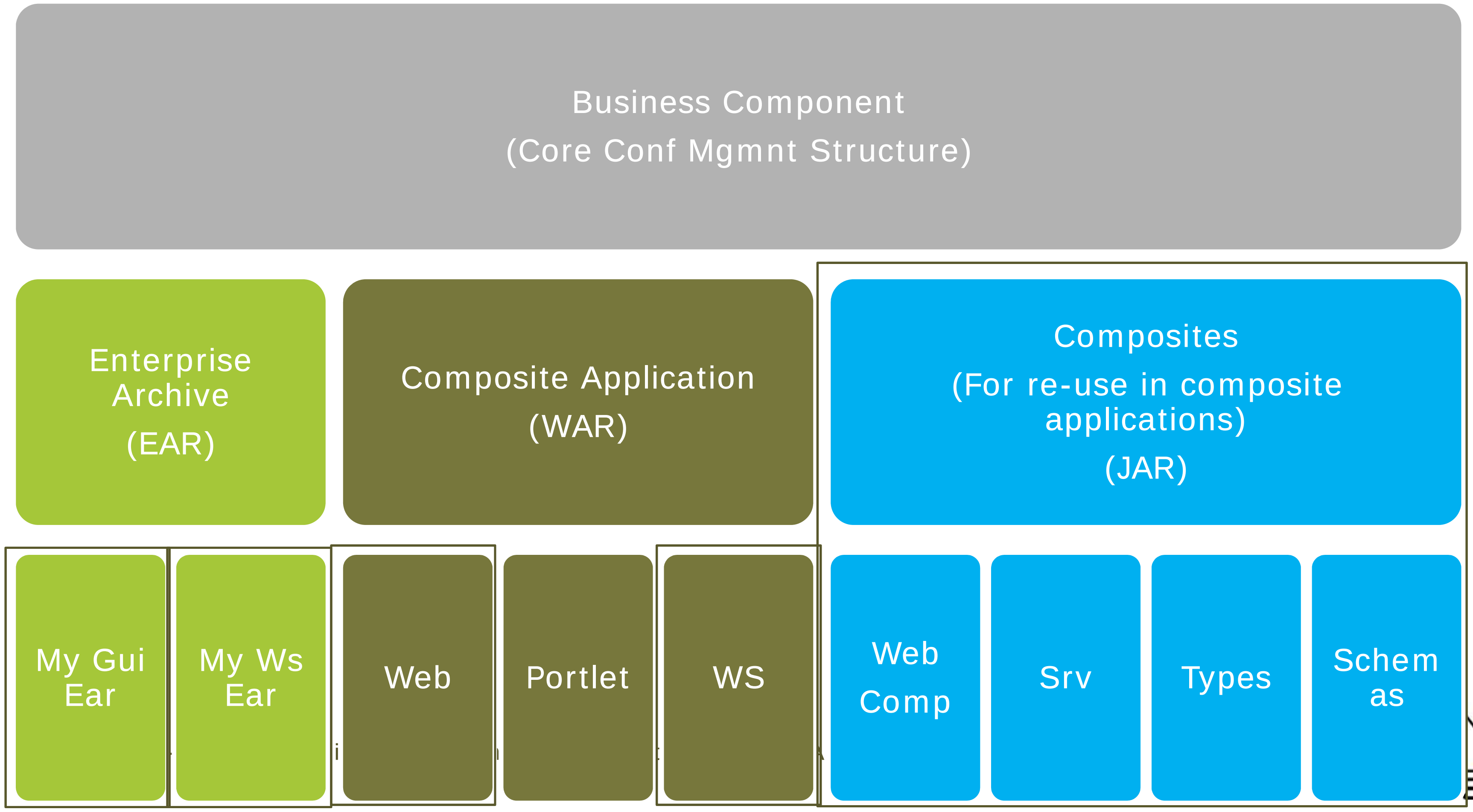
My wake-up call

- Customer requirements:
  - Web application to search for contact information on employees and locations of the enterprises sites located in the enterprise LDAP –based directory

How would an Enterprise Java Architect approach this?

Enterprise Java Architect Approach

- Follow the prescribed reference architecture
  - Presented at last years Cadec



The customer proposed a different approach

- Supplementary requirements clarified
  - The information displayed in search result should be up-to-date with yesterdays information in LDAP master (or better)
- Proposal
  - Give every information object in LDAP (not structural nodes) an HTTP URL that produces XML output
  - Produce one - or as few as practically possible - HTML files with all these URLs, so that our search engine can index them
  - Use the search engine client to search for information in our people directory and present the output of the selected URL
- Background
  - The customer uses Websphere Omnifind to search-enable enterprise content

Some reflexions

- There are many generic tools made for the web
- Customers are used to the web architecture
- We do find information using Google
- The solution was “quick and dirty” but still produced and re-used services in very short time and with very low risk
- Enterprise Java architects may need to extend their mind- and tool-set…

Welcome to the new world of Enterprise Mash-ups!

CALLISTA

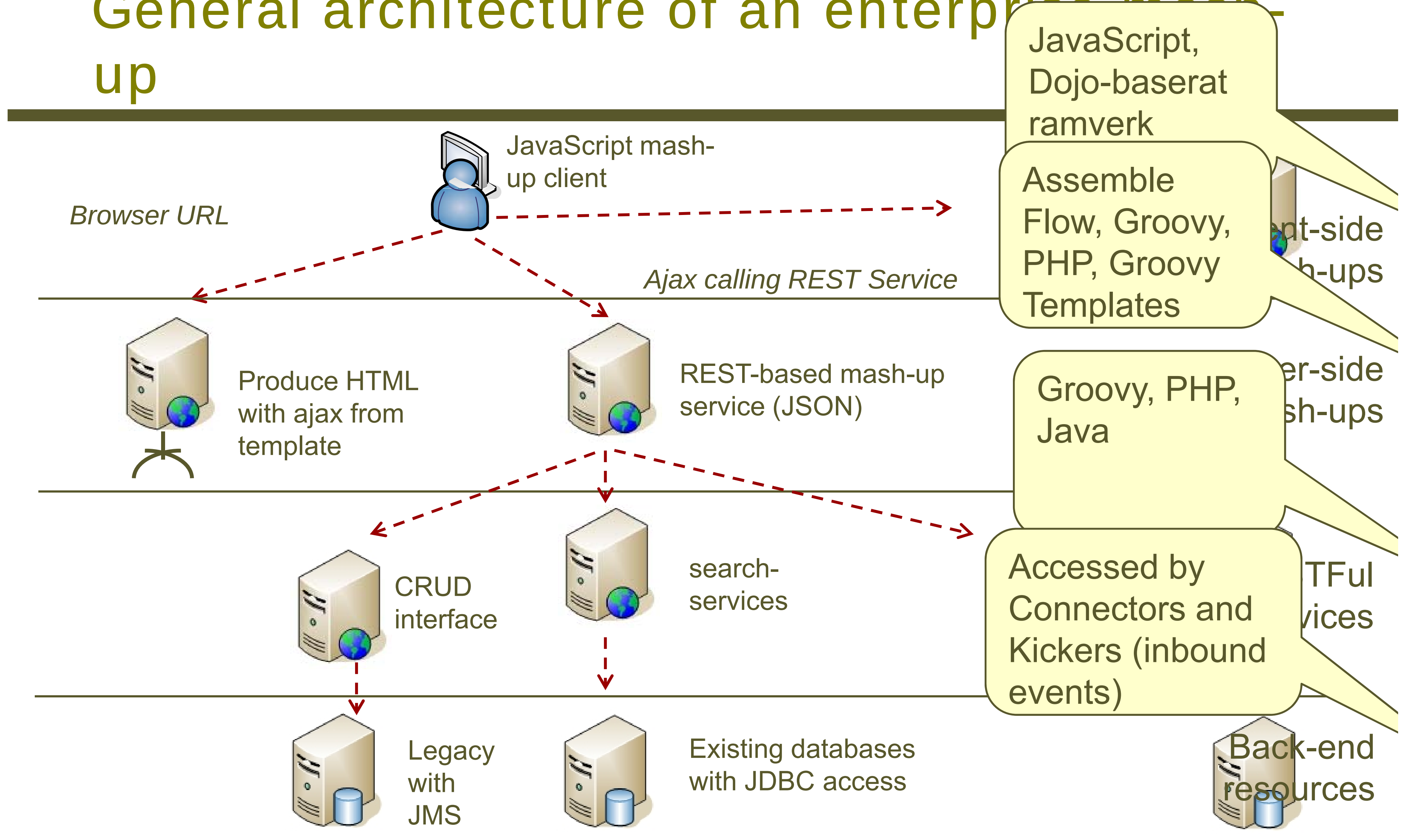
Cornerstones for Enterprise Mash-ups

- Solution Architecture
  - Produce new web resources by consuming existing.
    - Q&D(O) over Strategic & Engineered
  - 90% Integration logic, 10% Core business logic
  - Web concepts (REST services , browser scripting)
- Programming model
  - Generic formats with open binding.
    - Content before structure
  - Agile, short roundtrips
- Solution Category
  - Primarily for information consumption
  - Situational applications
- Lifecycle
  - Lifecycle not necessarily formal. Anyone depending of mash-up services is part of the game. Web URLs usually stay sable for a couple of years…
- Tooling
  - Even the tooling is a mash-up
    - Scripting, Java, resource adapters, integration middleware…

WebSphere sMash technology

- JVM-based application server used for developing and running mash-ups
- UI development and re-use toolkit for JavaScript based on Dojo JavaScript library
- Solid component management
  - The product is componentized
  - Solutions can be componentized
- It is a very lightweight container
  - bootstraps its features from an internet repository
  - Installation: Download, unzip, start
  - Initial size is 1.8 MB. Incrementally grows to 250 MB
- It is a very agile environment
  - Develop in the container
  - Basically Zero restarts: edit, refresh, debug...

General architecture of an enterprise mash-up


JavaScript mash-up client
Browser URL
Ajax calling REST Service
Produce HTML with ajax from template
REST-based mash-up service (JSON)
CRUD interface
search-services
Legacy with JMS
Existing databases with JDBC access
JavaScript, Dojo-baserat ramverk
Assemble Flow, Groovy, PHP, Groovy Templates
Groovy, PHP, Java
Accessed by Connectors and Kickers (inbound events)
nt-side
h-ups
er-side
sh-ups
TFul
vices
Back-end resources

CALLISTA

RESTFul Services in Groovy

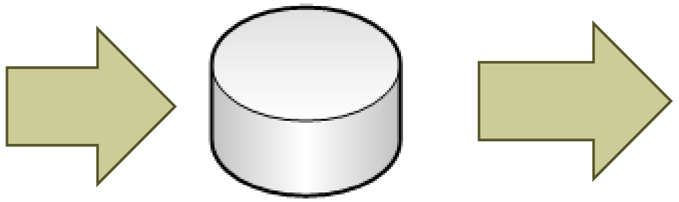
```
def onList() {
        // Http GET to http://localhost:8080/resources
        // Writes a list of all resources to the response (XML or JSON)
}

def onCreate() {
        // Http POST to http://localhost:8080/resources/registrations
        // Create new resource. Sets the location header of response to
        // http://localhost:8080/resources/registrations/<id of new resource>
}

def onRetrieve() {
        // Http GET to http://localhost:8080/resources/<id>
        // Writes the requested resource to the response (XML or JSON)
}

def onUpdate() {
        // Http PUT to http://localhost:8080/resources/<id>
        // No response
}

def onDelete() {
        // Http DELETE to http://localhost:8080/resources/<id>
        // No response
}
```

Special support for Database Resources

- Very similar top Grails
- Builds on metadata and coding-by-convention
- Basically creating a RESTFul api on top of database tables
- Dojo Grid + Detail Form also by convention

```
{
   "fields": {
      "name": {
         "label": "Namn",
         "required": true,
         "type": "string",
         "description": "",
         "default_value": "",
         "max_length": 50
      },
      "cadec": {
         "label": "Cadec",
         "required": true,
         "type": "boolean",
         "description": "",
         "default_value": ""
      },
```

Generates / Alters new database
Maps legacy



Zero Resource Model: Access using metadata

Demo - ZRM

√

- Create New Application
- Add Resource Metadata (json format)
- Add initial data fixture
- Synchronize database
- Create Groovy handler for resource

- Use RESTFul api from browser
  - http://localhost:8080/resources/registrations/1
- Query-by-convention also available
  - http://localhost:8080/resources/registrations?email__endswith =company2.com

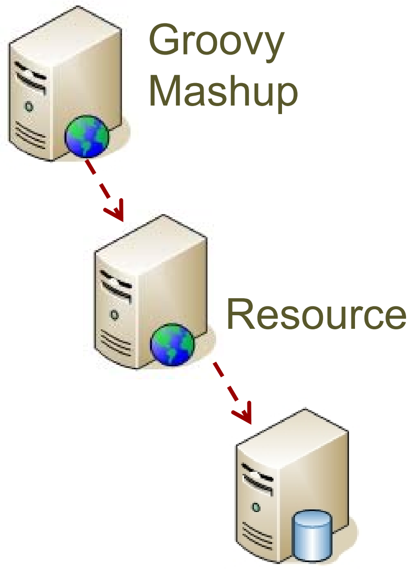
CALLISTA

ZRM – Model-driven UI

- Generate UI descriptor from ZRM model
  - Adds layout attributes, but still metadata on JSON format
- Descriptor used by UI builder
  - For CRUD user interfaces the “Rails” way, but with services

Coding by convention – Restless REST

- How to call a ZRM RESTFul service from a Groovy Mash-up script?

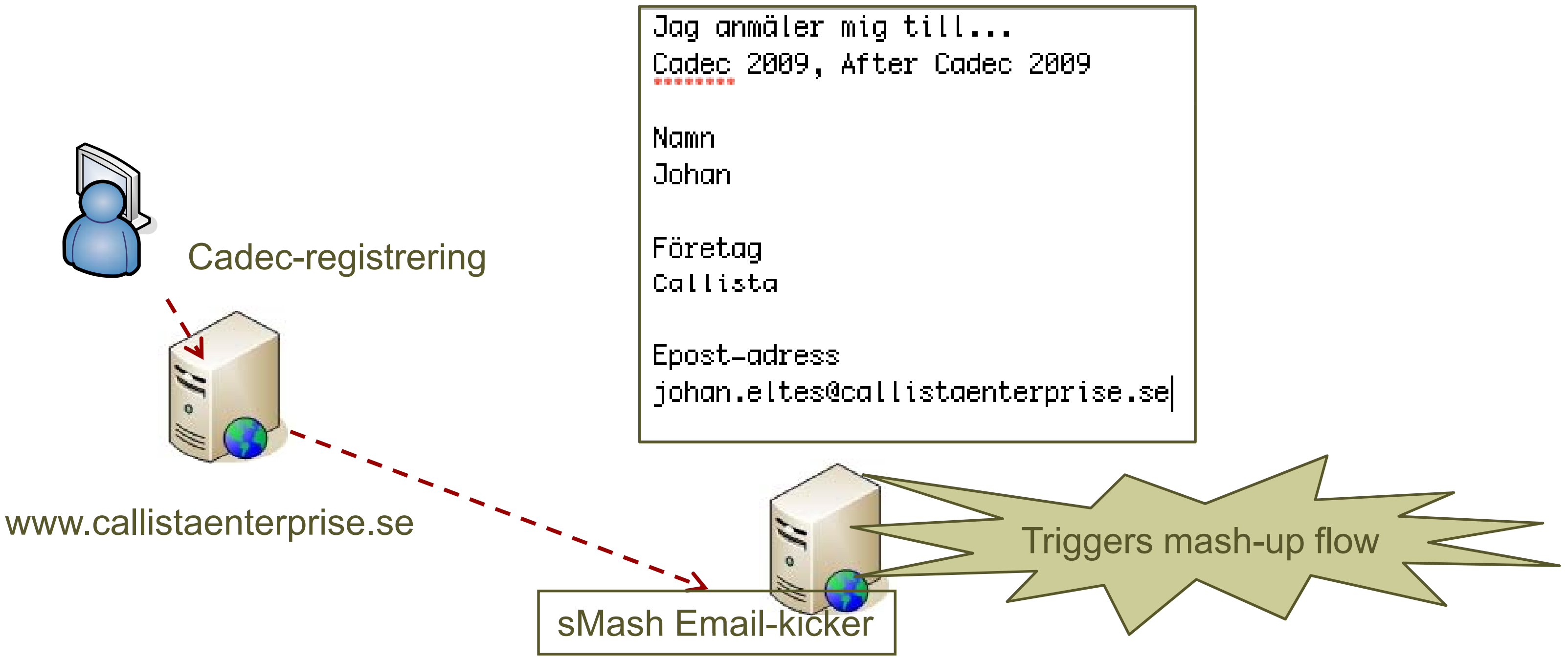


```
Type registrations = TypeCollection.retrieve(‘reqistrations')

List<Member> allRegistrations = registrations.list()

allRegistrations.each { registration ->  println registration.email}

List<Member> registrationsFromAcme = registrations.list(email__endswith: ‘acme.com')

Member newRegstration = registrations.create(name:‘Kalle', company:’Acme’,…)
```

CALLISTA

Flows – creating mash-ups graphically

- New email events should trigger creation of new resources
  - Use the POP3-kicker



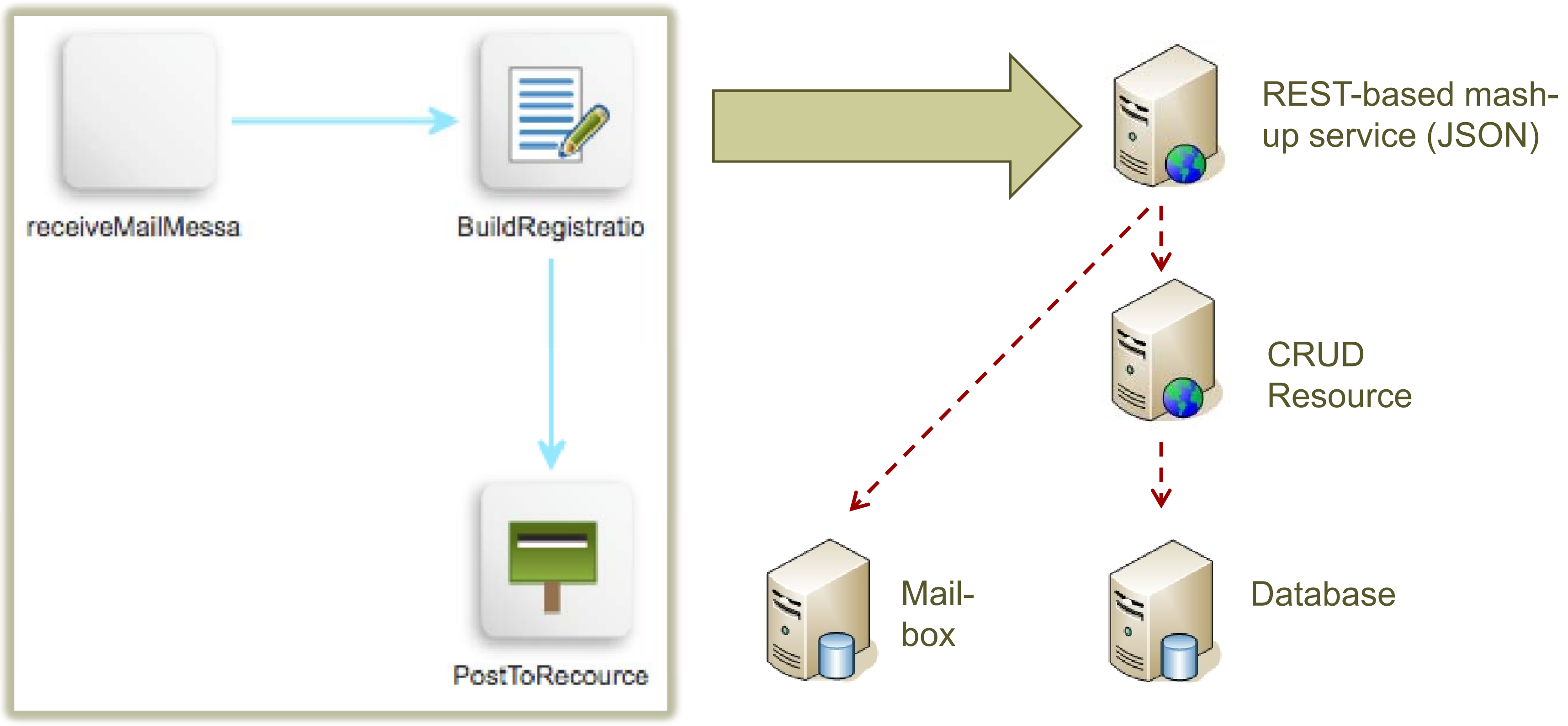
Jag anmäler mig till...
Cadec 2009, After Cadec 2009

Namn
Johan

Företag
Callista

Epost-adress
johan.eltes@callistaenterprise.se

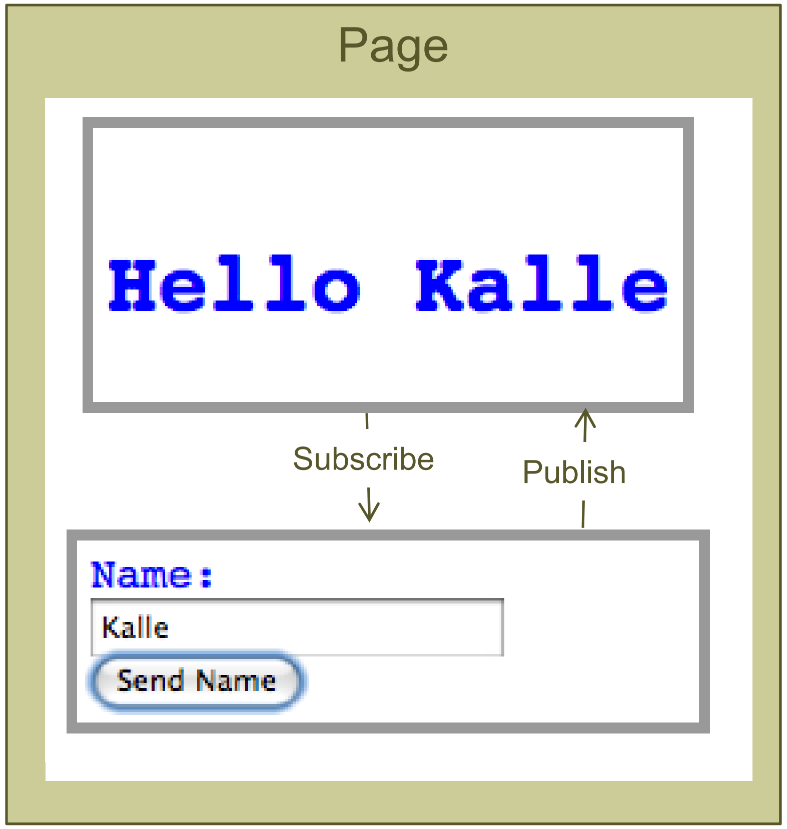
Sample mash-up flow



CALLISTA

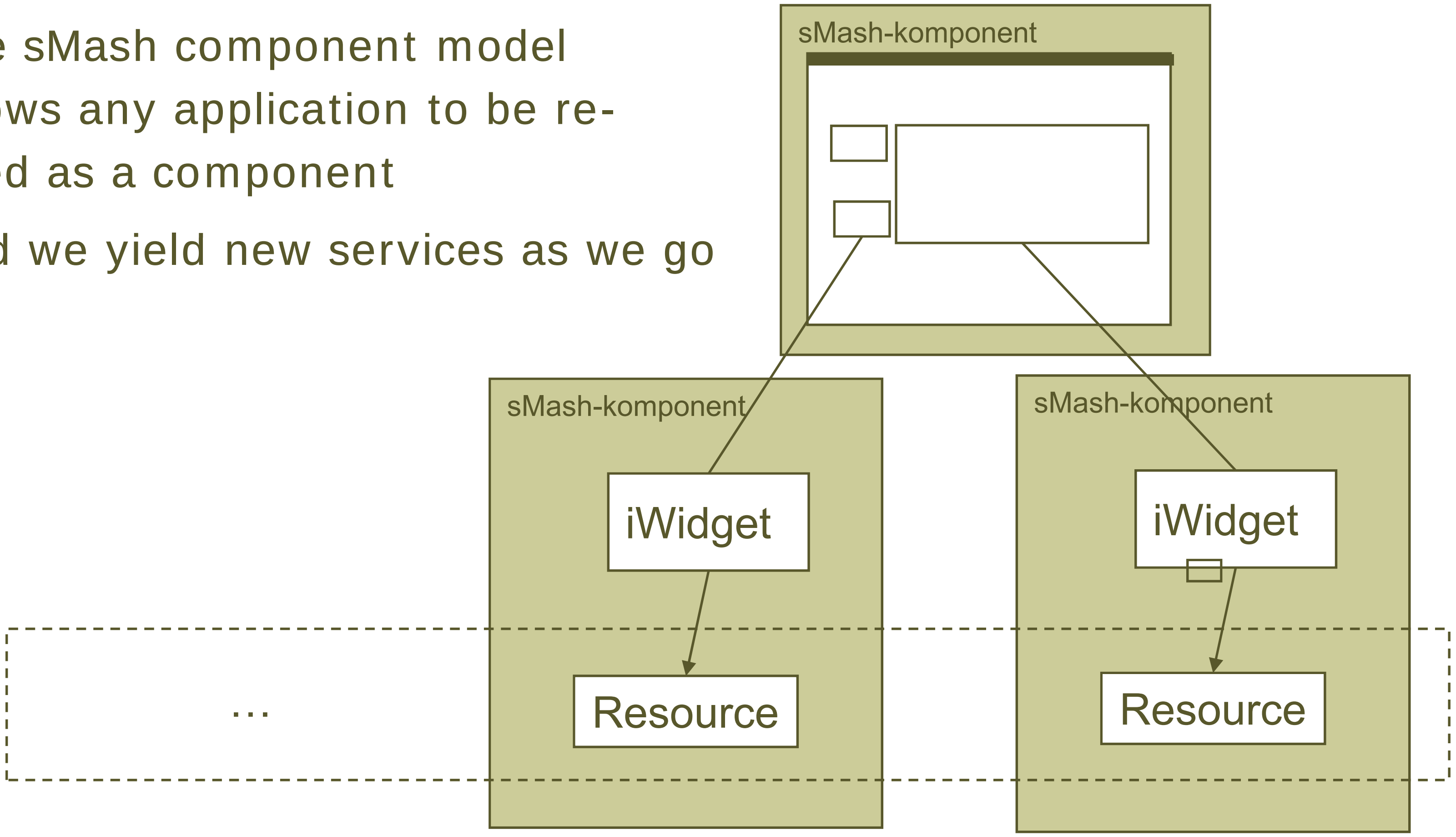
Client-side mash-ups in sMash

- iWidgets
  - Portal in the client
  - Same idea as collaborating portlets
  - Self-contained and packaged components
  - Interacts with server resources via ajax/JSON
  - Interacts with other iwidgets via client-side events
  - Multiple iWidgets composed and linked in webpages
- Would be like swing widgets, unless...



Supported by a solid multi-layer component model

- The sMash component model allows any application to be re-used as a component
- And we yield new services as we go

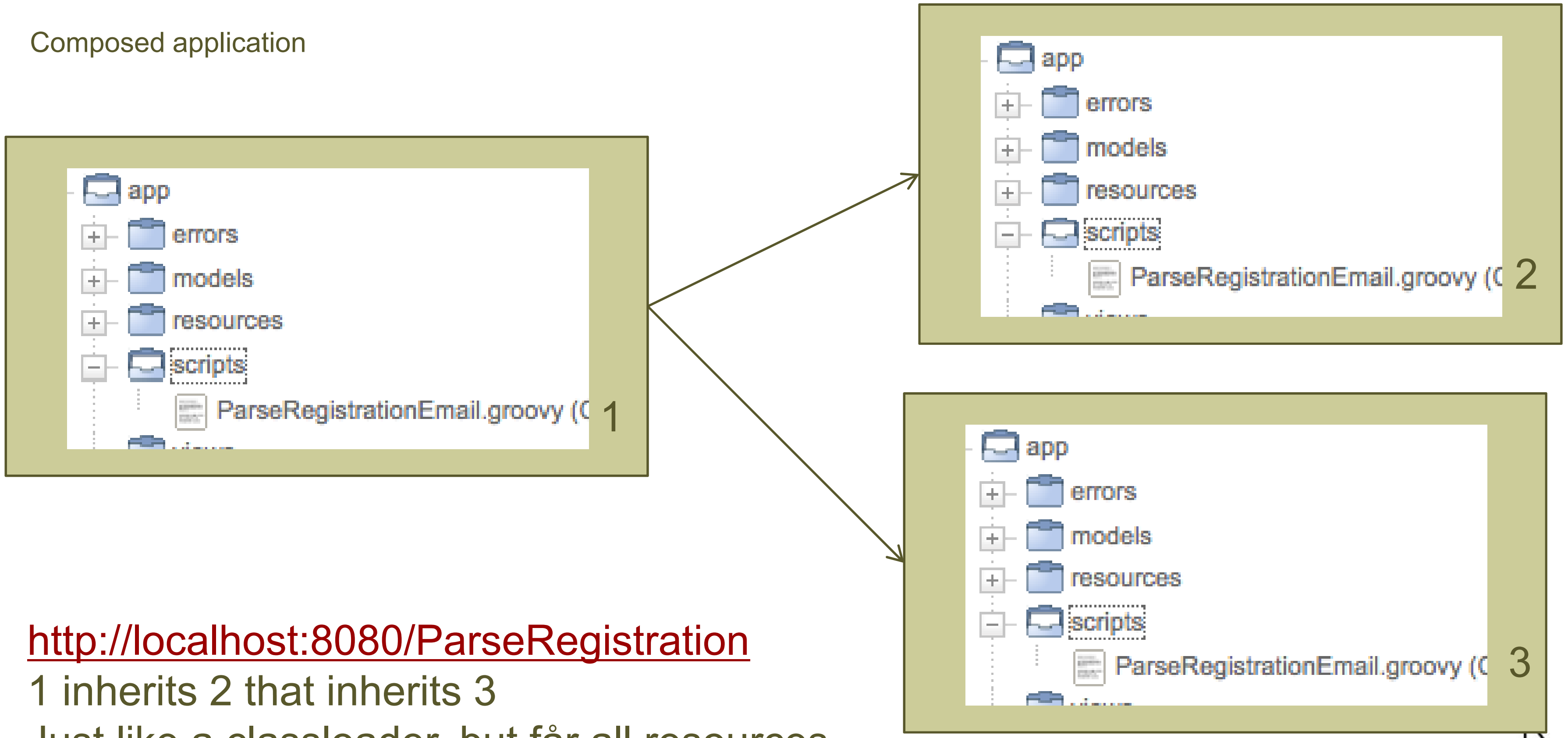


CALLISTA

Component and re-use in sMash

- All components have the same internal layout
- Dependencies are managed by Ivory

Composed application



http://localhost:8080/ParseRegistration

1 inherits 2 that inherits 3

Just like a classloader, but får all resources

Summary sMash

- Developed by IBM in a way very much unlike IBM
- Full access to Project Zero development process and development products
- Based on what seems to be the comming ”main stream” technologies för ”Web 2.0” and Cloud Computing
- A strange mix of high-level and techie (full http protocol access)
- A slick seemless environment for small teams and situational applications
- Lack of momentum – very small community
- Coding in the browser?
  - Great debugger, no intellicense

What about Open Source?

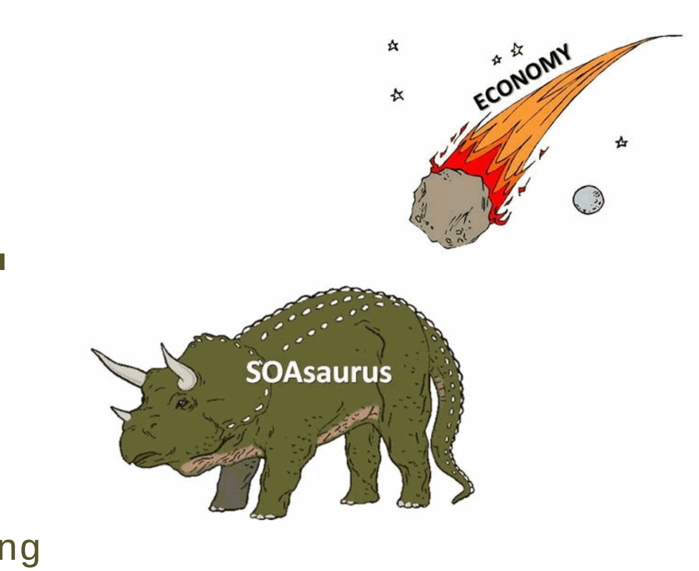
- Few products that define them selfs as Mash-up infrastructure
- WSO2 Mashup Server
  - Limited to JavaScript
  - JavaScript lacks libraries for server side integration
  - Primary focus is on accessing and publishing WebServices with JavaScript
- Several Mash-up systems in the cloud
  - Yahoo Pipes probably best known

Mash-up development skillset

- REST architecture (Resource model -> HTTP)
- JavaScript and JavaScript widget frameworks
- Server side dynamic languages (Groovy, PHP)
- Major REST-based application protocols (ATOM, RSS)
- Web Security

CALLISTA

Conclusions



- WebSphere Smash
  - A tailored infrastructure boosts RESTFul development
  - It represents a balanced model for agility and engineering
  - Agility and integration at this level has a price: Vendor lock-in (OS may catch-up)
- WSO2 Mashup Server
- General – Enterprise Mashups
  - Q&D(O) – Probably!
  - Intersects with integration, process and information management middleware
  - Unique in its dedication to the architecture of the web