The Future of Web Services Applications

Dan Wood
President, Karelia Software LLC
dwood@karelia.com

O’Reilly Mac OS X Convention
October 27-30, 2003
Orientation: Shortcomings of the Browser

- A page has arbitrary structure, layout, design
  - User must learn navigation for each site
- Minimal User Interface Elements & Feedback
- Minimal integration with system applications
- Minimal state (just cookies)
- Redundant data entry
- We’re only making slight improvements
  - Autofill, Keychain passwords
Orientation: Why Use Web Services?

- Aggregation
- Local Feel
- Better user experience outside of the browser
Orientation: Why Use Web Services?

- Incorporate related data for your convenience
  - e.g. track package via online store, not UPS
- Innovative uses of data
Past … Present … Future
The Past

• No computers!
The Past … Through the Present

• Private feeds
  – Ad-hoc protocol
  – Perhaps direct database access

• HTML Scraping
  – Publically available data intended for humans but used by computers
Problems with Private Feeds

- Reinventing the wheel
- No standards
  - Very hard to add new clients and servers
  - Hard to aggregate
- Insecure to allow direct database access
Problems with Scraping

- Reinventing the wheel
- Also no standards
- Very fragile
- Businesses don’t want their data repackaged
The Present

- Mac OS X Technologies
- Who is providing and using Web Services
- Capabilities
- Business Models
Web Services Technologies on the Mac

- SOAP from Cocoa and AppleScript
- CURLHandle
- Rendezvous APIs for local services
- Lower-level APIs like CFNetwork, XML Parser
- Server-side: Java, WebObjects, Apache Axis, etc.
A Few Companies Are Now Providing Web Services

- Some in full production
- Some experimental
  - Some piggyback off of scraping
- Noncommercial vs. Commercial
Most Users of Web Services Are Other Servers
Some Web Services clients are Desktop Applications
Many Services Still Limited

- Google: Search and spellcheck only
  - No news, images, groups
- Amazon: Search only
  - No browse, music/book preview, purchase
Limited actions

- Input: Search parameters
- Output: Result list
- Complex protocols like SOAP often not needed
- Structure not conducive to Interactivity
Business Models

- Totally Open Program
- Private Program
- Registration Required
A Totally Open Program

- Provider isn’t even aware of what the clients are
- Many experimental, unofficial feeds
- Encourages Innovation
- May not be commercially viable
- Success Story: RSS
A Private Program

- Many in existence
- Watson’s VersionTracker, PriceGrabber tools
- Easier to set up (technically and legally)
Private Program: Disadvantages

• Stifles innovation
• Often ad-hoc
• Expensive for client to use
  – Web Services doesn’t mean a client application is trivial!
  – Why should I write a tool to connect to JoesComputerStore.com?
The Middle Ground: A Registration Program

- Clients register themselves with service provider
- Commercial (free/commissioned): Amazon
- Commercial (pay): eBay
- Non-commercial (free): Google
How a Typical Registration Program Works

- Client signs a contract with service provider
- Provider gives client a “token”
- Client passes token with each transaction
- Provider can restrict client’s use
Registration: Advantages

• Client has a formal specification to work with
• Provider can track users and usage
Registration: Disadvantages

- Token must be kept private
  - Not easy when client is a desktop
  - SSL would help
- The program may cost money!
- Desktop clients can’t bottleneck traffic
eBay developer Program

- Where does the desktop client fit in?
- They might do better by *giving away* the services

![eBay Developers Program](image)

**Services / Fees**

Below is a handy chart to help you compare the service offerings and prices of our various tiers. Keep in mind that you may always upgrade your tier at a later date.

- = Included

<table>
<thead>
<tr>
<th></th>
<th>Individual</th>
<th>Basic</th>
<th>Professional</th>
<th>Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>License Type</td>
<td>Individual</td>
<td>Commercial</td>
<td>Commercial</td>
<td>Commercial</td>
</tr>
<tr>
<td>API Calls Included</td>
<td>50 per day</td>
<td>30,000 Per month</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><strong>Fees</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Annual Membership Fee</td>
<td>Free</td>
<td>$500</td>
<td>$1,000</td>
<td>$5,000</td>
</tr>
<tr>
<td>Monthly API Usage Fee</td>
<td>None</td>
<td>$2.90 per 1,000 over 30,000 limit</td>
<td>$1.60 per 1,000</td>
<td>$1.25 per 1,000</td>
</tr>
<tr>
<td>Certification Fee (per application)</td>
<td>$100</td>
<td>$200</td>
<td>$200</td>
<td>Included (1 per year)</td>
</tr>
<tr>
<td>Developer Technical Support Fee</td>
<td>N/A</td>
<td>$195 per case</td>
<td>$195 per case</td>
<td>$195 per case</td>
</tr>
</tbody>
</table>
Problems with The Present

• Not Enough Providers!
• Ad-hoc feeds are expensive in the long run
• Closed programs stifle innovation
• Desktop clients are at a disadvantage
• Business Models aren’t developed
The Future

- A Lofty Goal: To enable the user to completely avoid the web browser for many daily tasks.
How to bring more Service Providers On-Board?

- Find business models that work
- Expose sufficient functionality
- Design for the desktop as well as the server
- Conform to standards
- Build more standards
Business Models

- Free is easy if it brings the provider business
- Advertising: not just web banners
- Pay (flat or per-use) for premium services
- Different plans could expose different capabilities
Business Models

• Client could agree to display an advertisement with each transaction
• Token should be used to identify client only
• If restricted access desired, end user should establish account with service
Expose entire Web site’s functionality as service

- Searching
- Browsing/Navigation
- What’s New (News)
- Purchasing
- Account Management
- Client-stored data vs. Server-stored data
Exposé entire Web site’s functionality as service

- Provider’s Web site can be its own client! (Model-View-Controller partitioning)
Design for the Desktop

- Desktop application has much richer feature set
- Structured, quick transactions
- Clients are distributed over the world!
Conform to standards

- Use SOAP rather than ad-hoc XML (or worse)
- Use secure SOAP
- Standards make it easier for clients!
Build more standards

• More standards will make the client’s job easier
• Imagine common APIs for common functions
  – shopping carts
  – searching
  – browsing
Scenarios
The Present: Shopping on the Web

- Browser only, or assisted by Watson
- Ultimately, browser needed to make the purchase
- Experience is different on each web site
The Future: A Shopping Application

- Without browser, shop Amazon.com
- Run another App to shop at J&R Music World
- Yet another App to shop at the Apple Store
The Future: A Shopping Application

• Is a variety of desktop programs any better?
• Inconsistent user interface
• Can you trust each with your credit card number?
The Future: A Better Shopping Application

- What if a “shopping mall” protocol develops?
- A standard for browsing and buying
- One application, many stores
The Future: A Better Shopping Application

- User could compare prices among the vendors
- Purchase integrated with Address Book, Keychain
The Present: Searching the Web

- **Google** is still sometimes “shotgun”
- Many sites offer their own searches
- Searching syntax inconsistent
- Search results inconsistent
The Future: Searching the Web

• Standard searching protocol/syntax
• Search on a site, or over the whole web, consistently
• Search the sites you subscribe to
  – (Think Watson’s “Google” tool + NetNewsWire)
The Present: Reading the News

- Some sites offer full RSS feeds
- Others provide limited feeds or none
- Advertisement & subscriptions = Money
The Future: Reading the News

- Incorporate user accounts into news feeds
- Free users could view feeds with advertising
- Paid users could view feeds ad-free
- More sources would come on-line
- Browser could detect newsfeed for Web page
Will we ever abandon the browser?

- Still the easiest way to present information and simple functionality
- User doesn’t need any specialized software
- Java Web Start may be the bridge
FIN