

Rich Internet Applications (RIA)

Contents:



- Web Evolution
 - "2.0"
- Rich User Experience
 - AJAX
 - Open Laszlo

Mariano Cilia / cilia@informatik.tu-darmstadt.de

1

The Web as Platform: Netscape vs. Google

- Netscape
 - Flagship product was the web browser
 - Strategy: dominate the client-side to establish (and sell) server-side products
 - Webtop to replace Desktop with information updates, applets, ...
 - Information providers would purchase Netscape servers
- Both web browsers & servers are today commodities
 - The value moved up to services delivered over the web platform

[O'Reilly'05]

2

The Web as Platform: Netscape vs. Google (cont.)

- Google: a specialized Database!
 - Native web app
 - deliver services
 - customers paying (in)directly for their use
 - "no" software releases, just continuous improvement
 - No porting to different platforms
 - Massively scalable collection of PCs running Op-So OS + own apps (database management) and utilities (crawlers)
 - Without the data, the tools are useless, without the software, the data is unmanageable

3

The Web as Platform: DoubleClick vs AdSense

- Web Publishing
 - DoubleClick over 2000 big sites as customers
 - Internet dominated by top web sites
 - Formal sales contract
 - Publisher/ad-agency friendly advertising formats (banners, pop-ups)
 - AdSense (& Overture) serve hundreds of thousands
 - Ad placement on virtually any web page
 - Minimally intrusive, context-sensitive, consumer-friendly text advertising

4

The Web as Platform: DoubleClick vs AdSense (cont.)

- The **long tail**:
 - Collective power of the small sites that make up the bulk of the web's content
- Other stories
 - eBay: enables occasional TXs of only a few dollars, acting as automated intermediary
 - Napster: architected a system where every downloader also became a server

5

The Web as Platform: Akamai vs. BitTorrent

- Scalability aspects
 - Akamai: business with the big companies
 - Replication of high-demand sites
 - Revenues from those sites to benefit individuals
 - BitTorrent: Internet decentralization
 - Every client is also a server
 - The more popular the file, the faster it can be served
- New principle: The service automatically gets better the more people use it

6

Collective Intelligence

- **Yahoo!** born as a catalog of links
- **Google's** PageRank exploits the link structure of the web rather than the document characteristics
- **eBay** is based on the collective activity of its users
- **Amazon** uses "most popular" suggestion

7

Collective Intelligence (cont.)

- **Wikipedia** is an experiment of trust since any web user can add or modify an encyclopedia entry
- **Cloudmark**, a collaborative spam filter, aggregates the individual decisions of email users
- **Viral marketing**: recommendations propagating directly from user to user
- **Peer-Development**: the Internet infrastructure (linux, apache, mySQL, PHP, Perl, ...) relies on open-source (collective, net-enabled intelligence)

8

Architecture of Participation

- Users add value
 - But just a small % of users add value explicitly
 - Comments on books, electronics, ...
 - Ciao, amazon, guenstiger, ...
 - Aggregating user data and building value as a side-effect of ordinary use of the application
 - Amazon's suggestion: buyers of this item also bought ...
 - (systems get better the more people use them)

9

The data is key

- The race is on to own certain classes of core data
 - Location, identity, calendaring of public events, ...
 - In many cases, significant cost to create the data
 - In others, try to reach critical mass via user aggregation (and turn aggregated data into a system service)
- Trend: similar to the free software movement, proprietary databases becoming free (free data movement)

10

End of Software Release Cycle

- Internet era: software delivered as service (not as a product) leading to
 - Operations must become a core competency: continuous maintenance
 - Expertise in product development + expertise in daily operations
 - Google's sys admin, networking, load balancing techniques are top secrets!
 - Scripting languages (Perl, Python, PHP,...) playing a key role
 - Users must be treated as co-developers
 - Open-source from "release early and release often" to "the perpetual beta"
 - New features on a monthly, weekly basis!
 - Real time monitoring of user behavior to analyze which features are used, how they are used.
 - Improve and/or remove features

11

Lightweight Programming Models

- Large companies rely on complex web services stack
 - Creation of highly reliable programming env for distributed apps
- But INet apps adhere to those formalisms (WS) but also provide XML data over HTTP (XMLRPC, lightweight approach)
 - Amazon relies on WS for B2B but reports a 95% usage of XMLRPC

12

Lightweight Programming Models (cont.)

- Support for lightweight Prog models allow loosely coupled systems
 - more fragile but fine
- Think syndication, not coordination
- Design for "hackability" and remixability
 - The most successful web services are those that have been easiest to take in new directions (unimagined by their creators)
 - "some rights reserved"
 - Innovation in Assembly
 - When commodity components are abundant, you can create value simply by assembling them in novel or effective ways

13

Rich User Experiences

- The web used to deliver "applets" and other kinds of active content within the web browser
 - java applets
 - JavaScript and DHTML as lightweight client-side programmability
 - Flash
- The potential of the web to deliver "full scale apps" appeared with GMail, followed by Google Maps
 - All these relying on AJAX
 - Web developers finally able to build web apps as rich as local PC-based apps

14

Web Evolution by Example

- | ■ Web 1.0 | → | ■ Web 2.0 |
|---------------------|---|------------------|
| ■ doubleClick | → | ■ AdSense |
| ■ Akamai | → | ■ BitTorrent |
| ■ Mp3.com | → | ■ Napster |
| ■ Britanica Online | → | ■ Wikipedia |
| ■ Personal websites | → | ■ Blogging |
| ■ URL speculation | → | ■ Search engines |
| ■ Screen scraping | → | ■ Web services |
| ■ Publishing | → | ■ Participation |
| ■ CMS | → | ■ Wikis |

15

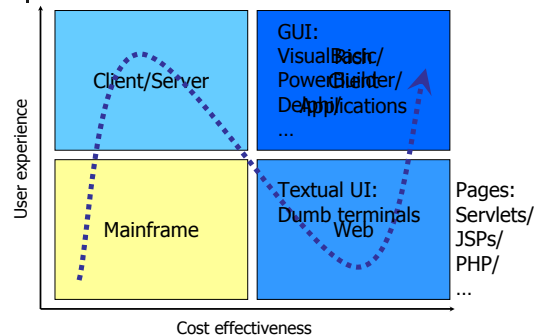
Wrap-up of 2.0

- Services, not packaged software
 - APIs
- You control your own data!
- Architecture of participation
 - Blogs, Wikis
- Trust your users (Users add value!)
 - User as contributor: pageRank, eBay reputation, Amazon reviews, Wikipedia, ...
- The Long Tail
 - Software that gets better the more people use it
 - Napster, BitTorrent
- Some rights reserved
 - Design for hackability and remixability
- The perpetual beta
- Cooperate, don't control
- Rich user experience
 - GMail, Google Maps, Yahoo! Maps, Zimbra

16

Rich User Experience

User Interaction



18

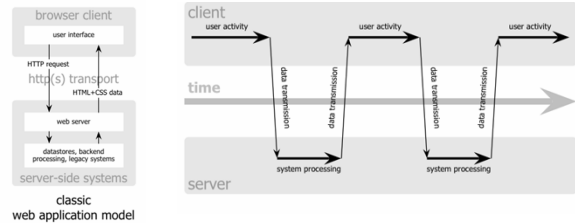
Classical Web Applications (1)

- Delivery means server-based, centrally manage deployment
 - Cost-effective
- Frees users to configure, install application updates
- Applications are virtually available anywhere

19

Classical Web Applications (2)

- User completes and **submits form**
- to a **web server** (user waits)
 - Sends a new **web page** as a response



20

Classical Web Apps (cons)

- Poor user experience in contrast to traditional locally running client code
 - Inferior user experience
 - A step back in the evolution of user interfaces
- User is blocked waiting for server response (new web page)
- Clients need to refresh and re-render the complete HTML page even if a small change is needed
- The advantages of web app outweigh the flaws
 - So end-users pay the price!
- Web app development is based on lowest common denominator

21

From Web to RIA: 3 Dimensions

- A new application distribution technique
 - Browse to an application, rather than install an application
 - Significant implications on support and updates
- A new development methodology
 - Markup meets object-oriented programming
- A new user experience
 - Best of Web
 - Best of GUIs

22

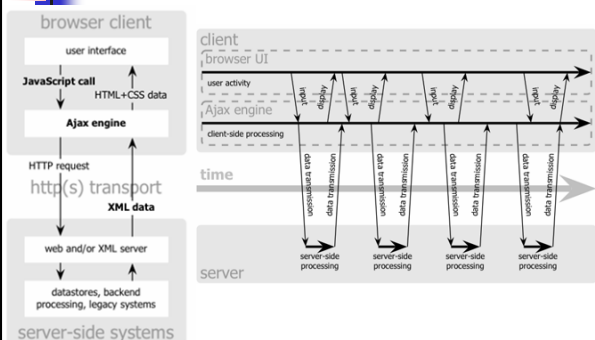
AJAX: Async. JavaScript and XML

- Set of technologies
 - Standards-based presentation using XHTML and CSS
 - Dynamic display and interaction using DOM
 - Data interchange and manipulation using XML and XSLT
 - Asynchronous data retrieval using XMLHttpRequest object
- JavaScript binding everything together

[AJAX]

23

AJAX Web App Model



24

AJAX - Characteristics

- Focus on a client fetching an HTML document (DOM) that acts as a container into which to inject content
 - this is done based on client events (UI)
 - mouse over, clicks, focus changes, timer, ...
 - using XML data retrieved from server-side components (XMLHttpRequest)
 - the DOM is modified accordingly
 - only the modified piece of the web page is updated/rendered on the browser
- Page-specific control logic embedded as JavaScript code

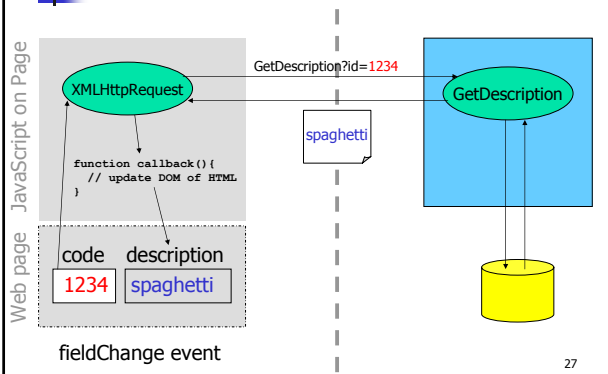
25

XMLHttpRequest is the key!

- Not new!
 - introduced by M\$ in IE years ago
- Not part of the JavaScript specification
- Nonetheless implemented in IE, Firefox, Safari, ...
 - Small differences by creating XMLHttpRequest object

26

AJAX - An Example



AJAX - Typical uses

- Online form data validation
 - e.g. userIDs, Serial Numbers, Codes, ... that requires server-side validation
- Autocompletion
 - e.g. email addresses (from contact database), ...
- Master-detail operations
 - e.g. an invoice form by fetching the description of codes of items
- Refreshing data on the page
 - e.g. Latest news (even relying on polling) does not need to reload the whole page

28

AJAX - Pro

- Data can be manipulated without having to render the entire page again
 - Quickly response to many user interactions
 - Avoids sending unchanged info across the network
- No browser plug-in required
 - Lightweight library can be loaded
 - Embedding <link> or <script> tags into web pages
- Better user experience

29

AJAX - Cons

- Usually no clear feedback during requests
 - User might experience (unexpected) delays
- Involve complex JavaScript code on the client
- Complexity now on the generation of web pages that embed complex JavaScript code
 - Design and Maintainability!
- Debugging
- Viewable Source code
 - Open your apps to hackers

30

Examples of AJAX

- Google:
 - Mail, Suggest, ...
- Zimbra Calendar:
 - <http://www.zimbra.com/>
- OpenRICO
 - <http://www.openRICO.org>
- BACKBASE
 - <http://www.backbase.com/>
- ARIWARE Open Source
 - <http://ariaware.com/products/arp/>

31

OpenLaszlo

- Free open source platform for RIA
- Applications written in LZX
 - XML + JavaScript
- Object-oriented development and data binding
 - Compiled/transformed into SWF (flash)
- Portable (relies on Flash)
 - Flash ubiquitous plug-in on browsers
 - Also runs outside a browser
- Supports rich media
- Firewall friendly
 - SOAP, XML-RPC, http,...

[Laszlo]

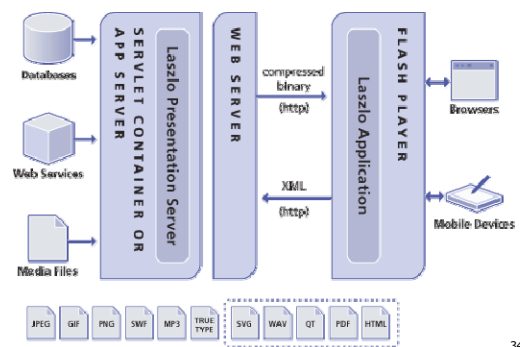
32

OpenLaszlo offers

- Direct manipulation
- Windowing
- Keyboard control
- Double-click
- Menus
- Drag-and-drop

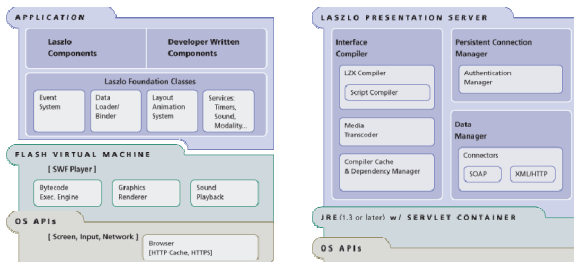
33

OpenLaszlo – General Architecture



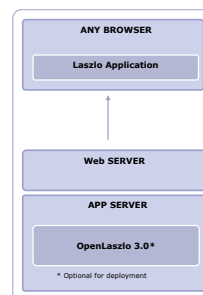
34

Client and Server (layers)



35

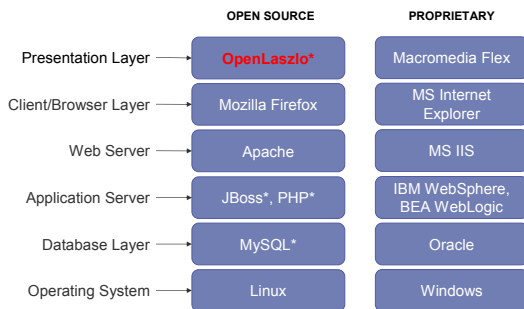
Platform for RIA



- OpenLaszlo 3.0
 - Open source (CPL)
 - Third-generation, proven technology
 - Standards-based XML-native language (LZX)
 - Deploy in "any" browser (plug-in!)
 - Ideal for rich client solutions
- New Features
 - SOLO (OpenLaszlo servlet is optional)
 - Unicode Support
 - Drawing API for dynamic visualizations
 - Dynamic Libraries
 - Optimizations for Flash 6 and Flash 7

36

Laszlo: Open source software provider at top of 'stack'



* Received VC funding in 2003/2004/04

38

UI: The End of Page Refresh?

Delivered into a Web browser without installation, and offers:

- Direct manipulation
- Declarative (vs. Procedural) Development
- Data Binding
- (Fully) Object-Oriented
- Scripting
- Event-Driven
- Pre-built Components
- Media
- Powerful Constraints System
- Keyboard/Mouse Control
- Drag-and-drop
- Animation
- Drawing API
- Layout
- Browser Integration

39

LZX: Laszlo's XML Application Description Language

- Client independent tags and APIs
- Interface with server via XML over HTTP, SOAP, XML-RPC, and Java RPC
- Runtime constraint system
- Hierarchical data binding with XPath
- Media, streaming support
- Extensible UI component framework

40

Demos

- <http://www.laszlo.com/partners/support/demos/>

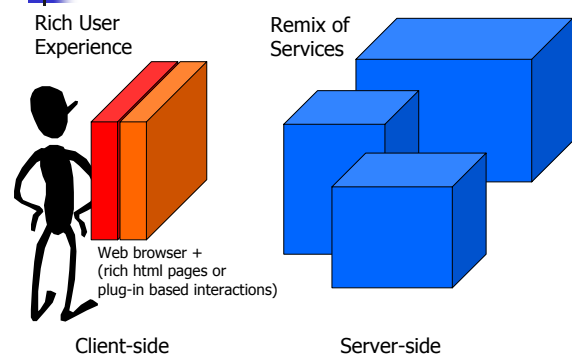
41

OpenLaszlo: Looking ahead

- Non-SWF runtimes
 - DHTML/AJAX, Java under investigation
 - Efficient device client (mobile/set-top box) under investigation
- Offline client
 - Synchronized data store
 - Move apps out of the browser
 - Desktop integration
- Working toward standardization
 - Incorporate CSS, XForms

42

New Kind of Applications



43

New Kind of Apps: Examples

- Amazon's API:
 - <http://www.amabuddy.com/>
 - <http://www.francisshanahan.com/zuggest.aspx>
- Google Map API:
 - <http://www.flashearth.com/>
 - <http://maps.habitamos.com/>
 - <http://www.gtraffic.info/>
 - <http://nycsubway.eyebearresearch.org/>
 - <http://www.acme.com/metro/>
 - <http://www.chicagocrime.org/>
- Yahoo Maps (beta):
 - <http://maps.yahoo.com/beta/>
- Search:
 - <http://www.doubletrust.net/>
- Combinations:
 - <http://www.2realestateauctions.com/>

44

Still a lot more to come...

- Cairngorm Open Source (Macromedia)
 - http://www.iterationtwo.com/open_source_cairngorm.html
- WidgetGallery (Yahoo!)
 - <http://www.widgetgallery.com/>
 - <http://www.konfabulator.com/>
- Live (M\$)
 - <http://www.live.com/>

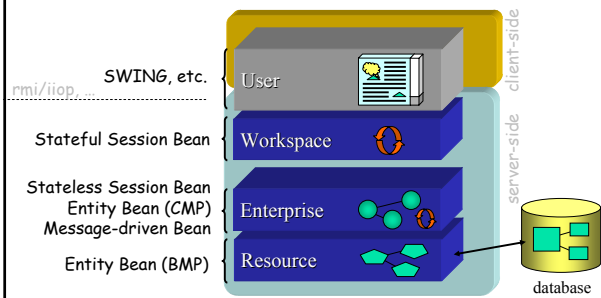
45

Fat Web Clients

- AJAX:
 - Browser: JavaScript enabled (no plug-in required)
 - Web page contains client-side code and optionally refers to some (javascript) libraries
 - Look&Feel can be changed (CSS)
 - Can fetch data from server without reloading the page
- OpenLaszlo:
 - Browser: Flash plug-in
 - Web page loads flash code that embeds the complete client-side code
 - Look&Feel embedded in code
 - Can fetch data from server without reloading the page

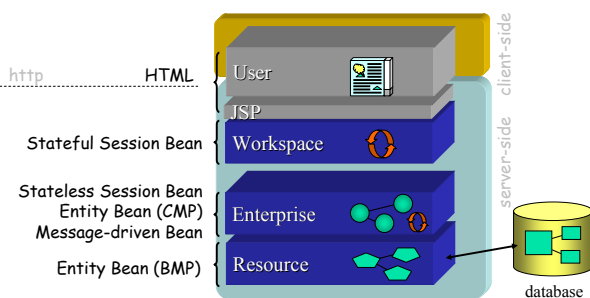
46

Multi-tier Architecture



47

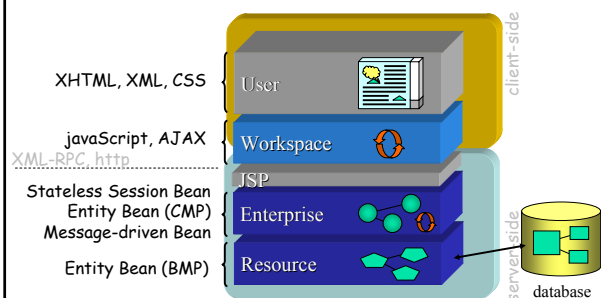
Multi-tier Architecture



48

Multi-tier Architecture

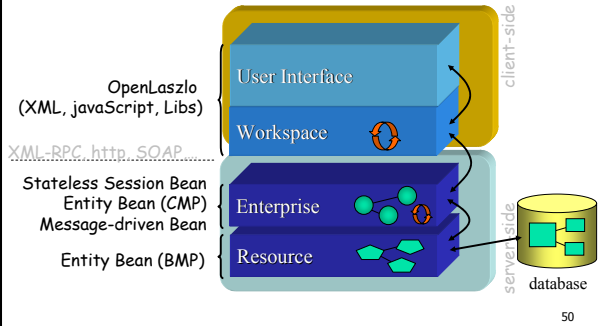
Separation of Concerns



49

Multi-tier Architecture

Separation of Concerns



Bibliography References

- [O'Reilly'05] Tim O'Reilly, What is Web 2.0, www.oreillynet.com, Sep 2005.
- [AJAX] Wikipedia on AJAX, <http://en.wikipedia.org/wiki/AJAX>
- [AJAX-Blueprints] SUN, AJAX Blueprints, <http://java.sun.com/blueprints/ajax.html>
- [AJAX-FAQ] JavaNet, AJAX FAQ, <https://blueprints.dev.java.net/ajax-faq.html>
- [Laszlo] Open Laszlo, various white papers, www.openlaszlo.org

51