WebViews: Accessing Personalized Web Content and Services

Juliana Freire
Bharat Kumar
Daniel Lieuwen

Database Systems Research
Bell Labs
The Ubiquitous Web

Lots of promises and expectations:

- “Mobile and wireless computing will dominate the Internet industry”, CACM, March 2001
- “More than 74 million cell phones are in use in the U.S. today, a figure that will rise to 139 million by 2003”, IDC, Nov 2000
- “The number of wireless data subscribers in the US will explode from 3 million in 1998 to 49 million in 2003 to 78 million in 2004 to 124 million in 2005.”, Gartner Dataquest, April 2001

Reality:

- “Wireless Net desperately seeking content providers”, news.com, Dec 1999
- “What is available is slow, text-based access to a relatively small number of sites”, news.com, January 2001
- “6.6 million people worldwide subscribed to Internet wireless services in 1999”, Banc of America Securities
Web pages became very complex

- Almost 90 different actions (85 links and 3 forms)
- 96 gif images
- ~113 lines of JavaScript code
- ~570 lines of HTML
Web navigation became very complex

Problems:
- many interactions are needed: 4 pages retrieved, ~400Kb transferred
- lots of irrelevant data and irrelevant choices
- data needs to be input over and over again
- this can be inconvenient from a desktop

Find lowest fares at Travelocity
Problem is worse from a PDA...

- Scenario:
  - network: Omnisky wireless data services over CDPD with throughput rates from 5-6kbps up to 12-13kbps
  - time to access flight list: 30-80secs (xfer only)
  - screen size: 160x160 pixels on a 6x6cm surface
  - input: pen-based

- Try to access Travelocity.com:
  - impossible: ProxiWeb and AvantGo can't handle required features
  - very slow: BrowseIt gets there, but after many minutes (and sometimes it times out...)
  - too many choices: hard to locate links and forms
## Web Anytime, Anywhere

- **A lot of hype!**

- **Internet/WWW was not designed to be viewed/accessed by diverse devices with:**
  - limited processing power and memory
  - restricted power consumption
  - small screens, or no screens
  - different input/output devices (e.g., stylus, voice)

### Comparison of Devices

<table>
<thead>
<tr>
<th></th>
<th>Phone</th>
<th>Palm Pilot</th>
<th>Typical laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Screen size</strong></td>
<td>N/A</td>
<td>160x160 (6x6cm)</td>
<td>1024/768 (13.1”)</td>
</tr>
<tr>
<td><strong>Bandwidth</strong></td>
<td>N/A</td>
<td>9-19kbps (5-15kbps)</td>
<td>56kbps</td>
</tr>
</tbody>
</table>
| **Input/Output**     | Keypad and Voice/Voice | Grafitti/Subset of HTML | Keyboard/HTML,XML ...
| **Processor**        | N/A   | 16-20Mhz | 600Mhz |
| **Memory**           | N/A   | 8MB | 128MB |

The Web is just too complex
WebViews: The personal Web simplifier

- A system for creating simplified and personalized views of Web sites that can be accessed from various devices.

- Enable rapid deployment of personalized services that can be accessed from diverse terminals.
Features

- Requires no cooperation from Web sites
- Easy to create, easy to maintain: no programming required
  - beats re-engineering Web sites or creating specialized wrappers (e.g., everypath.com, oraclemobile.com)
- Wide coverage: access to virtually any Web site, and from many different devices
  - beats wireless/voice portals: e.g., Audiopoint, BeVocal, Quack, TellMe that provide access a limited number of sites (e.g., news, weather, driving directions)
- Personalized access
  - beats proxies that filter/reformat content and provide no personalization or customization e.g., ProxiWeb, PhoneBrowser
- Simplifies transcoding
Some examples of Web views

- Travelocity fares to Hong Kong
- Houses for sale in Summit, NJ
- BMWs for sale in NY/NJ/CT
- Account balance at Fidelity
- Webster’s thesaurus
- Weather in Murray Hill, NJ
- Lucent’s employee directory
- etc....
Creating a Web view of Travelocity

Create Web view
“Juliana’s lowest fares to Hong Kong”

WWW10 Hong Kong, May/2001
Juliana Freire
Accessing the Travelocity WebView from a PDA

**Flight list**

- **Price:** 1 adult @ USD 829.10
- **Flight:** Continental Airlines flight 99 on a Boeing 777 Jet
  - Departs: Friday, Apr 27
  - From: Newark, NJ (EWR) at 11:35am
  - To: Hong Kong, Hong Kong (HKG) at 3:15pm Saturday, Apr 28
  - Stops: None

- **Flight:** Continental Airlines flight 98 on a Boeing 777 Jet
  - Departs: Saturday, May 05
  - From: Hong Kong, Hong Kong (HKG) at 12:05pm
  - To: Newark, NJ (EWR) at 3:35pm
  - Stops: None

*Most of the data is transferred through a fat pipe*

*Only a subset of the final page needs to be transcoded*

“Juliana’s lowest fares to Hong Kong”

WWW10 Hong Kong, May/2001

Juliana Freire
Outline

- Creating Web views

- Important issues:
  - Making views robust
  - Transcoding views

- The WebViews service

- Related Work

- Conclusions and Future Work
**WebViews Recorder**

- Extends WebVCR (WWW9)
- Transparently tracks and records users’ browsing actions: generates **smart bookmarks** - shortcuts to Web pages that do not have a well-defined URL
- Adapt to thin-client scenario: create **parameterized** smart bookmarks
  - need descriptive name for parameters
  - invalid selections: need to save more information in order to support user input
- Limitation: deterministic navigation
  - need support for conditional navigation

![Form](image)

<table>
<thead>
<tr>
<th>Juliana’s lowest fares</th>
<th>Depart. day:</th>
<th>Depart. month:</th>
<th>Destination:</th>
<th>Return day:</th>
<th>Return month:</th>
<th>Departure airport:</th>
<th>Login:</th>
<th>Password:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>27</td>
<td>April</td>
<td></td>
<td>6</td>
<td>May</td>
<td>EWR</td>
<td>jfreire</td>
<td>******</td>
</tr>
</tbody>
</table>

Submit
Clipping Web Pages

- Specify the components of a Web page to be extracted
- Requirements: standards-based, powerful, portable
- Choice: XPath

  e.g., //html/body/center[2]/div/table[2]/tr/td/table
  [position()>=3 and position()<=8]

- Hard to specify manually - need to provide GUI support
  - automatically generate expressions (details in the paper)

- Drawbacks:
  - ill-formed pages: must “tidy” HTML pages before applying XPath
  - slow processors
Web view specification

Retrieve page

<WEB-VIEW id="juliana_clippings">
<BOOKMARK idref="juliana_travel"/>
<REFRESH-INTERVAL>24 Hours</REFRESH-INTERVAL>

<EXTRACT fragment_name="first_3_itineraries">
<![CDATA[
  (//table/tr/td[(contains(string(), 'Price:') or contains(string(), 'Option')) and not(descendant::table)/parent::tr/parent::table)
  [position() >= 1 and position() <= 6]
]]>
</EXTRACT>

<EXTRACT fragment_name="first_itinerary">
<![CDATA[
  (//table/tr/td[(contains(string(), 'Price:') or contains(string(), 'Option')) and not(descendant::table)/parent::tr/parent::table)
  [position() >= 1 and position() <= 2]
]]>
</EXTRACT>
</WEB-VIEW>

<BOOKMARK id="juliana_travel">
<URL>http://travelocity.com</URL>
<LINK>
<loc>document.links[8]</loc>
<href>
<![CDATA[http://dps1.travelocity.com/lognlogin.ctl?tr_module=AIRG&SEQ=1]]>
</href>
</LINK>
<form>
  <!-- Login form -->
  <!-- 9 Best Itineraries link -->
</form>
</BOOKMARK>

<WEB-VIEW id="juliana_clippings">
<BOOKMARK idref="juliana_travel"/>
<REFRESH-INTERVAL>24 Hours</REFRESH-INTERVAL>

<EXTRACT fragment_name="first_3_itineraries">
<![CDATA[
  (//table/tr/td[(contains(string(), 'Price:') or contains(string(), 'Option')) and not(descendant::table)/parent::tr/parent::table)
  [position() >= 1 and position() <= 6]
]]>
</EXTRACT>

<EXTRACT fragment_name="first_itinerary">
<![CDATA[
  (//table/tr/td[(contains(string(), 'Price:') or contains(string(), 'Option')) and not(descendant::table)/parent::tr/parent::table)
  [position() >= 1 and position() <= 2]
]]>
</EXTRACT>
</WEB-VIEW>
What happens when Web pages change?

Before April, 2001

After April, 2001

WWW10 Hong Kong, May/2001

Juliana Freire
Robustness

- Ensure that the intended content is retrieved even if underlying site changes
- Built-in heuristics for robust navigation
  - identify at each step the correct action
  - use fuzzy matching
  - need to be efficient - executed multiple times
- Hints to identify fragments of Web pages
  - e.g., extract tables that contain “Price” or “Option”, or
  - extract text delimited by string1 and string2
- Not full-proof: but when Web views break, they are easy to fix
Transcoding WebViews

- Loosely vs tightly-coupled
- Palm and HTML-friendly devices: use existing proxies or no proxy
  - quality is reasonable for simple and small clippings
- Telephone (voice and touch-tone input/voice output): built our own transcoder
  - transcoding into VoiceXML is challenging...
  - tighter coupling with transcoder is advantageous
Voice enabling Web views

- Voice interfaces are very different from the usual visual (HTML) interfaces: intrinsically serial
- Transcoding an arbitrary HTML page into VoiceXML is unlikely to result in a reasonable user experience
- Voice views:
  - focused: simpler to transcode
  - can be annotated to generate better quality transcoding during access, e.g., how to read tables (row-wise vs column-wise), what is the header, which columns/rows to project
  - extra information saved is useful for transcoding into VoiceXML, e.g., user choices can be constrained for better recognition
The WebViews Service

- Web service accessed via HTTP requests
  
  http://webviews.bell-labs.com/cgi-bin?user=juliana&view=travel

- Parameters
  - e.g., departure day and month

- Device specific extraction
  - e.g., 1st itinerary if WAP, all itineraries if Palm

- Modes:
  - synchronous vs asynchronous (push/pull)
  - periodic updates (caching)
  - notification
Related Work

- Wrappers in information integration systems:
  - query Web sites as if they were databases, e.g., Information Manifold (VLDB'96), Web Integrator (SIGMOD'99)
  - extract structure from semi-structured data, e.g., NoDoSe (SIGMOD'99)
  - simpler extraction: less semantics, more robust

- Robust wrappers
  - WebVCR (WWW9), Phelps and Wilensky (WWW9), Davulcu et al (PODS 2000)

- Personalization systems and portals
  - e.g., MindIt, Yodlee, Octopus, ezlogin
  - similar but not very robust, and no support for “complex” navigation

- Wireless and Voice application service providers
  - e.g., OracleMobile, tellme, heyanita
  - cooperate with content providers

- Transcoding proxies
  - no personalization
Conclusions and Future Work

- **WebViews architecture:**
  - simplifies the creation of robust views of Web content and services
  - views can be tailored for specific devices
- **Explore different application scenarios**
  - expert users
  - cooperative Web sites
- **More “programmability”** (e.g., conditions, iterations)
- **Adopt CC/PP (?)**
- **Scalability/Security**
Different Application Scenarios

- **Tool for end-users**
  - allows wireless and voice access to far more Web content than other approaches

- **For Web content-providers, ISPs, ASPs**
  - does not require cooperation with Web sites
  - easy to setup and maintain (hence *cheaper*)

- **For corporations (Intranet)**
  - does not require tight cooperation with Web site designers
  - easy to setup and maintain (hence *cheaper*)
Summary: The WebView Service

- Allows end-users/content providers to easily create and maintain personalized Web views
  - A Web view is a set of instructions to retrieve a particular page and a specification of which components of the page should be extracted
- Uses existing HTML pages
- Based on standard technology: HTTP, HTML, XML, etc
- Web views give users single-click access to information of their interest, e.g., CNN headlines, weather information etc.
  - Web views can be parameterized, allowing an easy creation of simple Web queries/services
  - Different views can be created that are suitable for different types of terminals
- Device independent: allows access from various devices (PDA, mobile phone)
  - “clipped” information is transcoded into the desired format (WML / VoiceXML) before being shipped to the user via the corresponding gateway (WAP proxy / Voice gateway)
VoiceViews: Usage Scenario

1. User creates web view on desktop and uploads it to the WebTalk server.

2. User dials 1-800-555-0000

3. Customer is identified via ANI (and/or via additional input).

4. WebTalk server is queried for list of customer’s views, and the resulting list is sent back to the customer.

5. WebTalk server retrieves list of customer views.

6. Customer chooses desired view: “Press 1 for 401(k) balance, 2 for breaking news from Times of India, …”

7. Customer request is forwarded to WebTalk server, and the resulting VoiceXML script is sent back to the customer.

8. WebTalk server replays specified view and returns the extracted content in VoiceXML.

9. Customer hears: “…”