



This Presentation Courtesy of the
International SOA Symposium
October 7-8, 2008 Amsterdam Arena
www.soasympoium.com
info@soasympoium.com

Founding Sponsors



Platinum Sponsors



Gold Sponsors



Silver Sponsors



Enterprise Mashups with SOA

SYSTEMATIC THOUGHT LEADERSHIP FOR INNOVATIVE BUSINESS



Ümit Yalçinalp
Shel Finkelstein
SAP Research, Palo Alto
umit.yalcinalp@sap.com
shel.finkelstein@sap.com

SAP RESEARCH

THE BEST-RUN BUSINESSES RUN SAP



Outline



- Mission
- The Changing Computing Landscape
- Composition Characteristics
- Enterprise Mashup Application Platform
- Recent SAP Research Work
- Conclusions

Mission:



- Simplify creation, delivery and execution of enterprise apps
 - Emerging technologies for the Web, dynamic languages and programming environments
- *Advanced Web Technologies group at SAP Research, OCTO*
 - Ümit Yalçinalp, Shel Finkelstein, Tilman Giese, Rama Gurram, Brian Mo, Matthias Kunze, Jennifer Baldwin, Anne Hardy ...
- This talk is about trends & experiments, not product directions

This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. SAP assumes no responsibility for errors or omissions in this document. SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material.

Typical Application Architecture



- Presentation Layer
- Application/Business Logic Layer
- Data Access Layer

Traditional Web Applications



- Client
 - Displaying UI
- Web Server
 - Generation of presentation
- Midtier
 - Data integration
 - Composition
 - Business Logic
 - Validation
- Database



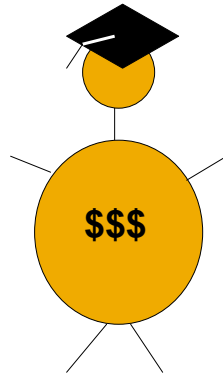
Today

Not (Client.isDumb()) and Not (Client.isThin())

New Clients



- Memory++ and processing power++
 - 128MB RAM and 620MHz CPU for smartphones
- Browser as an app platform **Plug-in as a Platform (PaaP)**
- Empowered by **Dynamic language runtimes** and proprietary APIs (e.g., Microsoft Silverlight, JavaScript libraries)
 - Client generates dynamic presentation layer
 - Client may include preliminary logic (e.g. validation)
- Deployment of same **controls** in both **standalone runtimes and browsers** (e.g., ADOBE AIR)
- Rich client controls (video, sounds, etc.) in applications
- **Local data cache** management by client apps (e.g., Google Gears)

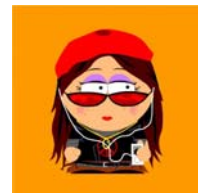


Wealthy Client

The Pendulum Swings Back...



- Client/Server concepts are back: Refresher for your memory
 - Clients responsibilities may include
 - presentation layer creation/management
 - execution of some business logic
 - integration
 - Middle Tier handles integration, transformation, dispatch, connectivity
 - Backend Servers provide services that are primarily data oriented
- Responsibilities on either client or server
 - composition
 - execution of business logic
 - events/messages



<http://www.sp-studio.de>

What are the Implications for Enterprise Applications?



- Use client as a real SOA tier
 - Offloads server
 - Crucial for user experience
 - Composition
- Design services to exploit wealthy clients
 - Utilize their characteristics and full potential
 - Make integration possible
 - Distribute application responsibilities
- Enterprise applications require new distribution of responsibilities across
 - Development
 - Assembly and composition
 - Deployment
 - Execution

Where are these capabilities?
They may be in the client, server or both

Goals for Components and Compositional Frameworks



- Must target the following
 - Flexible reuse
 - Fast development, composition and change
- Utilize rich metadata
- Supply contracts for development, composition, and deployment
- Define different skills/roles for different tasks
- May provide binding to different environments
- May be assembled to a composition by
 - Simple syntax/templates
 - Declarative languages/operators

Examples

- Unix pipes
- Yahoo pipes
- Relational queries
- SCA
- EJB

Composition Trends for Enterprise Apps



- Three Approaches:
 1. Mashups
 2. Composite Applications
 3. Service Platforms and Software as a Service (SaaS)

- Taxonomy for Enterprise Applications

For more info on 2, 3 and the taxonomy, see:

<http://events.oasis-open.org/home/sites/events.oasis-open.org/files/MashupsForCompositeEnterpriseApps.pdf>

Mashups Today



- Driven by Client side development
- Composition is handled on the client
- Lightweight and rapid development style
 - Scripting and dynamic languages (JavaScript, Ruby)
 - Component metadata
- Browser as a container enables rich client
 - *Plug-in as a Platform* enables wealthier client

Two Styles of Client-side Composition



- Composition with APIs
 - **Development Model:** Use APIs to supply client-side components with server-side data
 - **Execution Model:** Run components on client; fetch (async pre-fetch) data from server
(Google Maps)
- Composition with events
 - **Development Model:** Compose using events and metadata
 - **Execution Model:** Run components connected by pub/sub events
(OpenAjax Hub, SAP Research Enterprise Web Widget Framework, IBM QEDWiki)

Not mutually exclusive

- **Data services** for data aggregation and integration
 - Data Access (get, update, delete, insert, replicate,...)
 - Data API (Google Maps API)
 - Asynchronous Updates (AJAX)
 - Routing for multiple domains
 - Lifecycle beyond service interaction (SQLite, Google Gears)
- **Pub/Sub event services (on Client)**
- **Metadata services (from Server/from Client)**
 - Component and composition descriptions
 - Repository access

We are entering a new world where
data may be more important than software

Tim O'Reilly

Data-centric Services



- Focus on CRUD (Create, Retrieve, Update, Delete)
- Expose logical view of data
 - Can be coarse granular, aggregated or composed from distributed backend services and sources
 - Provide a unified facade for client consumption
 - May have explicit relationships among themselves
- Well aligned with **REST**
 - Data-centric design yield identifiable data resources
 - Resources can be identified by URIs
 - Resource Oriented Architectures does not need to clash with WS centric SOA
- Where is the application logic?
 - Can be implemented on client, server, or both

An Emerging Trend: Are Server-side Compositions also Mashups?



- Create a new service from existing services
 - Use public APIs or interfaces to link services (WS/data-centric interfaces)
 - Expose new service with standards
- Server-side mashups
 - Resemble traditional composite applications
 - Use scripting/lightweight composition
 - Avoid cross-domain composition issues
(Addresses browser-specific problem)
- Data-centric composition is still the focus
 - Use WS based composition to create/expose a new WS
 - WS Orientation does not contradict REST aligned design
 - Works when composed services do not address presentation
- Examples: WSO2, JackBe

Mashups are Successful



- Achieve “Goals for Components and Compositional Frameworks”, described on Slide 12
- Lightweight development
 - Scripting
 - Dynamic languages
- Key enabling services:

Server side data services +
Consumer side events +
Metadata repository

Trends for Composition



- **Metadata** enables composition, management and flexibility
 - **Component Descriptions**
 - Interfaces
 - Events published/subscribed to
 - Backend data sources
 - Cross-tier deployment capabilities
 - Requirements for composition
 - **Policies**
 - Authorization
 - Protocol bindings
 - Composition policies
 - Cross-tier deployment requirements
 - Quality of Service
 - **Composite application metadata**
 - Description of composite services
 - Description of composite data
 - Events/message flows
 - Derived policies and imposed policies for composites
- **Examples:** Component, Assembly and Deployment Descriptors for SCA and JEE; Widget Frameworks

■ **“Stateless” middle tiers**

- Mediation of service access, protocols, data representation, identity and other security
- Handling of composition and intermediation (broker, mediator, gateway, aggregator, etc.)
- Caching simply for performance optimization

■ **End-to-end cross-tier deployment and optimization**

- Leverages declarative metadata describing logic, data, components and compositions
- Enables different execution model for different client capabilities
- Allowing logic to run in different tiers
- Manual optimization → (semi-)automatic optimization?

- Research platform for cross-tier compositional apps
- Focus Areas
 - User experience and presentation
 - Data and service management
 - Business logic
 - Events and messages
 - Backend adaptation
- Dynamic languages for creating, consuming services
- Metadata centric, enriched with metadata services
 - Widgets
 - Property based composition
 - Pub/Sub
- Ultimately cross-tier optimization is our goal

- Taxonomy for Composite Apps/Mashups
- Data Services are well aligned with REST
 - Do not need protocol transformation, extra binding
 - Simplicity is very appealing
 - JSON serialization enables fast data binding
 - Used for both data and metadata services
- Client side Pub/Sub eventing enables lightweight composition
 - More flexible than **FIXED** interactions between components
- Property driven composition is declarative, easy to manage
- “Zero deployment” client is desirable

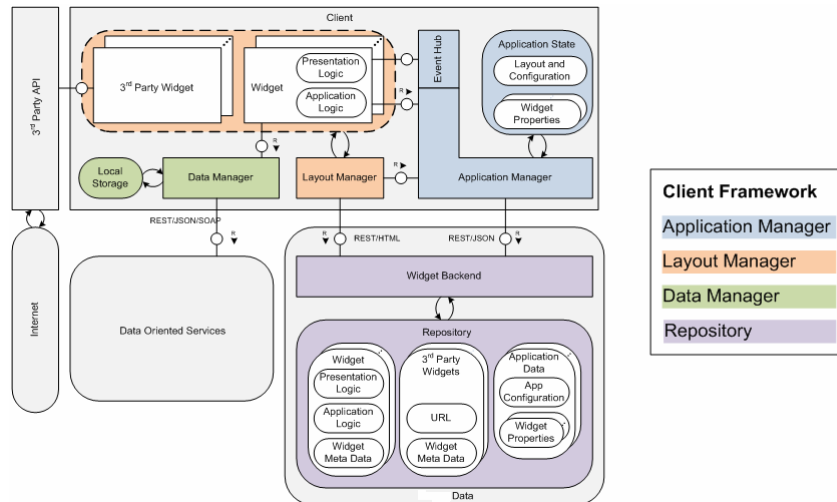
The screenshot shows the Bijoux Client application running in Mozilla Firefox. The interface is divided into several panels:

- Widget Repository:** A tree view on the left showing various widgets like Event Hub Debugger, Widget Editor, Theme Switcher, Contact Management, Contact Browser, Contact Search Quick, and Games.
- Contact Search Quick:** A central panel with a query editor. The query is set to "bijoux.contact_selected". Below it is a "Contact Browser" table with columns for First Name, Last Name, Email, and Telephone. The table contains two rows of data.
- Event Hub Debugger:** A panel on the right showing a table of events with columns for Topic and Data. The data includes "CMContactBrowser", "Jensen", "Yalc", and "CMSearchQuick".
- Widget Editor:** A panel on the right showing the XML configuration for the "Contact Search Quick" widget. The XML includes properties like name, scope, category, height, and a script for the search field.
- Third Party Apps:** A panel at the bottom left showing a list of tasks: "1. Contact Dev Team", "2. Prepare Talk", and "3. Type new task here".
- WeatherRadar:** A panel at the bottom right showing a weather map of the United States.

Yellow boxes with arrows point to the following components:

- Repository & Metadata based Management:** Points to the Widget Repository.
- Composite App:** Points to the Contact Search Quick and Contact Browser panels.
- Event Hub for Composites:** Points to the Event Hub Debugger.
- Third Party Apps:** Points to the Third Party Apps panel.

Client Perspective of Mashup



Conclusion



- Composite applications and mashups share a common taxonomy
 - Concepts move between tiers
 - Clients such as browsers are a full-fledged tier
 - Dynamic languages enable lightweight composition (client and/or server)
 - Metadata driven component and composition frameworks are more adoptive
 - In today's mashups data-driven services are central to composition
 - Client-side event bus enables composition
- End-to-end composite apps, deployed for cross-tier execution, are the next wave
 - How to distribute logic is not a finalized question
 - Enterprise qualities remain critical
 - Security, integrity, scalability, performance, availability, manageability, ...
 - Cross-tier optimization is an exciting future direction

References



- Rama Gurram et. al. : *A Web Based Mashup Platform for Enterprise 2.0*
<http://www.springerlink.com/content/0442382328227386/>
- Jonathan Marsh: *Mashup: Noun or Verb?*
<http://wso2.org/repos/wso2/people/jonathan/Mashup%20Noun%20or%20Verb.pdf>
- Charlton Barreto : *Web 20-20 Architecture for the New Internet*
<http://charltonb.typepad.com/talks/120407-cbb-web2020/Web2020TheNewInternet.pdf>
- Pat Helland: *The Irresistible Forces Meet the Moveable Objects*
<http://blogs.msdn.com/pathelland/attachment/7082107.aspx>
- Alistair Barros: *The Rise of Web Service Ecosystems*
<http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/mags/it/&toc=comp/mags/it/2006/05/f5toc.xml&DOI=10.1109/MITP.2006.123>

Enterprise Mashups with SOA



Ümit Yalcinalp
Shel Finkelstein

SAP Research, Palo Alto
umit.yalcinalp@sap.com
shel.finkelstein@sap.com