Collaborative Transcoding of Web Content for Wireless Browsers

by

DAVID MICHAEL SIRKIN

S.M. (Management) 1992, Massachusetts Institute of Technology

B.A.S. (C.S.E.) 1985, University of Pennsylvania

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September, 2000

Author [illegible]
Department of Electrical Engineering and Computer Science
September 1, 2000

Certified by [illegible]
Philip Greenspun
Research Scientist, Laboratory for Computer Science
Thesis Supervisor

Accepted by [illegible]
Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

Collaborative Transcoding of Web Content for Wireless Browsers

by

DAVID MICHAEL SIRKIN

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

Mobile device browsers require their own transport protocols, scripting and markup languages, creating obstacles to interoperability with current Web services. The current approaches to the problem have (1) service providers creating mobile versions of their services, or (2) gateways automatically transcoding markup. But not all developers will migrate their services, and automated transcoders can fail to return relevant content where HTML is not well-formed or pages are especially long. When transcoding is successful, small mobile displays often require that users search through dozens of screens to find desired information.

To address these shortcomings, a collaborative service was developed to guide users in selecting elements from Web content to serve as bookmarks to their mobile browsers. Specific research questions were whether and when bookmarks were better suited mobile users' information needs and would be used in preference to the alternatives. A user study was untertaken to compare gateway and collaborative approaches, considering setup effort, access effort, and page relevance. As the time to select elements decreases, subjects were more inclined to use bookmarks rather than an automated transcoder. Most indicated that the ability to access specific information quickly when needed outweighed the additional setup effort. This became more important as page length increased. The service was launched on the public Internet as part of the study. User feedback led to the addition of a well-received feature to bookmark native mobile sites.

Thesis Supervisor: Philip Greenspun
Title: Research Scientist, Laboratory for Computer Science

Acknowledgements

Particular thanks are owed to Andrew Grumet for contributing the idea for the bookmark service, and to Philip Greenspun for encouraging the usability study with real-world users. Their expert insight and advice over the duration of the project have made the service possible and the writeup sound. Thanks also to the user study subjects and public Internet users worldwide for their time and interest and responses to a lot of questions.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Mobile and Web Applications

Designing applications to interact with mobile and wired Internet users is one of the great challenges facing Web publishers and service developers today. Compared to the wired Internet, mobile networks are expensive, slow, and unstable [01]. Compared to desktop computers, mobile devices have limited and non-standard keypads, displays, processing power, and memory capacity [02]. Until 1997, the few wireless communication companies connecting to the Internet had to develop proprietary browsers, protocols, markup, and scripting languages to operate reliably within these constraints. Phone.com championed the Handheld Device Markup Language (HDML) operating through its UP.Browser, and Palm Computing developed Web Clippings accessed through Palm Query Applications (PQAs).

But between 1998 and 2000, service providers and device manufacturers have been working with the World Wide Web Consortium to establish industry-wide standards. Founded by Nokia, Ericsson, Motorola, and Phone.com, the WAP Forum recently introduced the Wireless Application Protocol (WAP). WAP defines a set of protocols in transport, session, and application layers similar in concept to the wired Internet but optimized for mobile networks and devices [03].

The Wireless Markup Language (WML), an Extensible Markup Language (XML) application with a public Document Type Definition (DTD) [04], is designed for use over narrow-band networks by devices with limited input and computational facilities. WML content is organized using card and deck metaphors. A card represents any single user interaction, such as a screen of text, a choice menu, or a text-entry field. Users navigate through a series of cards, review the contents of each, make a choice or enter any requested information, then move on to the next card.

Cards are grouped locally into decks. A deck is similar to a Web page in that it is the file transmitted to a client and is identified by its Uniform Resource Locator (URL).

Servers on the wired Internet reach clients on mobile networks by connecting through WAP Gateways that reside between the networks. A server passes WML or WMLScript documents to a gateway, that (1) converts from HTTP to the WAP protocol stack, (2) compiles the documents into the bytecode mobile browsers can read, then (3) forwards that code on to a client [05]. The process is then run in reverse for mobile clients to reach Internet servers. Figures 1 and 2 compare the paths for information flow for wired and mobile networks.

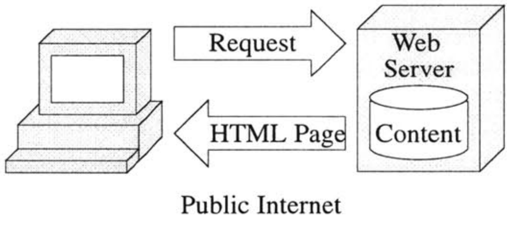


Figure 1: A Web server and client communicate using HTML and Internet Protocols. A Server connects directly to a desktop computer over the wired public Internet. Alternatively, it may connect through a firewall or proxy that resides between the public Internet and private wired intranet. The server and desktop computer rely on the same data format for requests and responses.

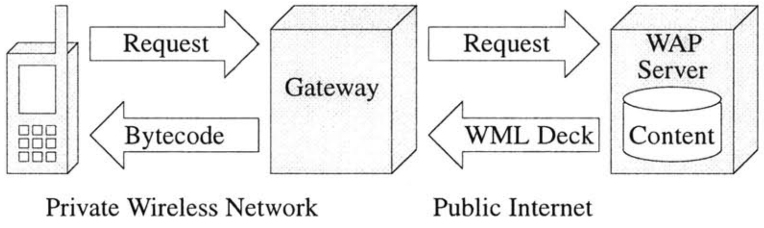


Figure 2: A WAP server and client communicate using WML and Wireless Application Protocols. A Server connects directly to a WAP gateway that resides between the wired public Internet and private wireless network. The gateway transforms requests and responses between the textual WML stored on the server and the compiled WML received by the device.

Unfortunately, since mobile browsers are designed to understand only wireless standards, they are unable to access Web content directly. Increases in wireless bandwidth may eventually lead both network architectures to converge [06], but the information appropriate for mobile use will still differ from that for desktop use. Differences in input and display are part of the reason. But more importantly, mobile users are more likely to require task-specific information for immediate use than to browse around for something that might be relevant.

1.2 Mobile and Web Interoperability

To serve both mobile and desktop users, application providers have three options:

- build versions of their services to serve mobile browsers directly;
- build versions of their services to generate a device-independent language, which is then translated into the native formats of requesting browsers;
- serve existing content as is and rely on downstream processing to transcode Web to wireless markup.

The first approach allows developers to take advantage of unique wireless network capabilities such as telephony interface and geographic positioning. Unfortunately, it also requires them to create, debug, and maintain duplicate sets of functionally similar code. Changes in data manipulation or presentation may require similar changes in both sets, depending on the modularity of the service.

Unless their services are already designed for device-independence, the second approach requires developers to modify their current services to generate XML, then apply Extensible Stylesheet Language (XSL) stylesheets to translate that to the appropriate markup. While the approach is the WAP Forum's recommended long-term solution, few publishers will adopt it until they have a sufficient mobile user base. As of July 2000, less than one percent of Web services have been modified to support mobile access [07].

For the third approach, WAP gateways are a natural location to transcode, as they already transform the textual WML stored on servers to the compiled WML received by devices [08]. Automated transcoding has the advantage of requiring little if any additional development effort by application providers, or access effort by users. But in practice it requires the original content to be well-formed, which is often not the case [09].

Nevertheless, companies such as Spyglass [10], Puma Technology [11], Digital Paths [12] and Avant Go [13] have already released products to implement this solution. Their code resides on a WAP gateway and acts as a proxy to convert HTML automatically to wireless device languages like HDML or WML. But not all Web content is well-suited to mobile viewing, even after successfully translating markup and accounting for scripts and images. A server cannot know what information on a particular page any mobile user wants, so it tries to deliver it all. Owing to memory constraints, WAP devices have compiled deck limits of approximately two to eight kilobytes [05]. As a result, most Web pages will be divided among multiple cards spread across several decks, and served separately. More than 50 cards per deck is not uncommon. Users then must try to find desired information buried in a deck. Sometimes that information is clipped, because the translated page exceeded the deck limit.

Oracle [14] and IBM [15] offer suites of middleware to perform the same task but in a different way. Adapters extract content from Web pages to translate to an intermediate device-independent format like XML. Transformers then translate the XML to any of several mobile formats. The intermediate format allows the use of one adapter per source and one transformer per device. But these suites are designed for application providers to select content from already-existing Web pages that they will make available to all mobile users. So the same problem occurs as with gateway transcoders, that mobile users with different information needs have to receive the entire (translated) Web page content.

Gateway and middleware transcoding may address the Web-to-mobile transmission problem, but they do not address the Web-to-mobile information problem. Mobile users know their own information needs better than automated or predefined transcoders. The questions addressed in this thesis are (1) whether we can design a service to lever this knowledge in advance in a way that makes the subsequent mobile experience more personal and precise, and (2) which users accessing what type of content benefit most from its use, when compared with the other approaches.

2 A WAP Bookmark Service

2.1 Goals

The new service should allow users to view Web page content they cannot access directly from their mobile browsers. The mobile view of that content should be more compact and relevant than automated transcoders can provide. So control over which elements are selected from the content should reside with the users. That is, users with different information needs should be able to select different elements from the same page. We do not want to save page content directly, only the means to retrieve certain elements at a later time, mapped to the right user and returned using a language mobile browsers understand. Since URLs below the top-level at many sites can be quite long, addresses should be entered using a desktop Web browser rather than a mobile device keypad.

2.2 Design

Users first create bookmarks using their desktop Web browser. Each bookmark pairs a target Web page address with a regular expression. The expression is used to match single or multiple occurrences of a text pattern within the page. Matching a single occurrence would be used on a page that contained, for example, a stock quote or weather forecast; matching multiple occurrences would be used on a page that contained a number of news headlines. Users then access these bookmarks as a select list using their mobile device browser.

Bookmarks are created by entering an address and composing a regular expression, or by copying an already existing bookmark. A larger store of bookmarks makes it more likely that one will already exist to suit a particular user's needs. Sharing bookmarks though a public exchange lets users make the most of that resource. So users can view their personal menus by requesting bookmarks that only they own; or they can view public summaries by requesting all public bookmarks or only those under one host or owned by another user. But even if no public bookmark exactly matches a user's needs, another may be found that is related in some way. For example, it may apply the desired regular expression but to a different page. In that case, the original address can be edited and the new bookmark saved back into that user's personal menu. Allowing bookmark authors to note details about a bookmark's components, operation, or use may further encourage collaborative learning about how to edit existing bookmarks or compose new regular expressions.

WAP device design encourages the use of remote resources. To conserve onboard memory, many mobile browsers do not store bookmarks (or home pages or cookies). Instead, they are stored on the service provider's local WAP server [16], which identifies the device uniquely and supplies the appropriate files during requests. But when users roam on another network or switch providers, they can lose these files [17]. Transcoders that reside on WAP gateways only serve users on the home network, so roaming users can lose the ability to access the Web. The bookmark service solves this problem by having users store their bookmarks on a server at MIT's Laboratory for Computer Science, which they can access from any network or through any gateway. Persistent storage on the server is achieved using a Relational Database Management System (RDBMS). Figures 3 and 4 compare the paths for information flow for automated gateway transcoders and the bookmark service.

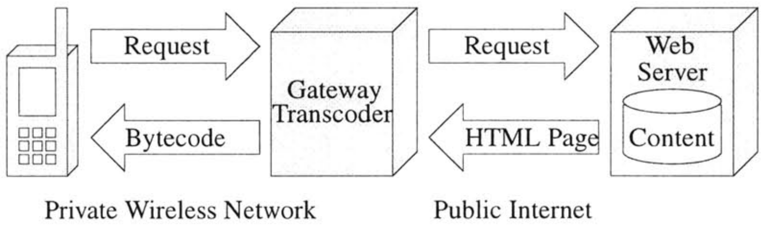


Figure 3: The automated transcoder resides on a WAP gateway. Requests from a client to the gateway are received, decompiled and forwarded on to a Web server as in Figure 2. Responses from the Web server to the gateway are first translated from HTML to WML by the transcoder. Then the gateway compiles and forwards the result to the client.

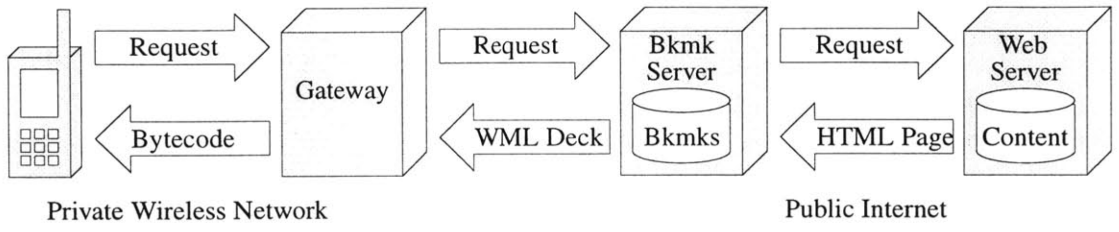


Figure 4: The bookmark service resides between the WAP gateway and content server. Requests from a client are processed by a gateway as before, but specify the bookmark to serve rather the page to return. The bookmark server issues page requests to a Web server. Responses are matched against the bookmark's regular expression, capturing selected HTML elements. Then the server forwards the result to the gateway as WML as in Figure 2.

2.2 Details

Development Environment

The bookmark service is composed of scripts that generate Web pages or WAP decks, and process user input and database information. It uses the AOLServer Tcl 8.3 API connected to an Oracle 8i RDBMS, and is built as an extension to the open-source ArsDigita Community System (ACS) toolkit. The ACS is a Web software system for building and managing online communities and collaboratively contributed content [18]. The service relies on existing ACS modules that handle user management functions such as registration, group membership and permissions, and site management tools such as request processing and server monitoring.

Security

The service allows users to upload and evaluate custom-defined regular expressions on its server. Doing so requires the use of the built-in Tcl procedure `eval`, which presents a potential security risk. In particular, an `exec` procedure included within the expression may be able to invoke arbitrary Unix programs. We can address the problem by taking two precautions. The first is to run the Web server as an unprivileged user in a `chroot()` environment. Then scripts running on the server will only be able to view files or execute the limited selection of programs under the server's root directory. But this does not prevent users from including `exec` commands. So rather than have users upload an entire regular expression while creating bookmarks, the second precaution only allows them to compose and submit the match pattern. A script then builds the full expression evaluating the user-supplied input as a string only. Steps that do so include (1) wrapping the match pattern in braces to force its literal interpretation, and (2) escaping brackets that would result in a Tcl procedure call.

Schema Mapping

Bookmarks are represented by rows in the `wap_bookmarks` table. Since different bookmarks may refer to the same page but apply different regular expressions (and the reverse), neither of these represents a unique identity. Each bookmark uses an integer derived from an Oracle sequence as a primary key. Here is the table definition:

```
create table wap_bookmarks (
     --                    Identify this particular bookmark.
     bookmark_id           integer primary key,
     bookmark_title        varchar(500),
     owner_id              integer not null references users(user_id),
     --
     --                    Want to view public bookmarks by host and owner.
     complete_url          varchar(1000) not null,
     host_url              varchar(500) not null,
     --
     --                    Where the_regexp is null, bookmark is a WAP deck.
     the_regexp            varchar(4000),
     --
     --                    Space-separated list of match vars to return.
     match_vars            varchar(1000),
     --
     --                    Match a single or multiple occurrences on page.
     multiple_p            char(1) default 'f' check(multiple_p in ('t','f')),
     --
     --                    For querying and presenting bookmarks in menu.
     sort_key              integer not null,
     --
     --                    Bookmarks can be hidden from view on mobile device.
     hidden_p              char(1) default 'f' check(hidden_p in ('t','f')),
     private_p             char(1) default 'f' check(private_p in ('t','f')),
     --
     --                    Additional owner-supplied text for generated page.
     preceding_text        varchar(1000),
     following_text        varchar(1000),
     --
     --                    User description, also for searching bookmarks.
     one_line_descrip      varchar(1000),
     detailed_descrip      varchar(4000),
     --
     --                    Meta-tag description, for searching bookmarks.
     meta_keywords         varchar(4000),
     meta_descrip          varchar(4000),
     --
     --                    Track when bookmark was created and last modified.
```

```
    creation_date        date not null default sysdate,
    modification_date    date,
    --                   --
    --                   Track when page was last alive and regexp worked.
    last_check_date      date,
    last_live_date       date,
    last_regexp_date     date
);
```

The first few attributes are those required to serve bookmarks: (1) the target Web page address, (2) the regular expression to apply, (3) which match variables to return, and (4) whether to match one or all occurrences in the page. The remaining attributes are used to view and manage bookmarks. The owner's identity refers to a unique key in the ACS user information table. The sort key, hidden and private flags, and preceding and following text permit custom menu view settings. One-line, detailed, and meta-tag descriptions allow users to better understand public bookmarks and find them through keyword searches. And dates are stored for the last time bookmarks were checked, hosts were reached, and regular expressions matched.

Because users can copy each other's bookmarks, there will be duplicate information in the table. An alternative representation would avoid this by using two tables: one to store the attributes required for serving a bookmark, and another to map users and their view and management settings to the first. When users copy a bookmark, they just reference a row in the attributes table. But the advantage of the approach is also its drawback. When a page's layout changes, only the original author has to update the attributes row, rather than all users who reference it. However, all bookmarks that reference that row are subject to the changes made by the original author.

Transactions

Creating a bookmark inserts a new row in the database. Copying a public bookmark into a personal menu also inserts a row in the database, having the same attributes as the original except for the primary key and owner. Users should be able to modify or delete bookmarks that reside in their own menu, but not in someone else's. So all updates must be constrained by first confirming the bookmark's owner.

2.3 Functions

In keeping with the design goal of streamlining the mobile experience, selecting page elements takes place beforehand using a desktop Web browser. Retrieving the elements then only requires following a link on the mobile display. Visitors to the site are directed to Web or WAP functions based on their connection's request headers. These lines are included with most HTTP requests to provide the server with information about the client. The bookmark service first checks the user agent and accept fields for the client's browser type and accepted document types, then makes the redirect decision. Figure 5 shows the page flow for the Web and WAP site functions.

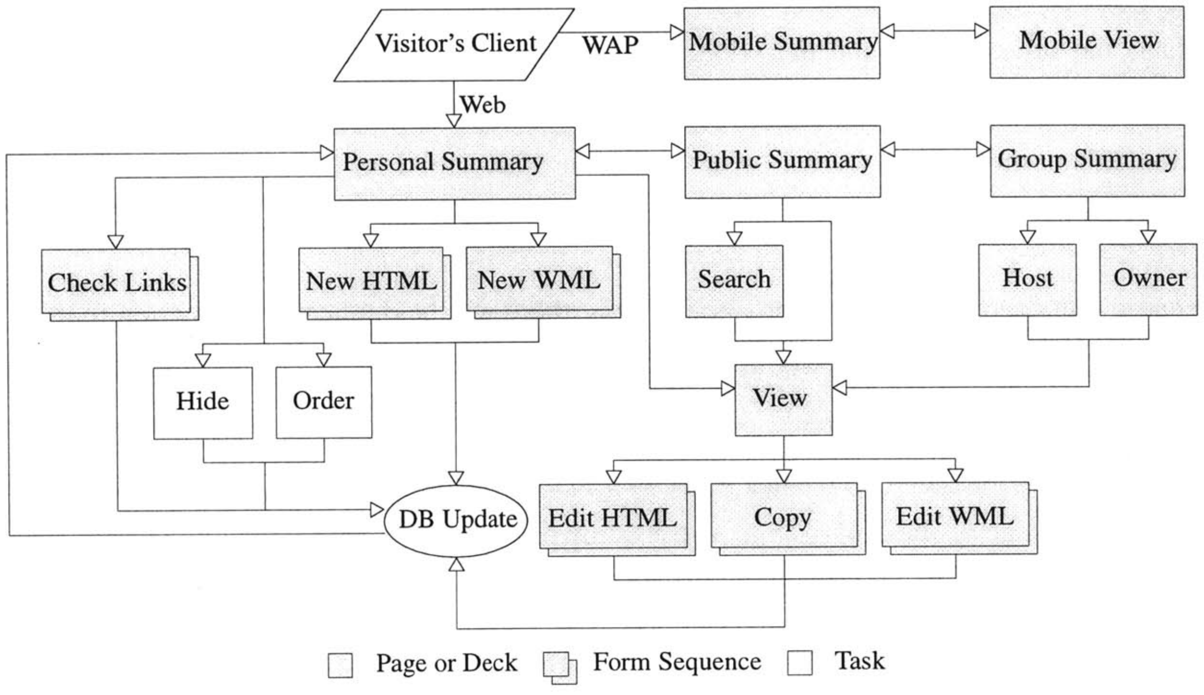
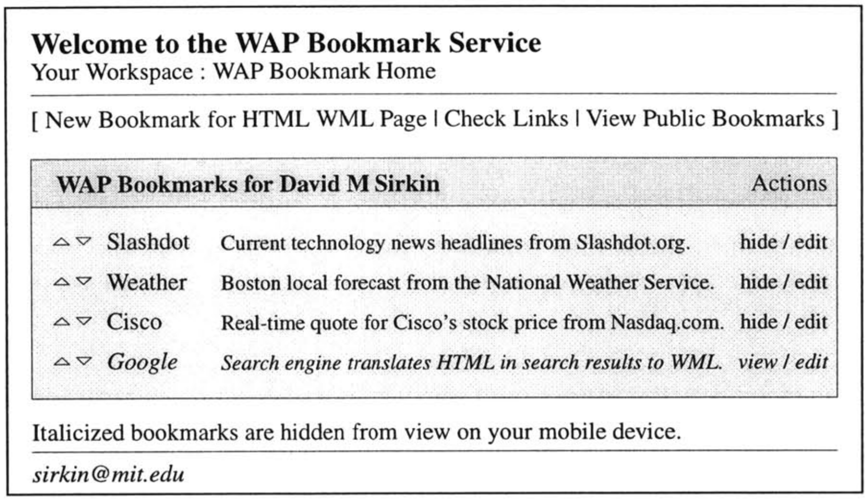


Figure 5: Page flow for bookmark service users. Users are directed to the appropriate area of the site for their client type. Web users create new bookmarks, view or edit their existing bookmarks, check the reference status of their existing bookmarks, or view and copy other users' bookmarks. They can also set the order of bookmarks in the list or hide them from the mobile view. Database inserts or updates are redirected back to the personal summary. WAP users view their current bookmarks in the designated order and apply individual bookmarks.

Once at the appropriate part of the site, users first see their personal bookmark summary. Figures 6 and 7 show this summary view using Web and WAP browsers.



Welcome to the WAP Bookmark Service
Your Workspace : WAP Bookmark Home

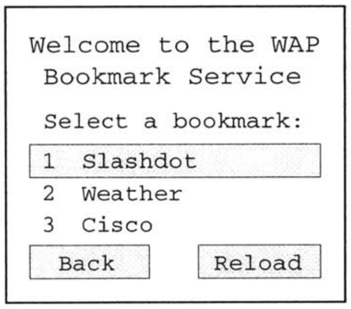
[New Bookmark for HTML WML Page | Check Links | View Public Bookmarks]

WAP Bookmarks for David M Sirkin		Actions
△▽ Slashdot	Current technology news headlines from Slashdot.org.	hide / edit
△▽ Weather	Boston local forecast from the National Weather Service.	hide / edit
△▽ Cisco	Real-time quote for Cisco's stock price from Nasdaq.com.	hide / edit
△▽ *Google*	*Search engine translates HTML in search results to WML.*	*view / edit*

Italicized bookmarks are hidden from view on your mobile device.

sirkin@mit.edu

Figure 6: Web home page view of bookmark service. Users select bookmarks from the list, change the order they appear or hide them from their mobile view. Users follow links from here to Web forms to create new bookmarks for Web or native WAP pages, edit existing bookmarks, check the status of bookmark references and expressions, and view summaries of public bookmarks.



```
Welcome to the WAP
 Bookmark Service
 Select a bookmark:
 1  Slashdot
 2  Weather
 3  Cisco
  Back        Reload
```

Figure 7: WAP home deck view of bookmark service. Users select bookmarks from the list, view the results on the following card, then update the view of that card or return home to select another bookmark. The Slashdot bookmark is highlighted to be selected by pressing a soft key or entering the bookmark number. The Google bookmark is absent, as it was indicated as hidden from the mobile view in Figure 6.

Create and Manage Bookmarks

Most Web pages that users want to bookmark include periodically updated content. These pages are usually generated by programs that insert elements of timely information into otherwise unchanging page markup. These elements are what bookmark service users need to view when mobile. Regular expressions indicate which elements to capture from a page by anchoring them between unchanging page markup. But while regular expressions are powerful commands, we do not want to limit the potential user base to programmers with experience composing them.

The service therefore provides two interfaces for creating bookmarks: advanced and friendly. The advanced interface has users compose components of the regular expression, while the friendly interface has users indicate the type of content to match. Both interfaces follow the same basic process: (1) select a target Web page and fetch its contents, (2) build a regular expression using user-provided input for a match pattern, (3) apply that expression to the page and display the results, and (4) iterate until the results match the user's intent. Where the two interfaces differ is what input they require and how they use that input to build the expression.

The advanced interface takes the more direct approach by having users compose the match pattern into a Web form. Patterns of this sort are most often built by selecting out a section of page markup that includes the desired text string. Replacing target elements within the string with meta-character sub-patterns such as `([^<]+)` or `(.+?)` instructs the regular expression to capture whatever matches those locations in the string. The first sub-pattern matches any text from the preceding character to the first < encountered, which often indicates the next markup tag. It is the better choice when the target elements are surrounded by markup. The second sub-pattern matches any text from the preceding character to the following character. It is the better choice when the target elements include markup. The workflow help text provided for the advanced interface is as follows:

```
The advanced interface works on most pages, since you compose the
regular expression yourself. You can only have one match pattern per
bookmark, but that pattern can include many sub-patterns. You can then
select which of the sub-patterns to view on your mobile browser.

1. View the source and find the text to select out

     Leave spaces and line feeds as they appear in the markup
     You may have to scroll to the side as well as up or down

2. Copy a block of markup around the text and paste it into the form

     Select a large enough block to identify the target text
     But smaller blocks may be required for multiple matches

3. Replace the targeted text with sub-patterns ([^<]+) or (.+?)

     Use the first if the text is followed by a markup tag
     Use the second if the text itself includes markup tags

4. Indicate whether to match single or multiple occurrences

5. View the results and iterate until you are satisfied
```

Say for example we are camera shopping and want a bookmark to return the recent classified ads on photo.net. The first step is to set the target page address to http://www.photo.net/gc/domain-top?domain_id=2 and look at the source. There we find that all of the ads have the following form, where `number` and `item and price` are different for each entry:

```
<li><a href="view-one.tcl?classified_ad_id=number">
item and price
</a>
```

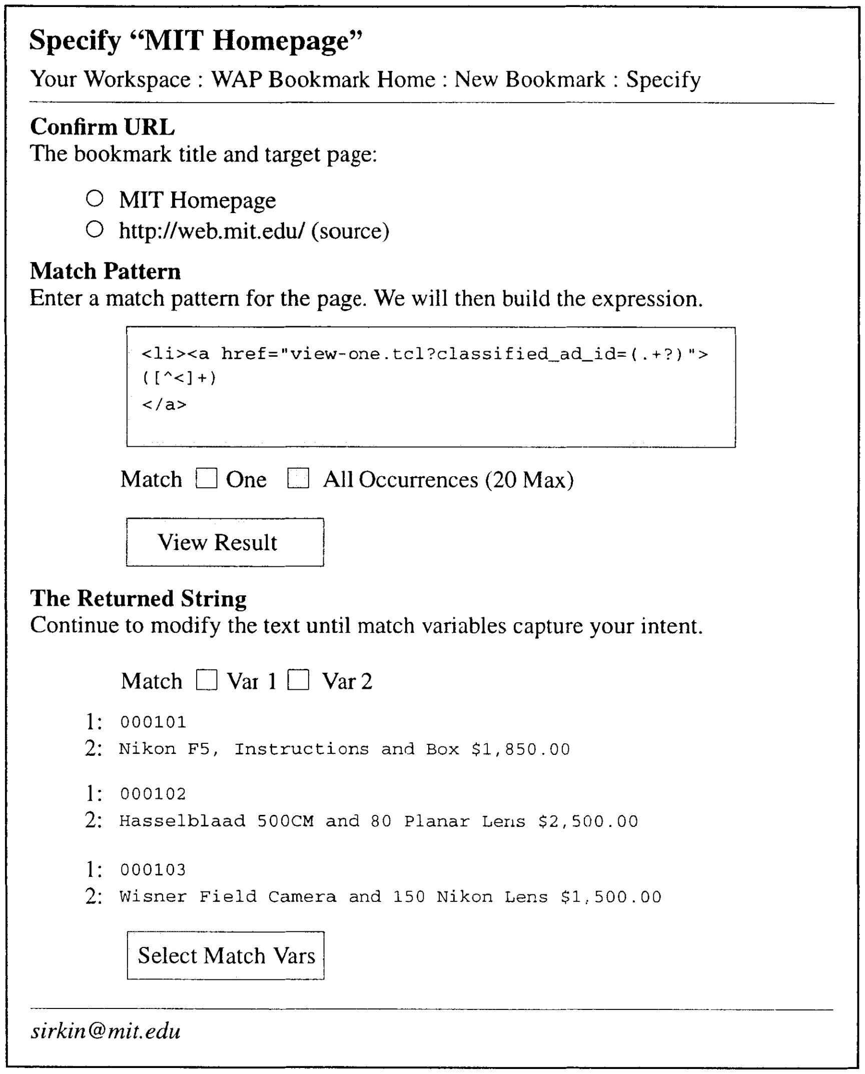
We would paste the string into the Web form and replace the `number` and `item and price` elements with sub-patterns. One possible match pattern would be:

```
<li><a href="view-one.tcl?classified_ad_id=(.+?)">
([^<]+)
</a>
```

The line feeds are considered part of the match pattern, and are used to differentiate the recent ads from other unordered list items in the page having line feeds at different places. The script then takes over by counting the parenthesis-enclosed meta-characters within the user-submitted pattern as the number of match variables. The regular expression is assembled by enclosing the pattern within braces, and appending the variable containing the page content followed by the match variables. In this case, the resulting regular expression would be:

```
regexp {<li><a href="view-one.tcl?classified_ad_id=(.+?)">
([^<]+)
</a>} $url_text match var_1 var_2
```

The expression is then evaluated and the results are displayed back to the same Web form. If the user has opted to match all occurrences within the page, the first match is removed and the expression is re-evaluated on the revised page. These two steps repeat until no more matches are found, when as before the results are displayed to the same Web form. In this way, users receive immediate feedback on how their bookmarks work, and the pattern can be continually revised. Users then select which of the returned match variables to place into the mobile view and which to ignore. For the classified ads bookmark, the `number` element would probably be ignored. Then after soliciting any other text to appear in the bookmark, and any brief or detailed descriptions, the script inserts the new bookmark into the database. Figure 8 shows the advanced interface composition view.



Specify "MIT Homepage"

Your Workspace : WAP Bookmark Home : New Bookmark : Specify

Confirm URL

The bookmark title and target page:

- ○ MIT Homepage
- ○ http://web.mit.edu/ (source)

Match Pattern

Enter a match pattern for the page. We will then build the expression.

```
<li><a href="view-one.tcl?classified_ad_id=(.+?)">
([^<]+)
</a>
```

Match ☐ One ☐ All Occurrences (20 Max)

The Returned String

Continue to modify the text until match variables capture your intent.

Match ☐ Var 1 ☐ Var 2

```
1: 000101
2: Nikon F5, Instructions and Box $1,850.00

1: 000102
2: Hasselblaad 500CM and 80 Planar Lens $2,500.00

1: 000103
2: Wisner Field Camera and 150 Nikon Lens $1,500.00
```

sirkin@mit.edu

Figure 8: Advanced interface composition view. Users enter a match pattern into the form and select how many occurrences of that pattern to match against the original page. If the pattern includes more than one parenthesis-enclosed sub-pattern, the script counts them and applies each as a match variable. The form then allows the user to select which of the match variables to place into the mobile display and which to ignore.

The friendly interface generates the match pattern rather than having users compose it directly. Here users enter a text string that they wish to match against the page into the Web form. Strings of this sort are typically just copied from the browser window and pasted into the form. The workflow help text provided for the friendly interface is as follows:

```
The friendly interface works best on simple pages where the text you
want to select does not include hidden characters like &nbsp;. You can
only have one match pattern per bookmark, but you can choose to match
all occurrences of that pattern. You can then have several bookmarks,
where each matches different elements, for any page.

1. View the page and find the text to select out

2. Copy the text from the page and paste it into the form

3. Indicate the selectivity to apply to building the expression

     The most selective option will return at most one match
     Less selective options result in smaller match patterns

4. Indicate the sub-pattern to apply to building the expression

     Tag works most often; it selects text to the next markup tag
     Text works when the selected text itself includes markup tags

5. Indicate whether to match single or multiple occurrences

6. View the results and iterate until you are satisfied
```

For the classified ads bookmark, one possible string to match would be the first ad appearing on the page, shown as the first match in Figure 8:

```
Nikon F5, Instructions and Box $1,850.00
```

The script applies the user-provided string in a manner similar to how a user working with the advanced interface would. It finds the string's location in the page, selects out a section of page markup surrounding it, and replaces the original string in that selection with the meta-character sub-patterns to capture the match. Users can select which of the two default sub-patterns to use. The script then continues with the remaining steps as it would with the advanced interface.

The first problem with building the match pattern this way is that we cannot always locate the user-provided string. Text as it appears on a browser window is not formatted exactly as in its markup. One apparent space may actually be several spaces or a line feed or a tag. The string entered into the form will not include any tags and may have line feeds in different places, depending on the width of the user's browser window. There are two ways to resolve the problem. The first method is to just ask the user to select the string from the source window rather than the browser window. But the idea behind the friendly interface is to help the user avoid searching through the markup. So the second method attempts to account for space, line feed and tag discrepancies. It replaces every space or line feed in the string with meta-characters that represent: "find a space or line feed here," so the string to match would become:

```
Nikon[ \n]+F5,[ \n]+Instructions[ \n]+and[ \n]+Box[ \n]+$1,850.00
```

Then as the string is matched against the page to find its location, the earlier discrepancies become matches. In cases where tags are contained within the string itself, users can select the second meta-character sub-pattern, which will select all characters including the tags up to the next part of the match pattern. Any captured markup is then stripped from the result when returned.

The second problem is that we have to determine how much markup to select around the string. If the selection is too small, the expression may match too much of the page. Surrounding markup consisting of say, only `<li>` and `</li>` tags, would match all unordered list items in the page. While in some cases this will be what the user intends, but in others like the classified ads page it will not. Alternatively, if the selection is too large, the expression will match too little of the page. That is, it will only match that one location rather than all occurrences that would be similar if the pattern were less selective. So the script makes an initial guess at how much markup to select and iterates matching the pattern against the page, doubling the size of the guess each time, until only one match is found. Users can indicate on the form how selective the final match pattern should be. The most selective option represents the pattern that matched the page just once. Each successively less selective option represents a pattern that includes half the surrounding markup of the preceding, more selective option. The current form includes three selectivity options, so there are four times more characters in the most selective option than in the least. Since the smallest HTML tag is three characters, a reasonable initial guess is four times that, or twelve characters.

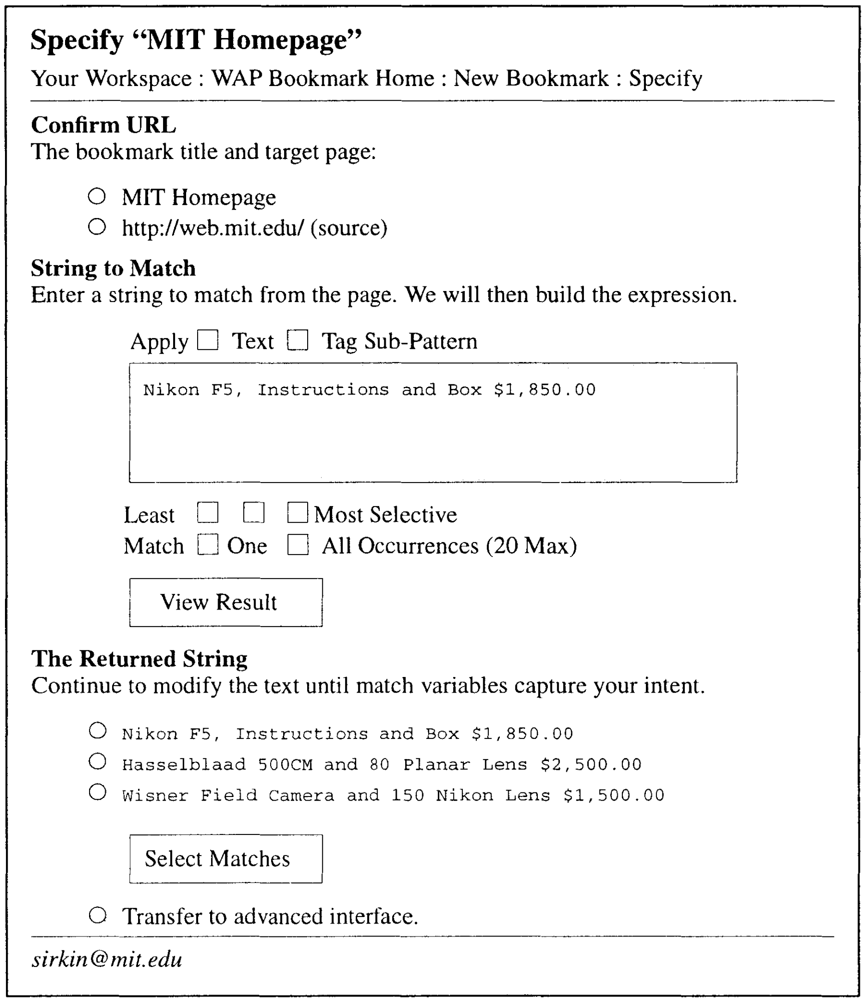
For the classified ads bookmark, the initial guess is too selective. The match pattern that results from selecting twelve characters around the string contains the `number` element, which is unique for each item:

```
id=number">
([^<]+)
</a>
```

As a result, the expression will only match that one occurrence in the page, where we were trying to match all similar occurrences. But the user can opt for a less selective match pattern. The least selective option only selects three characters around the string (counting line feeds), so the match pattern that results is exactly the right size to match all occurrences:

```
">
([^<]+)
</
```

Figure 9 shows the friendly interface composition view.



Specify "MIT Homepage"

Your Workspace : WAP Bookmark Home : New Bookmark : Specify

Confirm URL
The bookmark title and target page:

- ○ MIT Homepage
- ○ http://web.mit.edu/ (source)

String to Match
Enter a string to match from the page. We will then build the expression.

Apply ☐ Text ☐ Tag Sub-Pattern

```
Nikon F5, Instructions and Box $1,850.00
```

Least ☐ ☐ ☐ Most Selective
Match ☐ One ☐ All Occurrences (20 Max)

View Result

The Returned String
Continue to modify the text until match variables capture your intent.

```
○ Nikon F5, Instructions and Box $1,850.00
○ Hasselblaad 500CM and 80 Planar Lens $2,500.00
○ Wisner Field Camera and 150 Nikon Lens $1,500.00
```

Select Matches

○ Transfer to advanced interface.

sirkin@mit.edu

Figure 9: Friendly interface composition view. Users enter a string of text from the original page into the form and select (1) which meta-character sub-pattern to apply, (2) how selective the script should be in building the match pattern, and (3) how many occurrences of that pattern to match against the page. If the form cannot return the user's desired elements, she may transfer the current script-built match pattern to the advanced interface for further editing.

Having users enter a single string from the page rather than the full match pattern, regular expressions will only have one match variable. As a result, they cannot use the friendly interface to capture several different types of elements from more complex pages. But in those cases it does provide an initial match pattern that can be revised using the advanced interface.

Users can edit any part of a bookmark as well. But without the original string to match used in the friendly interface to compose the regular expression, a friendly interface to edit the expression makes little sense. The user could just as easily create a new bookmark from scratch. So only the advanced interface is provided to modify the match pattern of existing bookmarks.

Since Web page locations and layouts change periodically, references may become stale and regular expressions may no longer match. The service allows users to check their bookmarks, view target pages for what may have changed, and offers them the option to edit or delete those bookmarks that no longer function as originally intended. Appendix A includes the code for viewing and customizing a personal menu, creating and editing bookmarks, and checking bookmark status.

Share and Copy Public Bookmarks

Several summary views enable users to share and copy public bookmarks by helping them find bookmarks that suit their needs. Views can be ordered by recency or popularity, or grouped by host or owner. Ordered views bring new or frequently copied bookmarks to the top of the list. But other bookmarks may suffer from a positive feedback network externality where the same few are copied frequently, making them even more recent and popular, so they remain at the top of the list. Alternatively, grouped views allow users to find bookmarks based on a common Web page address or owner. Users can find all the different bookmarks for one page in the same place. If users know or find someone else with common interests, all of that user's bookmarks will also be in the same place. For users looking for particular bookmarks, there is a form to search the database for keywords appearing in bookmark titles, owner names, and owner-provided or meta-tag descriptions. Appendix B includes the code for the public summary views, and for searching and copying bookmarks.

Serve Content to WAP Browsers

Once a bookmark is in the database, serving it to the mobile browser is relatively straightforward. Users view bookmarks in their personal menu as options in a WML form select list. Since some browsers don't display more than ten options in one list, bookmarks falling after the tenth on any card are placed onto the following card. Choosing a bookmark invokes a script to apply the regular expression to the target page once or multiple times, removing the match each time as described earlier. The match variables indicated to be placed in the view are retrieved, inserted into a WML deck template, and returned using the appropriate WAP MIME types. Appendix C includes the code for serving bookmarks to mobile devices.

3 Usability Analysis

3.1 User Study

Procedure

The study design focused on regular expression composition and revision. Six subjects were monitored while creating bookmarks for three different Web pages using both interfaces, for a total of 36 observations. To reduce variability, subjects bookmarked the same pages as each other and selected the same elements from those pages irrespective of the interface used.

Subjects were selected to represent a range of prior experience composing regular expressions. They selected themselves into one of three groups representing a lot, some, or minimal prior experience. Target Web pages and elements were chosen to represent a range of difficulty using both interfaces and to create realistic and useful bookmarks. The elements to capture were the following:

- all of the news headlines from Slashdot's home page at http://www.slashdot.org;
- the last sale price for Cisco stock from Nasdaq's real-time quotes page at http:// quotes.nasdaq-amex.com/quote.dll?symbol=csco&symbolx=&page=multi&mode=stock;
- the current regional forecast from the National Weather Service's Boston Office page at http://www.nws.noaa.gov/er/box/shtml/xm15.shtml.

The quantitative measure was the time to compose an expression so that it returned the specified page elements. Additional observations included how many steps subjects required to generate their first results, how many times they had to revise the match pattern until the expression worked properly, and how common difficulties or successes influenced their composition time.

New bookmarks were viewed on a mobile device and users compared the results to the same Web pages accessed through a gateway transcoder. The benchmark transcoder was Google's WAP Search at http://wap.google.com, a search engine that translates Web content to WML for presenting search results and for visiting Web pages reached by following the links to these results [19]. Alternatively, some users compared bookmark results to the same information accessed through native WAP services, such as Yahoo's stock quotes and weather.

Subjects were then asked how frequently they would use the bookmark service in preference to a transcoder, if composing expressions were to take fewer than 2, 2 to 5, 5 to 10, 10 to 20, or greater than 20 minutes. Responses indicated how sensitive users were to the setup effort required to create bookmarks. The actual times to compose expressions were then compared with the responses that indicate sensitivity to see if they were within similar ranges.

Results

Figure 10 presents aggregated results of the mean time to compose regular expressions for subjects using each interface, grouped by their level of prior experience. Right away we see that the friendly interface produces significantly shorter composition times than the advanced interface for subjects of every type. But there are other important features in the figure. First, the amount of time it takes to compose a regular expression using the friendly interface is relatively stable across all subjects. This makes sense, since the process of selecting and pasting a string to match is roughly the same for anyone working with that interface. Second, the amount of time it takes to compose a regular expression using the advanced interface increases as subjects have less experience doing so. This also makes sense, since the process of searching the page source, then entering and revising a match pattern to return specific elements requires some amount of skill. Taking these results together, the friendly interface provides more benefit to increasingly less experienced subjects, exactly what it was designed to do.

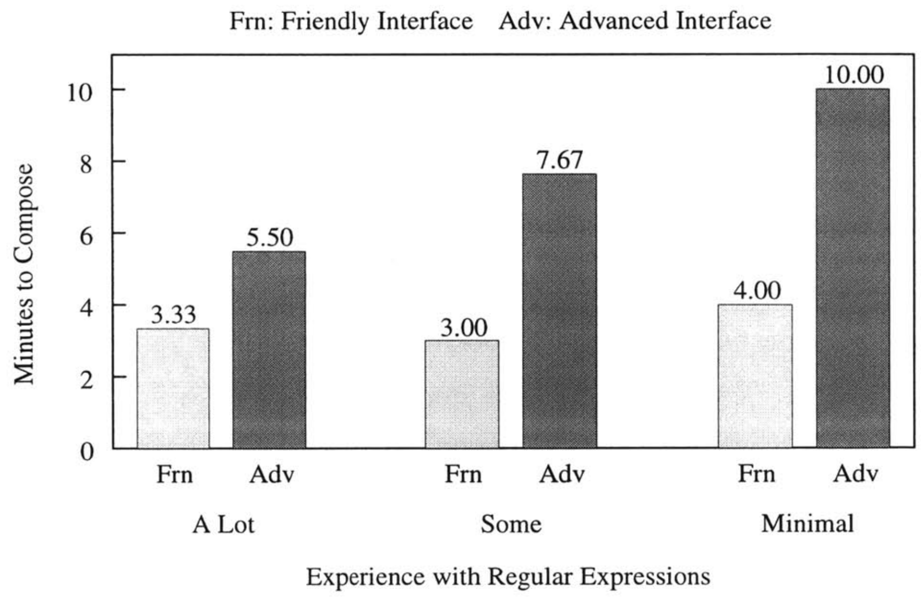


Figure 10: Mean time to compose regular expressions for subjects using friendly and advanced interfaces grouped by their level of prior experience. The difference between composition times is not statistically significant for subjects with a lot of experience, but is for those with some or minimal experience.

To confirm the significance of this conclusion, we can calculate the r^2 correlation coefficient values for the set of observed times. The r^2 indicates how different the mean values are for different subject groups, holding other factors constant. Increasing the difference between means or decreasing the variance in observations increases the r^2. Our study includes two categorical variables: the subject's prior experience and the interface used; and one quantitative variable: the composition time. So a more specific interpretation is as follows: knowing the subject's prior experience improves the prediction of that subject's composition time by r^2 percent over the best prediction without knowing it. Likewise, knowing the interface used improves the prediction in the same way. Tables 1 and 2 show the coefficient values for both cases.

A Lot	**Some**	**None**
0.13	0.57	0.71

Table 1: r^2 correlation coefficient values for the time to create bookmarks based on subjects' prior experience composing regular expressions. Responses are compared across both interfaces used holding the prior experience group constant.

So for subjects with a lot of experience, knowing which interface they used accounts for only a small percent of the difference in mean composition time. Variation in observed times accounts for the rest. Expert users therefore cannot really expect the friendly interface to provide reliably quicker times. But for subjects with decreasing experience, the interface accounts for increasingly more of the difference in composition time. These users can reasonably expect the friendly interface to improve their times. But it is important to note that this expectation only holds for comparing bookmarks that select the same page elements. Since the advanced interface allows users to compose more complex regular expressions, such as those containing more than one match variable, a direct comparison may not always be possible.

Friendly	**Advanced**
0.37	0.56

Table 2: r^2 correlation coefficient values for the time to create bookmarks based on the interface subjects used. Responses are compared across all prior experience groups holding the interface used constant.

For subjects using the friendly or advanced interface, knowing their level of experience accounts for a moderate percent of the difference in mean composition times. These two values do not differ from each other as much as one might initially think. After all, Figure 10 shows that the mean times using the friendly interface are quite close, so the r^2 should be small. Likewise, the mean times using the advanced interface are not that close, so the r^2 should be large. But recall that r^2 also varies inversely with the variance, which accounts for the difference. For all subjects, selecting and pasting a string to match with the friendly interface produces a low variance of 1.14 minutes^2. But the experimental nature of composing a match pattern from scratch with the advanced interface produces a relatively high variance of 16.31 minutes^2.

Understanding how users compose regular expressions is only half of the story. We also want to know how well creating bookmarks through the Web and accessing them when mobile compares with the alternative means of viewing the same Web content. Figure 11 presents the responses to how frequently subjects expected to use the bookmark service in preference to a transcoder, after having created their bookmarks and viewed the same content using both means.

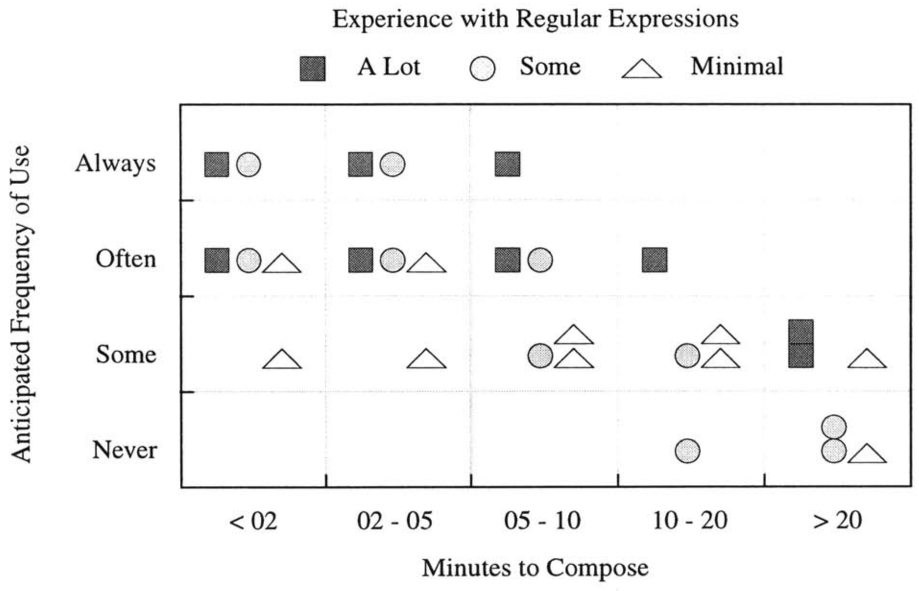


Figure 11: Responses for anticipated use of bookmark service depending on composition time. Users with a lot of experience expect to use the service regularly, with weak dependence on the time to compose regular expressions. Similarly, users with minimal experience expect to use it sometimes, with weak dependence on the time. Users with some experience are the most sensitive to composition time, and will likely switch to an alternative means to access Web content if setup effort is too great.

There is a trend along the diagonal, where subjects are more inclined to use the service as the time to compose expressions decreases. But as before, there are other important features in the data. Responses by user type do not strictly follow this trend. Subjects with a lot of experience and those with little experience had similar replies: that their choice to use the bookmark service over a transcoder depended more on the ability to access specific information when needed than on how much setup effort was required to create bookmarks. The difference between their responses is that 80 percent of experienced subjects would use the service often or always, while a similar percentage of inexperienced subjects would only use it sometimes. Subjects with some experience were the most sensitive to bookmark composition time, and therefore have the most influence the trend in the figure. 40 percent of them would rely on the service often or always if it took fewer than five minutes to compose bookmarks. Alternatively, 40 percent would use the service only sometimes or never if it took more than ten minutes. But when questioned further, nearly all subjects indicated that their responses would shift further toward favoring the bookmark service as the number of cards or decks returned by the automated transcoder increased. Since the number of cards and decks depend on the length of the original Web content, longer pages bias subjects toward the bookmark service.

For the friendly interface, the mean composition times for subjects of all experience levels fall into the range anticipated regular service use. But for the advanced interface, times for subjects with less experience fall into the range of infrequent use. Since subjects with some experience were the most sensitive to composition time, they are the most likely to discontinue using the service if their first few bookmarks are of complex pages, or require the advanced interface.

Observations

Most users followed similar processes to create bookmarks using each interface. In almost all cases, the time to compose bookmarks decreased after the first few were completed. The decrease occurred for both interfaces, but more so for the advanced interface, where the number of steps to get first results declined by a third on average. The number of times expressions were revised after these first results increased slightly on average for most users as well. But combined with their shorter overall times, this suggests a growing familiarity and willingness to experiment with the interface.

Some bookmarks though, could only be composed by transferring from the friendly to the advanced interface; for example, pages with content that is updated every minute or second. A string copied from the browser might not match into its page location because the content had changed between the time it was copied and the time the expression was evaluated. So the original string was no longer in the page. The remedy was to redesign the service to cache target pages for several minutes while users composed or edited match patterns. Their regular expressions could then be matched against the unchanging cached versions. This had the further benefit of reducing the number of GET requests to the content server.

But caching aggravates another problem: literal string matching. That is, composing a match pattern that inadvertently matches unchanging elements in the page. Section or table headings are typical examples. The problem does not prevent bookmarks from functioning, it just returns more information than may be required. More troublesome though, is the opposite: composing a match pattern that mistakes updated elements in the page for being unchanging. Table contents that appear to be headings are the most frequently encountered examples. On the user study weather page, the indicated timeframe for the current forecast (morning, afternoon, or evening) changed three times a day, but was included in a couple of subjects' match patterns as an unchanging element. In these cases, bookmarks may work fine on the current page but not when the page is updated later on. Unfortunately, the bookmark service has little way to know at composition time which elements are unchanging and which are not, especially if the page does not update for hours or days. Instead, users can check their bookmark reference status more frequently after creating new bookmarks. Doing so will help them identify literal string matches, but still not at composition time.

3.2 Public Launch

The bookmark service was launched on the public Internet at http://sirkin.lcs.arsdigita.com simultaneously with start of the user study. The site was introduced by posting its goals, description, and address to WAP service announce and developer groups. At the time, about a dozen public bookmarks were made available for users to copy and to help acquaint them with how the service works.

Visitors began registering right away, but for two weeks only a few copied any bookmarks and even fewer created new ones. The bookmark service's goals were compelling enough to bring potential users to the site, but not sufficiently so to get them involved and bring them back. Perhaps the design using regular expressions was overly technical. But even if creating bookmarks was perceived as intimidating, copying them should have been straightforward enough—particularly for the initial set of users, who were primarily WAP developers.

Several possible explanations for the lack of copying presented themselves. First, there may have been insufficient breadth in the original set of public bookmarks to pique users' interest. As a remedy, more varied bookmarks were placed into the set, but little changed over the following two weeks. Second, users may not have been made well aware of the public bookmark exchange and that copying was even possible. So their default personal menu view was modified to highlight the five most recently posted bookmarks with pointers to the public summary views, but again little changed. Third, a network effect may have been the strongest influence on activity and there were as yet too few users and bookmarks for the feedback to build strength. This explanation has some support. Once the number of users reached about thirty, the percentage of new bookmarks copied relative to those created from scratch steadily began to increase, and the trend has continued. Copies now account for about half of newly posted Web page bookmarks. But the nearly fifty current users are still likely too few to confirm any of these hypotheses.

A number of users suggested that the service should allow them to bookmark native WAP sites in addition to selected Web page elements. The feature was readily integrated into the existing procedure for creating new bookmarks. Native site bookmarks are stored in the same table as the Web page bookmarks, but can be differentiated by having null entries in the regular expression column. Once the feature was added, an update was posted to the same discussion groups as the earlier announcements. Within a few days, current users returned and new users registered and began to bookmark native sites. Within two weeks of the feature's launch, native bookmarks accounted for nearly half of all bookmarks (both new and copied) in the system.

Modifying the existing forms and writing a few scripts to enable native bookmarks took two hours. Alternatively, modifying the friendly interface to permit users to enter text from the browser view rather than the source view took three days. While we definitely want the interface to work that way, its doing so is somewhat transparent to users. It isn't a feature that can be advertised and then expected to bring in new users. So here is the first important insight: user value is not necessarily correlated with developer effort. The return-for-effort of some features can be extremely high, and for others can be rather low. It is better to build easier features first, especially if they have a lot of potential value. And the second important insight: launching and testing with real-world users is a critical step to building a more robust and useful service.

4 Conclusions and Future Work

4.1 Building a Community

User study subjects and public Internet users sorted themselves into two groups for accessing Web content when mobile:

- setup-sensitive: the more time that was required to create bookmarks, the less likely they were to use the service. At the same time, the less experience they had composing regular expressions, the less likely they were to use the service.

- setup-insensitive: the choice between using the bookmark service or an automated transcoder depended more on their need for specific elements of information than on the up-front effort required to create bookmarks.

The bookmark service should suit the needs of users in both groups, except for those who are both setup-sensitive and have only some or minimal experience with regular expressions. These users will be more satisfied using automated transcoders, except where the original content is especially long and they have no means other than the bookmark service for viewing desired information. The two features initially designed to serve these users are (1) having a less-powerful but more friendly composition interface, and (2) sharing bookmarks that are already in the system. Monitoring site activity such as the frequency of use for each interface and the number of bookmarks copied over time suggests that the second is the more powerful and often used feature.

For new and varied bookmarks to be shared, they first have to be created and saved into the system. One approach is to have regular expression experts create a large library of bookmarks.

But a library would not strengthen the collaborative network effect, which relies on a continually increasing store of new bookmarks. So the challenge is to encourage users, especially those with less regular expression experience, to contribute the new bookmarks.

Several alternatives can be implemented, ranging from an enhanced user guide to a new interface design, although the latter requires some HTML parsing, which the current interfaces avoid.

1. Provide a tutorial of example regular expressions being applied to several types of Web page. Working through examples can quickly acquaint users with the process of creating regular expressions. Recall that user study subject composition times decreased after having created only a few bookmarks.

2. Provide a dictionary of pre-defined match patterns to be applied to different target pages. Since updated content is often included in lists or tables, two of these might strip away all content except text between unordered list items, or between table definition entries. Lists and tables are sometimes nested, so the patterns would have to ignore any outer levels. The pre-defined patterns may only work on simpler pages, but they provide a bridge between the tutorial and the current interfaces.

3. Provide the means to select out Web page structural components rather than text elements. That is, a new interface would present the user with tag-delimeted page blocks, shown as they are in the original page, but wrapped within a Web form that allows users to select which ones to return to the mobile browser. Example page blocks might be paragraphs, tables, ordered and unordered lists, and images. Knowing the block types selected and their occurrence in the page, the interface could build a match pattern wrapped by the appropriate markup.

4.2 Continuing Development

Because it was designed as an ArsDigita Community System module, the bookmark service lends itself well to integration with other ACS services. For example, the collaborative areas of the site could be expanded to include a discussion forum or chat room, or users could opt to be emailed when new bookmarks of a specific type are posted.

Promising areas for further development include having the service manage bookmarks more autonomously, and increasing the variety of clients it can serve by applying multiple templates. Checking the status of bookmark references and regular expressions can be made into a scheduled procedure, performed during off-peak hours on a daily or weekly schedule. The procedure can also be made smarter. In addition to just confirming that regular expressions match the page, it can analyze their results as well. For example, if an expression is supposed to match multiple occurrences within a page but the check finds only one, the bookmark can be marked as possibly broken. Alternatively, the match itself can be examined to make certain it is not null and contains at least some printable characters. Also, since regular expressions may match pages initially but not after the content is updated—what we earlier called literal string matching—the procedure can check newer bookmarks more frequently than it might otherwise and notify the owner if any of these problems are encountered. One indication of literal string matching is where bookmarks fail to match but only periodically. In this case, the number of recent failures could be counted and matched against the number of checks over the same period; where the result is lower, a literal match has been found.

While the system is designed to place selected Web page elements into WML deck templates, placing those elements into XML, HDML or other templates should be relatively straightforward. After all, most of the system's complexity is directed toward creating and managing bookmarks. To do so, each template could be stored as a row in a bookmark-templates table. Then when a bookmark is applied, the required template would be fetched from the database, filled with the captured Web page elements, and served using the appropriate MIME types. The decision regarding which template to apply could be based either on an option set by the user or by the connection's characteristics. In the first case, each bookmark in the user's personal menu view would include a select list of the available templates. The selected value would be stored as an additional column in the `wap_bookmarks` table. In the second case, the script that serves bookmarks would read the user-agent or accept headers much as it does now, then select the template.

Another take on using multiple templates is for the bookmark service to serve results to desktop computers using an HTML template. A regular expression could be designed to capture all of the text in a target page, but eliminate any images. The result would resemble a Lynx browser.

5 References

[01] T. Kamada, "Compact HTML for Small Information Appliances," World Wide Web Consortium, W3C Note, 9 Feb. 1998. Available http://www.w3.org/TR/1998/NOTE-compactHTML-19980209

[02] T. Kamada, T. Asada, M. Ishikawa and S. Matsui, "HTML 4.0 Guidelines for Mobile Access," World Wide Web Consortium, W3C Note, 15 Mar. 1999. Available http://www.w3c.org/TR/NOTE-html40-mobile

[03] Wireless Application Protocol Forum, "Wireless Application Protocol Wireless Markup Language Specification, Version 1.3," WAP-191-WML, 19 Feb. 2000. Available http://www1.wapforum.org/tech/terms.asp?doc=WAP-191-WML-20000219-a.pdf

[04] Wireless Application Protocol Forum, "Wireless Markup Language Document Type Definition, Version 1.1," 1999. Available http://www.wapforum.org/DTD/wml_1.1.xml

[05] E. Lyngaas et al., "The Wireless FAQ," Online Document, 31 Jul. 2000, (Cited 1 Aug. 2000). Available http://www.allnetdevices.com/faq

[06] S. Pemberton et al., "XHTML 1.0: The Extensible HyperText Markup Language—A Reformulation of HTML 4.0 in XML 1.0," World Wide Web Consortium, W3C Recommendation, 26 Jan. 2000. Available http://www.w3.org/TR/WD-html-in-xml

[07] Info-Communications Development Authority of Singapore, "Infocomm Technology Roadmap (Broadband Access and Mobile Wireless)," July 2000 Release. Available http://www.ida.gov.sg/Website/IDAContent.nsf/vSubCat/Technology+DevelopmentInfocomm+Technology+Roadmap

[08] Nokia Internet Communications, "Nokia WAP Server 1.1," Product Data Sheet, May 2000. Available http://www.nokia.com/corporate/wap/gateway_case4.html

[09] T. Powell, "Wireless Web: The Final Frontier?," *ITWorld.com*, 17 Apr. 2000. Available http://mithras.itworld.com/articles/columns/f_net-powell-0417.html

[10] OpenTV, Inc., “Spyglass Prism from OpenTV: Content Delivery and Transformation Platform,” Product Data Sheet, 2000. Available http://www.opentv.com/docs/prism30.pdf

[11] A. Fox, “Information Delivery Infrastructure in the Mobile Wireless Age,” ProxiNet Corp. White Paper, 1999. Available http://www.proxinet.com/papers/idimwa.pdf

[12] J. Cummisky, J. Purdy and T. Nozick, “Digital Paths; Delivering Internet Content to the Palm of Your Hand,” Digital Paths White Paper, 1 Mar. 2000. Available http://www.digitalpaths.com/wp

[13] AvantGo, Inc., “AvantGo Channel Development Guide,” Development Guide, 2000. Available http://avantgo.com/developer/reference/AvantGoChannelDevelopment.pdf

[14] Oracle Corp., “Exchanging Data via XML: Source to XML, XML to Target,” Online document, Feb. 2000 (Cited 1 Aug. 2000). Available http://technet.oracle.com/tech/xml/info/htdocs/portaltogo/ptgpaper/ptgquote.htm

[15] T. Fielden, “Transcoding Publisher Reformats Data,” *InfoWorld.Com*, 29 May 2000. http://www.infoworld.com/articles/mt/xml/00/05/29/000529mttrans.xml

[16] Phone.com, Inc., “UP.Link WAP Enhanced Services,” Product Data Sheet, 2000. Available http://www.phone.com/pub/UPLink_WAPES.pdf

[17] Phone.com, Inc., “Interoperability and Compliance Testing,” White Paper, Mar. 2000. Available http://www.phone.com/pub/IOTWP_0400.pdf

[18] P. Greenspun, *Philip and Alex’s Guide to Web Publishing*. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1999. Available http://www.arsdigita.com/books/panda

[19] Google, Inc., “Google Goes Mobile With Industry’s First Comprehensive Wireless Search Engine,” Online document, 27 Apr. 2000 (Cited 1 Aug 2000). Available http://www.google.com/pressrel/pressrelease20.html

6 Appendices: Bookmark Service Functions

Appendices are organized by site function as described in Chapter 2. Where a task such as creating a new bookmark requires several Web forms to complete, its scripts are numbered sequentially. The highest numbered script in any sequence performs the actual database insert or update. Script sequences are included under one section heading. The service's top-level page is listed here, as it naturally falls outside any of the other functions.

Redirect to Web or WAP Function

```
# /www/wap/bookmark/index.tcl
# By sirkin@mit.edu on 07/02/00

# Top-level page for the WAP Bookmark System. Redirects the server based
# on the user-agent type. Note that WWW pages use MIME type of text/html
# whereas WML pages use text/vnd.wap.wml.

# To serve all WAP documents, the /home/aol30/*.ini file should include:

#    [ns/mimetypes]
#    .wml=text/vnd.wap.wml
#    .wmlc=application/vnd.wap.wmlc
#    .wmls=text/vnd.wap.wmlscript
#    .wmlsc=application/vnd.wap.wmlscriptc
#    .wbmp=image/vnd.wap.wbmp

# Absolute redirects seem reliable, so use ad_returnredirect.

switch [util_guess_doctype] {
    wml {
        ad_returnredirect "wap-home"
    }
    default {
        ad_returnredirect "www-home"
    }
}
```

A: Create and Manage Bookmarks

Users view their personal menu summary, change their custom view settings, create and edit bookmarks, and check their bookmark status in the Web portion of the site. `swap` allows users to change the order of bookmarks in their personal menu, and `toggle-hide` allows them to hide or unhide bookmarks from view on their mobile device. The `new` and `edit` sequences of pages are for HTML page bookmarks, while the `wap-new` and `wap-edit` sequences are for WML deck bookmarks. Users can link to `delete` bookmarks from either of the edit pages. `links-check` confirms reference HTTP status, first using a HEAD request and if that fails, using a GET. The regular expression is applied to the returned document. If either the second status request or the expression fails, the bookmark is potentially broken and the user can link to `view`, to see the bookmark in more detail, or to `links-delete`, where it can be removed.

Personal Menu Summary View

```
# /www/wap/bookmark/www-home.tcl
# By sirkin@mit.edu on 07/10/00

# Front page viewed from Web browser. Presents a list of user's bookmarks
# with links to view, edit, or hide from the mobile device. Also provides
# links to create new HTML or WML bookmarks, check link status, or view a
# public summary of other users' bookmarks.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

# An optional splash at the page top showing the five most recent bookmarks.

if { 0 } {
    set n_new 5
    set sql_query_new "
        select  *
        from    (select   max (bookmark_id) as bookmark_id,
                          max (bookmark_title) as bookmark_title
                 from     wap_bookmarks
                 where    private_p = 'f'
                 group by complete_url, the_regexp
                 order by bookmark_id desc)
        where    rownum <= $n_new"

    db_foreach $sql_query_new {
        lappend new_list "
<a href=view?[export_url_vars bookmark_id]>$bookmark_title</a>"
    }

    set new_text "
5 most recently posted bookmarks: [join $new_list ", "].
<p>"
} else {
    set new_text ""
}

set name [db_string "
    select first_names || ' ' || last_name
    from users where user_id = $user_id"]

set title "Welcome to the [ad_parameter SystemName wap-bm]"

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws "WAP Bookmark Home"]
<hr>
<table cellpadding=0 cellspacing=0 border=0 width=100%>
  <tr>
    <td align=left width=100%>
      \[ New Bookmark for <a href=wap-new>WML</a> <a href=new>HTML</a> Page |
      <a href=links-check>Check Links</a> |
      <a href=public-view>View Public Bookmarks</a> \]
    </td>
    <td align=left nowrap>
      <a href=user-guide target=new_win>User Guide</a>
    </td>
  </tr>
</table>
<p>
$new_text
<table cellpadding=2 cellspacing=0 border=0 width=100%>
  <tr bgcolor=#dddddd>
    <td align=left>
      <img src=pics/folderopen.gif border=0>
    </td>
    <td align=left width=100% nowrap>
      <b>WAP Bookmarks for $name</b>
    </td>
    <td align=right nowrap>
      Actions
    </td>
  </tr>
</table>

<table cellpadding=2 cellspacing=0 border=0 width=100%>"

# Link check dates have to be in full format otherwise checks made at different
# times on the same day will read as being the same check. Also since WML pages
# have no the_regexp field, use that to provide a link to view the page or not.

# Where value in the_regexp is null, the bookmark references a native WAP site.

set sql_query_own "
```

```
    select   bookmark_id,
             bookmark_title,
             nvl (one_line_descrip, '&nbsp;') as one_line,
             hidden_p,
             to_char(last_check_date, 'YY-MM-DD HH24:MI:SS') as check_date,
             to_char(last_live_date, 'YY-MM-DD HH24:MI:SS') as live_date,
             to_char(last_regexp_date, 'YY-MM-DD HH24:MI:SS') as regexp_date,
             decode (the_regexp, null, 't', 'f') as wap_p
    from     wap_bookmarks
    where    owner_id = $user_id
    order by sort_key"

set flg_p "f"
db_foreach $sql_query_own {

    if { $hidden_p == "t" } {
        set em_opn "<i>"
        set em_cls "</i>"
        set un_hid "view"
    } else {
        set em_opn ""
        set em_cls ""
        set un_hid "hide"
    }

    set flg_space "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;"

    if { $wap_p == "t" } {
        set edit_link "wap-edit"
        if { $live_date != $check_date } {
            set flg_p "t"
            set flg_space "<img src=pics/flg.gif border=0>"
        }
    } else {
        set edit_link "edit"
        if { $regexp_date != $check_date } {
            set flg_p "t"
            set flg_space "<img src=pics/flg.gif border=0>"
        }
    }

    append page "
  <tr bgcolor=#eeeeee>
    <td align=center>
      $flg_space
    </td>
    <td align=right nowrap>
      <a href=swap?bookmark_id=$bookmark_id&swap=up>
      <img src=pics/up.gif border=0></a>
      <a href=swap?bookmark_id=$bookmark_id&swap=dn>
      <img src=pics/dn.gif border=0></a>
    </td>
    <td align=left nowrap>
      <a href=view?[export_url_vars bookmark_id]>
      $em_opn$bookmark_title$em_cls</a>
    </td>
    <td align=left width=100% nowrap>
      <font size=-1>
        $em_opn$one_line$em_cls
      </font>
    </td>
    <td align=right nowrap>
      <font size=-1>
        <a href=toggle-hide?[export_url_vars bookmark_id]>
        $em_opn$un_hid$em_cls</a> /
        <a href=$edit_link?[export_url_vars bookmark_id]>
        $em_opn edit$em_cls</a>
      </font>
    </td>
  </tr>"

} if_no_rows {
    append page "
  You have no private bookmarks stored in the database.
  <br>
  Try copying a <a href=public-view>public bookmark</a>
  or read the <a href=user-guide target=new-win>user-guide</a>."
}

append page "
</table>
<p>
Italicized bookmarks are hidden from view on your mobile device."

if { $flg_p == "t" } {
    append page "
<br>
Flagged bookmarks did not respond properly when last checked."
}

append page "
[wbm_footer]"

ns_return 200 text/html $page
```

Swap Bookmark Order

```
# /www/wap/bookmark/swap.tcl
# By sirkin@mit.edu on 07/18/00

# Swaps the position, in the personal view summary, of the current bookmark and
# the one that immediately precedes or follows it, depending on which direction
# (up or down) the user has selected.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]
```

```
ad_page_variables {
    bookmark_id
    swap
}

# The target value is the smallest sort_key greater than the current sort_key
# if moving down in the list, or the largest sort_key smaller than the current
# sort_key if moving up in the list.

if { $swap == "up" } {
    set max_min "max"
    set lt_gt "<"
} else {
    set max_min "min"
    set lt_gt ">"
}

set current_key [db_string "
    select sort_key
    from   wap_bookmarks
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"]

set swap_key [db_string "
    select $max_min (sort_key)
    from   wap_bookmarks
    where  sort_key $lt_gt $current_key
    and    owner_id = $user_id"]

if { ![empty_string_p $swap_key] } {
    set sql_update_swap_key "
        update wap_bookmarks
        set    sort_key = $current_key
        where  sort_key = $swap_key
        and    owner_id = $user_id"

    set sql_update_current_key "
        update wap_bookmarks
        set    sort_key = $swap_key
        where  bookmark_id = $bookmark_id
        and    owner_id = $user_id"

    # Use a transaction to avoid having two bookmarks with same sort_key.

    db_transaction {
        db_dml $sql_update_swap_key
        db_dml $sql_update_current_key
    }
}

ad_returnredirect ""
```

Hide or Unhide Bookmark

```
# /www/wap/bookmark/toggle-hide.tcl
# By sirkin@mit.edu on 07/18/00

# Hides or unhides bookmarks from view on mobile browser. This might be done
# when user doesn't want to scroll through several screens of bookmarks, but
# also doesn't want to remove them from his personal menu permanently.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
}

# Set hidden_p = 't' if its current value is 'f' and vice-versa.

set sql_update "
    update wap_bookmarks
    set    hidden_p = decode (hidden_p, 't', 'f', 'f', 't')
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"

if { [catch {db_dml $sql_update} errmsg] } {
    ad_return_error "Database Update Failed" "There was an error
    making this update into the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

ad_returnredirect ""
```

View Bookmark

```
# /www/wap/bookmark/view.tcl
# By sirkin@mit.edu on 07/14/00

# Main page for viewing individual bookmark. May be called from several other
# scripts, so track the return_url. May be used to display Web page or native
# WAP site bookmarks, so check whether the_regexp exists or is null to decide
# whether to apply the regexp to the page and display the results or not.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
    {host_url {}}
    {viewed_user_id {}}
    {return_url {}}
    {regexp_view {f}}
    {device_view {f}}
}

# Where value in the_regexp is null, the bookmark references a native WAP site.

set sql_query "
    select bookmark_title,
           complete_url,
           the_regexp,
           match_vars,
           multiple_p,
           owner_id,
           preceding_text,
           following_text,
           detailed_descrip,
           decode (the_regexp, null, 't', 'f') as wap_p
    from   wap_bookmarks
    where  bookmark_id = $bookmark_id
    and    (owner_id = $user_id or private_p = 'f')"

if { [catch {db_1row $sql_query} errmsg] } {
    ad_return_error "Database Select Failed" "There was an error
    making this select from the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

# Build a custom context bar since view can be called from www-home or either
# public-view or group-view. The second two require two more backlink choices.

set choices [list "<a href=/pvt/home>Your Workspace</a>"
"<a href=\"\">WAP Bookmark Home</a>"]

if { ![empty_string_p $return_url] } {
    if { ![empty_string_p $host_url] } {
        lappend choices "<a href=group-view>Public Bookmarks</a>"
        "<a href=$return_url?[export_url_vars host_url]>One Host</a>"
    } else {
        if { ![empty_string_p $viewed_user_id] } {
            lappend choices "<a href=group-view>Public Bookmarks</a>"
           "<a href=$return_url?[export_url_vars viewed_user_id]>One User</a>"
        } else {
            lappend choices "<a href=public-view>Public Bookmarks</a>"
        }
    }
}

lappend choices "View Bookmark"
set context_bar [join $choices " : "]

# The max number of iterations through page if all occurrences is selected.

set n_max_match 20

set title "View \"$bookmark_title\""
set page "
[wbm_header $title]
<h2>$title</h2>
$context_bar
<hr>
<h3>Target Page URL</h3>
<ul>
  <li>
    \"$bookmark_title\"
  </li>
  <li>
    <a href=$complete_url target=new_win>$complete_url</a>
    (<a href=view-source:$complete_url>source</a>)
  </li>
</ul>"

# Show the owner of this bookmark a link to edit it, but use the right link
# depending on whether it is a Web page or WAP site bookmark. Show everyone
# else a link to copy it, sending along the new_bookmark_id.

if { $owner_id == $user_id } {
    if { $wap_p == "t" } {
        set edt_cpy_link "
    <a href=wap-edit?[export_url_vars bookmark_id]>Edit Bookmark</a>"
    } else {
        set edt_cpy_link "
    <a href=edit?[export_url_vars bookmark_id]>Edit Bookmark</a>"
    }
} else {
    set new_bookmark_id [db_string "
        select wap_bookmark_id_sequence.nextval from dual"]
    set edt_cpy_link "
    <a href=copy?[export_url_vars bookmark_id new_bookmark_id]>
    Copy Bookmark</a>"
}

set edt_cpy_text "
<ul>
  <li>$edt_cpy_link
  </li>
</ul>"

if { $wap_p == "f" } {

    # Default to not showing the regexp, but provide the option to do so.

    if { $regexp_view == "t" } {
        set regexp_vars "regexp_view=f"
        set regexp_text "Hide"
        append page "
<h3>Expression to Apply</h3>
<blockquote>
  <pre>[ns_quotehtml "$the_regexp"]</pre>
</blockquote>"
```

```
    } else {
        set regexp_vars "regexp_view=t"
        set regexp_text "Show"
    }

    append page "
<ul>
  <li>
    <a href=view?[export_url_vars bookmark_id host_url viewed_user_id
return_url device_view]&$regexp_vars>$regexp_text Expression</a>
  </li>
</ul>"

    if { [catch {ns_httpget $complete_url} url_text] } {
        append page "
Unable to reach host for further info. Try using View Bookmark link once more.
$edt_cpy_text
[wbm_footer]"
        ns_return 200 text/html $page
        return
    } else {
        if { ![eval $the_regexp] } {
            append page "
The regexp didn't match the page. Confirm the full expression:
<blockquote>
  <pre>[ns_quotehtml "$the_regexp"]</pre>
</blockquote>
Pages change frequently. You may have to edit the bookmark or create a new one.
$edt_cpy_text
[wbm_footer]"
            ns_return 200 text/html $page
            return
        }
    }

    # Default to showing results in a list, but provide the option to view
    # them within a textarea, to simulate the results on a mobile display.

    if { $device_view == "t" } {
        set bullet "o"
    } else {
        set bullet "  <br>
        <img src=pics/dot.gif border=0>"
    }

    if { ![empty_string_p $preceding_text] } {
        append string "
$preceding_text"
    }

    regexp {regexp .*{(.*)} \$url_text} $the_regexp match pattern

    # Patterns may begin with -text, which would cause the regexp or regsub
    # match to fail. In that case, use -- to indicate the end of switches.
    if { [string first "-" $pattern] == 0 } {
        set sw "-- "
    } else {
        set sw ""
    }

    # This regsub destructively matches the full page text, removing the match.

    set n_match 1
    set mult_regsub "regsub $sw{$pattern} \$url_text {\1} url_text"

    # Execute the following code once if not multiple_p. Otherwise, loop until
    # we find all matches in the url_text or reach a max number of iterations.

    while { [eval $mult_regsub] && $multiple_p == "t" &&
            $n_match <= $n_max_match || $n_match == 1 } {

        foreach var $match_vars {
            append string "
$bullet [ns_striphtml [set $var]]"
        }
        incr n_match
        eval $the_regexp
    }

    if { ![empty_string_p $following_text] } {
        append string "
$following_text"
    }

    if { $device_view == "t" } {
        set device_vars "device_view=f"
        set device_text "Matching String View"
        append page "
<h3>As Viewed on Device</h3>
<form>
  <blockquote>
    <textarea rows=6 cols=30 wrap>$string</textarea>
  </blockquote>
</form>"
    } else {
        set device_vars "device_view=t"
        set device_text "Device View"
        append page "
<h3>The Matching String</h3>
<blockquote>
  $string
</blockquote>"
    }

    append page "
<ul>
  <li>
```

```
    <a href=view?[export_url_vars bookmark_id host_url viewed_user_id
return_url regexp_view]&$device_vars>$device_text</a>
  </li>
</ul>"

    if { ![empty_string_p $detailed_descrip] } {
        append page "
<h3>Description</h3>
<blockquote>
  $detailed_descrip
</blockquote>"
    }
}

append page "
$edt_cpy_text
[wbm_footer]"

ns_return 200 text/html $page
```

New HTML Bookmark

```
# /www/wap/bookmark/new.tcl
# By sirkin@mit.edu on 07/10/00

# First page in a sequence to insert HTML bookmark into the database. Prompts
# user for the target page url and bookmark title. A radio-button allows user
# to opt for friendly or advanced interface. Since do not know which value is
# selected when this page loads, the user is directed to new-2-frn by default.
# If the advanced interface is selected, new-2-frn will redirect to new-2-adv.

ad_maybe_redirect_for_registration

ad_page_variables {
    {bookmark_title {}}
    {complete_url {}}
    {interface {frn}}
}

if { $interface == "frn" } {
    set frn_chk "checked"
    set adv_chk ""
} else {
    set frn_chk ""
    set adv_chk "checked"
}

set title "New Bookmark"

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] "New Bookmark"]
```

```
<hr>
<form action=new-2-frn>
  <h3>Target Page URL</h3>
  If you leave the title blank, we will attempt to get it from the page.
  <br>
  Titles with fewer than a dozen characters best fit most displays.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          Title (Optional):
        </td>
        <td align=left>
          <input size=30 name=bookmark_title value=\"$bookmark_title\">
        </td>
      </tr>
      <tr>
        <td>
          Enter the URL:
        </td>
        <td>
          <input size=30 name=complete_url value=\"$complete_url\">
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          Use
          <input type=radio name=interface value=frn $frn_chk> Friendly
          <input type=radio name=interface value=adv $adv_chk> Advanced
          Interface
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          <input type=submit value=\"Create Bookmark\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/new-2-adv.tcl

```
# By sirkin@mit.edu on 07/10/00

# Second page in a sequence to insert HTML bookmark into the database. Prompts
# user for input to build a regexp to apply to the page. Then directs to new-3.
# The advanced interface has the user enter a match pattern directly into a
# form. The regexp is assembled by (1) counting up the parenthesis-enclosed
# sub-patterns within the match pattern, (2) appending a var containing the
# url_text, and (3) appending the match vars.

ad_maybe_redirect_for_registration

ad_page_variables {
    {bookmark_title {}}
    complete_url
    {pattern {}}
    {multiple_p {f}}
}

# Check user-entered url and title. Try to find title in target page if blank.

if { [empty_string_p $complete_url] } {
    ad_return_complaint 1 "
    <li>You must enter a URL to add a bookmark."
    return
}

set bookmark_title [string trim $bookmark_title]
set complete_url [string trim $complete_url]

if { ![regexp {^[^:\"]+://} $complete_url] } {
    set complete_url "http://$complete_url"
}

if { [empty_string_p $bookmark_title] } {
    if { [catch {ns_httpget $complete_url} url_text] } {
        ad_return_complaint 1 "
        <li>Unable to detect a title as the host is unreachable."
        return
    }

    regexp -nocase {<title>([^<]*)</title>} $url_text match bookmark_title

    if { [empty_string_p $bookmark_title] } {
        ad_return_complaint 1 "
        <li>Unable to detect a title as the host does not provide one."
        return
    }
}

if { $multiple_p == "f" } {
    set one_chk "checked"
    set all_chk ""
} else {
    set one_chk ""
    set all_chk "checked"
}

# The max number of iterations through page if all occurrences is selected.
```

```
set n_max_match 20

set title "Specify \"$bookmark_title\""

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "new?[export_url_vars
bookmark_title complete_url]" "New Bookmark"] "Specify"]
<hr>
<table cellpadding=0 cellspacing=0 border=0 width=100%>
  <tr>
    <td align=right>
      <a href=user-guide target=new_win>User Guide</a>
    </td>
  </tr>
</table>
<form action=new-2-adv>
[export_form_vars bookmark_title complete_url]
  <h3>Confirm URL</h3>
  The bookmark title and target page:
  <ul>
    <li>
      \"$bookmark_title\"
    </li>
    <li>
      <a href=$complete_url target=new_win>$complete_url</a>
      (<a href=view-source:$complete_url>source</a>)
    </li>
  </ul>
  <h3>Expression to Apply</h3>
  Enter the match pattern for the page. We will build the expression.
  <br>
  You may wish to copy text from the source window to help you compose.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          Pattern:
        </td>
        <td align=left>
          <textarea rows=4 cols=30 name=pattern>
            [ns_quotehtml $pattern]
          </textarea>
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          Match
          <input type=radio name=multiple_p value=f $one_chk> One
          <input type=radio name=multiple_p value=t $all_chk> All
          Occurrences ($n_max_match max)
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          <input type=submit value=\"View Result\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>"

if { [empty_string_p $pattern] } {
    append page "
[wbm_footer]"
    ns_return 200 text/html $page
    return
} else {
    append page "
<h3>The Returned String</h3>"
}

# Web forms change a line-feed into a carraige-return line-feed, causing the
# regexp match to fail. Eliminate the carraige-return from every such pair.

regsub -all -nocase {\x0d\x0a} $pattern "\x0a" pattern

# Count the () within the pattern as the number of match vars.

set n_match_vars [regsub -all {\([^\(]*\)} $pattern "" match]

if { $n_match_vars == 0 } {
    append page "
You did not enter a sub-pattern to match, try editing your expression.
[wbm_footer]"
    ns_return 200 text/html $page
    return
}

if { [catch {ns_httpget $complete_url} url_text] } {
    append page "
Unable to reach host, try using the View Result button once more.
[wbm_footer]"
    ns_return 200 text/html $page
    return
}

# Eliminate any {} the advanced user may have wrapped the pattern with. Do
# not do the regsub earlier because the {} would then confusingly disappear
# from the form textarea.

regsub {^\{(.*)\}$} $pattern {\1} pattern
```

```
# Patterns may begin with -text, which would cause the regexp or regsub
# match to fail. In that case, use -- to indicate the end of switches.

if { [string first "-" $pattern] == 0 } {
    set sw "-- "
} else {
    set sw ""
}

set the_regexp "regexp $sw{$pattern} \$url_text match"

# To finish the regexp statement, append the match var names. The subsequent
# [eval $the_regexp] call will set values for these vars within the script.

for {set i 1} {$i <= $n_match_vars} {incr i} {
    append the_regexp " var_$i"
}

if { [catch {set match_p [eval $the_regexp]} errmsg] } {
    append page "
There is an error in the regular expression match pattern syntax:
<blockquote><pre>$errmsg</pre></blockquote>
[wbm_footer]"
    ns_return 200 text/html $page
    return
} else {
    if { $match_p == 0 } {
        append page "
The regexp didn't match the page. Confirm the full expression:
<blockquote><pre>[ns_quotehtml "$the_regexp"]</pre></blockquote>
[wbm_footer]"
        ns_return 200 text/html $page
        return
    }
}

append page "
<form action=new-3>
[export_form_vars bookmark_title complete_url the_regexp multiple_p]
<input type=hidden name=return_url value=[ad_conn url]>
  Continue to modify the regexp until the match vars capture your intent.
  <br>
  Then select the match vars you wish to include in your mobile page.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>"

if { $n_match_vars > 1 } {
    if { $multiple_p == "t" } {
        append page "
      <tr>
        <td>"

        for {set i 1} {$i <= $n_match_vars} {incr i} {
            append page "
            <input type=checkbox name=match_vars value=var_$i>
            Match Var $i"
          }
          append page "
          </td>
        </tr>"
      }
  } else {
      append page "
        <input type=hidden name=match_vars value=var_1>"
  }

  # This regsub destructively matches the full page text, removing the match.

  set n_match 1
  set mult_regsub "regsub $sw {$pattern} \$url_text {\1} url_text"

  # Execute the following code once if not multiple_p. Otherwise, loop until
  # we find all matches in the url_text or reach a max number of iterations.

  while { [eval $mult_regsub] && $multiple_p == "t" &&
          $n_match <= $n_max_match || $n_match == 1 } {

      if { $n_match_vars > 1 && $multiple_p == "t" } {
          append page "
        <tr>
          <td>
            &nbsp;
          </td>
        </tr>"
      }

      for {set i 1} {$i <= $n_match_vars} {incr i} {
          append page "
        <tr>
          <td>"

          if { $n_match_vars > 1 } {
              if { $multiple_p == "t" } {
                  append page "
            $i:"
              } else {
                  append page "
            <input type=checkbox name=match_vars value=var_$i>"
              }
          } else {
              append page "
            <img src=pics/dot.gif border=0>"
          }
          append page "
            [ns_striphtml [set var_$i]]
          </td>
        </tr>"
      }
```

```
    incr n_match
    eval $the_regexp
}

append page "
      <tr>
        <td align=left>
          <input type=submit value=\"Select Matches\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/new-2-frn.tcl

```
# By sirkin@mit.edu on 07/16/00

# Second page in a sequence to insert HTML bookmark into the database. Prompts
# user for input to build a regexp to apply to the page. Then directs to new-3.

# The friendly interface has the user enter a string to match from the target
# page directly into a form. The match pattern is assembled by (1) finding the
# string location in the page, (2) selecting out chars surrounding it, and (3)
# replacing the original string with ([^<]+) or (.+?), based on user's choice.

# The regexp is assembled by (1) counting up the parenthesis-enclosed sub-
# patterns within the match pattern, (2) appending a var containing the url_
# text, and (3) appending the match vars.

ad_maybe_redirect_for_registration

ad_page_variables {
    {bookmark_title {}}
    complete_url
    {sub_ptrn {tag}}
    {selective {mid}}
    {text {}}
    {pattern {}}
    {multiple_p {f}}
    {interface {frn}}
}

# Redirect if the advanced interface option is selected in the preceding page.

if { $interface == "adv" } {
    ad_returnredirect "new-2-adv?
    [export_url_vars bookmark_title complete_url]"
}

# Check user-entered url and title. Try to find title in target page if blank.
```

```
if { [empty_string_p $complete_url] } {
    ad_return_complaint 1 "
    <li>You must enter a URL to add a bookmark."
    return
}

set bookmark_title [string trim $bookmark_title]
set complete_url [string trim $complete_url]

if { ![regexp {^[^:\"]+://} $complete_url] } {
    set complete_url "http://$complete_url"
}

if { [empty_string_p $bookmark_title] } {
    if { [catch {ns_httpget $complete_url} url_text] } {
        ad_return_complaint 1 "
        <li>Unable to detect a title as the host is unreachable."
        return
    }

    regexp -nocase {<title>([^<]*)</title>} $url_text match bookmark_title

    if { [empty_string_p $bookmark_title] } {
        ad_return_complaint 1 "
        <li>Unable to detect a title as the host does not provide one."
        return
    }
}

if { $sub_ptrn == "tag" } {
    set sub_ptrn {([^<]+)}
    set tag_chk "checked"
    set txt_chk ""
} else {
    set sub_ptrn {(.+?)}
    set tag_chk ""
    set txt_chk "checked"
}

if { $multiple_p == "f" } {
    set one_chk "checked"
    set all_chk ""
} else {
    set one_chk ""
    set all_chk "checked"
}

set lst_chk ""
set mid_chk ""
set mst_chk ""

switch $selective {
    "lst" { set lst_chk "checked" }
    "mid" { set mid_chk "checked" }
    "mst" { set mst_chk "checked" }
}

# The max number of iterations through page if match all occurrences selected.

set n_max_match 20

set title "Specify \"$bookmark_title\""

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "new?[export_url_vars
complete_url bookmark_title]" "New Bookmark"] "Specify"]
<hr>
<table cellpadding=0 cellspacing=0 border=0 width=100%>
  <tr>
    <td align=right>
      <a href=user-guide target=new_win>User Guide</a>
    </td>
  </tr>
</table>
<form action=new-2-frn>
[export_form_vars bookmark_title complete_url]
  <h3>Confirm URL</h3>
  The bookmark title and target page:
  <ul>
    <li>
      \"$bookmark_title\"
    </li>
    <li>
      <a href=$complete_url target=new_win>$complete_url</a>
      (<a href=view-source:$complete_url>source</a>)
    </li>
  </ul>
  <h3>Text to Match</h3>
  Enter the match pattern for the page. We will then build the expression.
  <br>
  You may wish to copy text from the source window to help you compose.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
        </td>
        <td align=left>
          Apply
          <input type=radio name=sub_ptrn value=tag $tag_chk> Tag
          <input type=radio name=sub_ptrn value=txt $txt_chk> Text
          Pattern
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          Least
          <input type=radio name=selective value=lst $lst_chk>
          <input type=radio name=selective value=mid $mid_chk>
          <input type=radio name=selective value=mst $mst_chk>
          Most Selective
        </td>
      </tr>
      <tr>
        <td>
          Enter Text:
        </td>
        <td align=left>
          <textarea rows=4 cols=30 name=text>
            [ns_quotehtml $text]
          </textarea>
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          Match
          <input type=radio name=multiple_p value=f $one_chk> One
          <input type=radio name=multiple_p value=t $all_chk> All
          Occurrences ($n_max_match max)
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          <input type=submit value=\"View Result\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>"

if { [empty_string_p $text] } {
    append page "
[wbm_footer]"
    ns_return 200 text/html $page
    return
} else {
    append page "
<h3>The Returned String</h3>"
}

if { [catch {ns_httpget $complete_url} url_text] } {
    append page "
Unable to reach host, try using the View Result button once more.
[wbm_footer]"
    ns_return 200 text/html $page
    return
}

# A series of transformations on pasted-in text to match its place in page.

set text [string trim $text]

# Web forms change a line-feed into a carraige-return line-feed, causing the
# regexp match to fail. Eliminate the carraige-return from every such pair.

regsub -all -nocase {\x0d\x0a} $text "\x0a" text

# Escape certain meta-chars in the pasted-in text so that later regexps and
# regsubs will treat them as literal text rather than try to interpret them.

set esc {([[\]{}(){}?$\\*+])}
regsub -all $esc $text {\\\1} text

# Spaces and line-feeds as viewed on a Web page do not necessarily occur at
# the same place as in that page's source. Regsub any spaces and line-feeds
# in the pasted-in text into [ \n]+ (that is, either spaces or line-feeds).

set sp_lf {[ \n]+}
regsub -all $sp_lf $text $sp_lf index_text

# Find the indices in the Web page for beginning and end of pasted-in text.

set indices ""
regexp -indices $index_text $url_text indices

# A \\\? is required in the text pattern to match a \? in the ptrn_string.
# Re-run the previous esc regsub and sp_lf regsub on the original pattern.

regsub -all $esc $text {\\\1} text
regsub -all $sp_lf $text $sp_lf text

if { [empty_string_p $indices] } {
    set frame_regexp "regexp {frameset.*</frameset>} \$url_text"
    if { [eval $frame_regexp] } {
        append page "
It appears that the page you are viewing is built using html frames.
<br>
Try opening the frame that includes the target text in its own window.
[wbm_footer]"
        ns_return 200 text/html $page
        return
    } else {
        append page "
Unable to find a match in the page, try viewing the source to confirm.
<br>
Alternatively, try checking your text for leading or following spaces.
[wbm_footer]"
        ns_return 200 text/html $page
        return
    }
}

set beg_text [lindex $indices 0]
set end_text [lindex $indices 1]

set done_p "f"
set n_match 0
set n_chars 0
set n_chars_incr 12

# Build a match pattern by selecting the pasted-in text and its surrounding
# n_chars. Regsub the pasted-in text from this string and replace it with a
# sub-pattern. Match the resulting ptrn_string against the full page text.

# Increment the pattern size by n_chars and match against the full page text,
# stopping when the only match is the original pasted-in text. The result is
# the most selective pattern size option. Others are half or quarter of that.

# Initial n_chars is 12, since the smallest HTML tag is 3 chars (say, <p>)
# and there is a factor of 4 between the least and most selective options.

while { $done_p == "f"} {
    if { $n_match <= 1 && $n_chars != 0 } {
        set done_p "t"
        if { $selective == "mid" } {
            set n_chars [expr $n_chars / 2]
        } else {
            if { $selective == "lst" } {
                set n_chars [expr $n_chars / 4]
            }
        }
    } else {
        incr n_chars $n_chars_incr
    }

    set beg_ptrn [expr $beg_text - $n_chars]
    set end_ptrn [expr $end_text + $n_chars]
    set ptrn_string [string range $url_text $beg_ptrn $end_ptrn]

    # Rename ptrn_string to prevent each iteration from accumulating escapes.

    regsub -all $esc $ptrn_string {\\\1} esc_ptrn_string
    regsub $text $esc_ptrn_string $sub_ptrn pattern

    # Patterns may begin with -text, which would cause the regexp or regsub
    # match to fail. In that case, use -- to indicate the end of switches.

    if { [string first "-" $pattern] == 0 } {
        set sw "-- "
    } else {
        set sw ""
    }
    # How many times more does the text pattern match the full page text?

    set n_match [regsub -all $sw$pattern $url_text {} match]
}

set the_regexp "regexp $sw{$pattern} \$url_text match var_1"

if { [catch {set match_p [eval $the_regexp]} errmsg] } {
    append page "
There is an error in the regular expression match pattern syntax:
<blockquote><pre>$errmsg</pre></blockquote>
[wbm_footer]"
    ns_return 200 text/html $page
    return
} else {
    if { $match_p == 0 } {
        append page "
The regexp didn't match the page. Confirm the full expression:
<blockquote><pre>[ns_quotehtml "$the_regexp"]</pre></blockquote>
[wbm_footer]"
        ns_return 200 text/html $page
        return
    }
}

append page "
<form action=new-3>
<input type=hidden name=return_url value=[ad_conn url]>
<input type=hidden name=match_vars value=var_1>
[export_form_vars bookmark_title complete_url the_regexp multiple_p]
  Continue to modify the text until the match vars capture your intent.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>"

# This regsub destructively matches the full page text, removing the match.

set n_match 1
set mult_regsub "regsub $sw{$pattern} \$url_text {\1} url_text"

# Execute the following code once if not multiple_p. Otherwise, loop until
# we find all matches in the url_text or reach a max number of iterations.

while { [eval $mult_regsub] && $multiple_p == "t" &&
        $n_match <= $n_max_match || $n_match == 1 } {

    append page "
      <tr>
        <td>
          <img src=pics/dot.gif border=0> [ns_striphtml $var_1]
        </td>
      </tr>"

    incr n_match
    eval $the_regexp
```

```
}

append page "
      <tr>
        <td align=left>
          <input type=submit value=\"Select Matches\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>
<ul>
  <li>
    Transfer to <a href=new-2-adv?[export_url_vars complete_url bookmark_title
pattern multiple_p]>advanced</a> interface.
  </li>
</ul>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/new-3.tcl

```
# By sirkin@mit.edu on 07/10/00

# Third page in a sequence to insert HTML bookmark into the database. Prompts
# user for page text to appear in mobile display (such as header or footer),
# one-line and detailed descriptions. Then directs to new-4.

ad_maybe_redirect_for_registration

ad_page_variables {
    bookmark_title
    complete_url
    the_regexp
    {match_vars -multiple-list}
    multiple_p
    return_url
}

if { [empty_string_p $match_vars] } {
    ad_return_complaint 1 "<li>You did not select any match vars."
    return
}

set bookmark_id [db_string "
    select wap_bookmark_id_sequence.nextval from dual"]

set title "Layout \"$bookmark_title\""

set page "
[wbm_header $title]
<h2>$title</h2>
```

```
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "new?[export_url_vars
bookmark_title complete_url]" "New Bookmark"] [list "$return_url?
[export_url_vars bookmark_title complete_url multiple_p]" "Specify"] "Layout"]
<hr>"

if { [catch {ns_httpget $complete_url} url_text] } {
    append page "
<h3>Host Unavailable</h3>
Unable to reach host, try using the Select Match Vars button once more.
[wbm_footer]"
    ns_return 200 text/html $page
    return
} else {
    if { ![eval $the_regexp] } {
        append page "
<Expression Match Failed</h3>
Unable to match page, try using the Select Match Vars button once more.
[wbm_footer]"
        ns_return 200 text/html $page
        return
    }
}

append page "
<form action=new-4>
[export_form_vars bookmark_id bookmark_title complete_url the_regexp
match_vars multiple_p]
  <h3>Additional Text</h3>
  Enter any brief additional text to appear on the mobile page.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          Preceding text:
        </td>
        <td align=left>
          <input size=30 name=preceding_text>
        </td>
      </tr>"

regexp {regexp {(.*)} \$url_text} $the_regexp match pattern

# Patterns may begin with -text, which would cause the regexp or regsub
# match to fail. In that case, use -- to indicate the end of switches.

if { [string first "-" $pattern] == 0 } {
    set sw "-- "
} else {
    set sw ""
}

# This regsub destructively matches the full page text, removing the match.

set n_match 1
set n_max_match 20
set mult_regsub "regsub $sw{$pattern} \$url_text {\1} url_text"
# Execute the following code once if not multiple_p. Otherwise, loop until
# we find all matches in the url_text or reach a max number of iterations.

while { [eval $mult_regsub] && $multiple_p == "t" &&
        $n_match <= $n_max_match || $n_match == 1 } {

    foreach var $match_vars {
        append page "
      <tr>
        <td>
        </td>
        <td>
          <img src=pics/dot.gif border=0> [ns_striphtml [set $var]]
        </td>
      </tr>"
    }
    incr n_match
    eval $the_regexp
}

append page "
      <tr>
        <td>
          Following text:
        </td>
        <td align=left>
          <input size=30 name=following_text>
        </td>
      </tr>
    </table>
  </blockquote>
  <h3>Bookmark Description</h3>
  A few words to describe the bookmark and any extended details on its use.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          Brief Description:
        </td>
        <td align=left>
          <input size=30 name=one_line_descrip>
        </td>
      </tr>
      <tr>
        <td>
          Extended Details:
        </td>
        <td align=left>
          <textarea rows=4 cols=30 name=detailed_descrip></textarea>
        </td>
      </tr>
      <tr>
```

```
        <td>
        </td>
        <td align=left>
          <input type=submit value=\"Save Bookmark\">
        </td>
    </table>
  </blockquote>
</form>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/new-4.tcl

```
# By sirkin@mit.edu on 07/10/00

# Inserts HTML bookmark, defined in new, new-2 and new-3, into the database.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
    bookmark_title
    complete_url
    the_regexp
    match_vars
    multiple_p
    preceding_text
    following_text
    one_line_descrip
    detailed_descrip
}

if { [regexp {([^:\"]+://[^/]+)} $complete_url host_url] } {
    set host_url "$host_url/"
} else {
    set host_url ""
}

set insert "
    insert into wap_bookmarks
       (bookmark_id,
        bookmark_title,
        host_url,
        complete_url,
        the_regexp,
        match_vars,
        multiple_p,
        preceding_text,
        following_text,
        owner_id,
        sort_key,
        one_line_descrip,
        detailed_descrip)
    values
       ($bookmark_id,
        '$QQbookmark_title',
        '[DoubleApos $host_url]',
        '$QQcomplete_url',
        '$QQthe_regexp',
        '$QQmatch_vars',
        '$QQmultiple_p',
        '$QQpreceding_text',
        '$QQfollowing_text',
        $user_id,
        $bookmark_id,
        '$QQone_line_descrip',
        '$QQdetailed_descrip')"

# Protect against a double-click on the Web form inserting two copies
# into the database. The bookmark_id value was generated in new-3.

if { [catch {db_dml $insert} errmsg] } {
    if { [db_string "select count(*) from wap_bookmarks
         where bookmark_id = $bookmark_id"] == 0 } {

        ad_return_error "Database Insert Failed" "There was an error
        making this insert into the database. The error message was:
        <p><blockquote><pre>$errmsg</pre></blockquote>"
        return
    }
}

ad_returnredirect ""
```

New WML Bookmark

```
# /www/wap/bookmark/wap-new.tcl
# By sirkin@mit.edu on 08/02/00

# First page in a sequence to insert WML bookmark into the database. Prompts
# user for the target deck url and bookmark title. Then directs to wap-new-2.

ad_maybe_redirect_for_registration

ad_page_variables {
    {bookmark_title {}}
    {complete_url {}}
}

set title "New Bookmark"

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "www-home" "WAP Bookmark Home"] "New Bookmark"]
```

```
<hr>
<form action=wap-new-2>
  <h3>Target Page URL</h3>
  If you leave the title blank, we will attempt to get it from the page.
  <br>
  Titles with fewer than a dozen characters best fit most displays.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          Title (Optional):
        </td>
        <td align=left>
          <input size=30 name=bookmark_title value=\"$bookmark_title\">
        </td>
      </tr>
      <tr>
        <td>
          Enter the URL:
        </td>
        <td>
          <input size=30 name=complete_url value=\"$complete_url\">
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          <input type=submit value=\"Create Bookmark\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/wap-new-2.tcl

```
# By sirkin@mit.edu on 08/02/00

# Second page in a sequence to insert WML bookmark into the database. Prompts
# user for one-line description. Then directs to wap-new-3.

ad_maybe_redirect_for_registration

ad_page_variables {
    {bookmark_title {}}
    complete_url
}

# Check user-entered url and title. Try to find title in target deck if blank.
```

```
if { [empty_string_p $complete_url] } {
    ad_return_complaint 1 "
    <li>You must enter a URL to add a bookmark."
    return
}

if { ![regexp {^[^:\"]+://} $complete_url] } {
    set complete_url "http://$complete_url"
}

set bookmark_title [string trim $bookmark_title]
set complete_url [string trim $complete_url]

if { [empty_string_p $bookmark_title] } {
    if { [catch {ns_httpget $complete_url} url_text] } {
        ad_return_complaint 1 "
        <li>Unable to detect a title as the host is unreachable."
        return
    }

    # First look for title in WAP card. If unsuccessful, look in Web page.
    # This may work because WAP sites often have same address as Web pages.

    regexp -nocase {<card.+title="([^"]*)"} $url_text match bookmark_title

    if { [empty_string_p $bookmark_title] } {
        regexp -nocase {<title>([^<]*)</title>} $url_text match bookmark_title

        if { [empty_string_p $bookmark_title] } {
            ad_return_complaint 1 "
            <li>Unable to detect a title as the host does not provide one."
            return
        }
    }
}

set bookmark_id [db_string "
    select wap_bookmark_id_sequence.nextval from dual"]

set title "Save \"$bookmark_title\""

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "wap-new?
[export_url_vars bookmark_title complete_url]" "New Bookmark"] "Save"]
<hr>
<form action=wap-new-3>
[export_form_vars bookmark_id bookmark_title complete_url]
  <h3>Confirm URL</h3>
  The bookmark title and target page:
  <ul>
    <li>
      \"$bookmark_title\"
    </li>
    <li>
      $complete_url
    </li>
  </ul>
  <h3>Bookmark Description</h3>
  A brief sentence to appear alongside the title in your bookmark menu.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          Description:
        </td>
        <td align=left>
          <input size=30 name=one_line_descrip>
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          <input type=submit value=\"Save Bookmark\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/wap-new-3.tcl

```
# By sirkin@mit.edu on 07/10/00

# Inserts WML bookmark, defined in wap-new and wap-new-2, into the database.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
    bookmark_title
    complete_url
    one_line_descrip
}

if { [regexp {([^:\"]+://[^/]+)} $complete_url host_url] } {
    set host_url "$host_url/"
} else {
    set host_url ""
}

set insert "
```

```
    insert into wap_bookmarks
       (bookmark_id,
        bookmark_title,
        host_url,
        complete_url,
        owner_id,
        sort_key,
        one_line_descrip)
    values
       ($bookmark_id,
        '$QQbookmark_title',
        '[DoubleApos $host_url]',
        '$QQcomplete_url',
        $user_id,
        $bookmark_id,
        '$QQone_line_descrip')"

# Protect against a double-click on the Web form inserting two copies
# into the database. The bookmark_id value was generated in wap-new-2.

if { [catch {db_dml $insert} errmsg] } {
    if { [db_string "select count(*) from wap_bookmarks
         where bookmark_id = $bookmark_id"] == 0 } {

        ad_return_error "Database Insert Failed" "There was an error
        making this insert into the database. The error message was:
        <p><blockquote><pre>$errmsg</pre></blockquote>"
        return
    }
}

ad_returnredirect ""
```

Edit HTML Bookmark

```
# /www/wap/bookmark/edit.tcl
# By sirkin@mit.edu on 07/12/00

# First page in a sequence to update HTML bookmark in the database. Prompts
# user for the target page url and bookmark title, and whether bookmark is
# public or private. Then directs to edit-2.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
}

set sql_query "
    select bookmark_title,
           complete_url,
           private_p
    from   wap_bookmarks
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"

if { [catch {db_1row $sql_query} errmsg] } {
    ad_return_error "Database Select Failed" "There was an error
    making this select from the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

if { $private_p == "f" } {
    set pub_chk "checked"
    set prv_chk ""
} else {
    set pub_chk ""
    set prv_chk "checked"
}

set title "Edit \"$bookmark_title\""

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] "Edit Bookmark"]
<hr>
<form action=edit-2>
[export_form_vars bookmark_id]
  <h3>Target Page URL</h3>
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          URL:
        </td>
        <td>
          <input size=30 name=complete_url value=$complete_url>
        </td>
      </tr>
      <tr>
        <td>
          Title:
        </td>
        <td align=left>
          <input size=30 name=bookmark_title value=\"$bookmark_title\">
        </td>
      </tr>
    </table>
  </blockquote>
  <h3>Privacy</h3>
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
```

```
          <input type=radio name=private_p value=f $pub_chk> Public
          <input type=radio name=private_p value=t $prv_chk> Private
        </td>
      </tr>
      <tr>
        <td>
          <input type=submit value=\"Edit Bookmark\">
        </td>
      </tr>
    </table>
  </blockquote>
  <ul>
    <li>
      <a href=delete?[export_url_vars bookmark_id]>Delete this bookmark</a>
    </li>
  </ul>
</form>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/edit-2.tcl

```
# By sirkin@mit.edu on 07/12/00

# Second page in a sequence to update HTML bookmark in the database. Prompts
# user for input to build a regeaxp to apply to the page. Directs to edit-3.

# Follows the advanced interface in new-2-adv, where the user enters a match
# pattern directly into a form. The regexp and match_vars in the database are
# applied first, but user can edit the match pattern, creating a new regexp.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
    bookmark_title
    complete_url
    {pattern {}}
    {multiple_p {}}
    private_p
}

# Check user-entered url and title. Try to find title in target page if blank.

if { [empty_string_p $complete_url] } {
    ad_return_complaint 1 "
    <li>You must enter a URL to edit a bookmark."
    return
}

set bookmark_title [string trim $bookmark_title]
set complete_url [string trim $complete_url]
```

```
if { ![regexp {^[^:\"]+://} $complete_url] } {
    set complete_url "http://$complete_url"
}

if { [empty_string_p $bookmark_title] } {
    if { [catch {ns_httpget $complete_url} url_text] } {
        ad_return_complaint 1 "
        <li>Unable to detect a title as the host is unreachable."
        return
    }

    # First look for title in WAP card. If unsuccessful, look in Web page.
    # This may work because WAP sites often have same address as Web pages.

    regexp -nocase {<title>([^<]*)</title>} $url_text match bookmark_title

    if { [empty_string_p $bookmark_title] } {
        ad_return_complaint 1 "
        <li>Unable to detect a title as the host does not provide one."
        return
    }
}

# The max number of iterations through page if all occurrences is selected.

set n_max_match 20

set title "Specify \"$bookmark_title\""

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "edit?[export_url_vars
bookmark_id]" "Edit Bookmark"] "Specify"]
<hr>
<form action=edit-2>
[export_form_vars bookmark_id bookmark_title complete_url private_p]
  <table cellpadding=0 cellspacing=0 border=0 width=100%>
    <tr>
      <td align=right>
        <a href=user-guide target=new_win>User Guide</a>
      </td>
    </tr>
  </table>
  <h3>Confirm URL</h3>
  The bookmark title and target page:
  <ul>
    <li>
      \"[string trim $bookmark_title]\"
    </li>
    <li>
      <a href=$complete_url target=new_win>$complete_url</a>
      (<a href=view-source:$complete_url>source</a>)
    </li>
  </ul>"
set sql_query "
    select the_regexp,
           match_vars,
           multiple_p as db_multiple_p
    from   wap_bookmarks
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"

if { [catch {db_1row $sql_query} errmsg] } {
    ad_return_error "Database Select Failed" "There was an error
    making this select from the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

# Extract the pattern for the regexp, and use the multiple_p value from the
# db on first viewing the page. Otherwise use the (possibly revised) values
# submitted through form.

if { [empty_string_p $pattern] } {
    regexp {regexp .*{(.*)} \$url_text} $the_regexp match pattern
    set multiple_p $db_multiple_p
}

if { $multiple_p == "f" } {
    set one_chk "checked"
    set all_chk ""
} else {
    set one_chk ""
    set all_chk "checked"
}

append page "
  <h3>Expression to Apply</h3>
  Edit the match pattern (only) for the page. We will build the expression.
  <br>
  You may wish to copy text from the source window to help you edit.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          Pattern:
        </td>
        <td align=left>
          <textarea rows=4 cols=30 name=pattern>
            [ns_quotehtml $pattern]
          </textarea>
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
```

```
          Match
          <input type=radio name=multiple_p value=f $one_chk> One
          <input type=radio name=multiple_p value=t $all_chk> All
          Occurrences ($n_max_match max)
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          <input type=submit value=\"View Result\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>
"

if { [empty_string_p $pattern] } {
    append page "
[wbm_footer]"
    ns_return 200 text/html $page
    return
} else {
    append page "
<h3>The Returned String</h3>"
}

# Web forms change a line-feed into a carraige-return line-feed, causing the
# regexp match to fail. Eliminate the carraige-return from every such pair.

regsub -all -nocase {\x0d\x0a} $pattern "\x0a" pattern

# Count the () within the pattern as the number of match vars.

set n_match_vars [regsub -all {\([^\(]*\)} $pattern "" match]

if { $n_match_vars == 0 } {
    append page "
You did not enter a sub-pattern to match, try editing your expression.
[wbm_footer]"
    ns_return 200 text/html $page
    return
}

if { [catch {ns_httpget $complete_url} url_text] } {
    append page "
Unable to reach host, try using the View Result button once more.
[wbm_footer]
    ns_return 200 text/html $page
    return
}

# Eliminate any {} the advanced user may have wrapped the pattern with. Do
# not do the regsub earlier because the {} would then confusingly disappear
# from the form textarea.

regsub {^\{.*\}$} $pattern {\1} pattern

# Patterns may begin with -text, which would cause the regexp or regsub
# match to fail. In that case, use -- to indicate the end of switches.

if { [string first "-" $pattern] == 0 } {
    set sw "-- "
} else {
    set sw ""
}

set the_regexp "regexp $sw{$pattern} \$url_text match"

# To finish the regexp statement, append the match var names. The subsequent
# [eval $the_regexp] call will set values for these vars within the script.

for {set i 1} {$i <= $n_match_vars} {incr i} {
    append the_regexp " var_$i"
}

if { [catch {set match_p [eval $the_regexp]} errmsg] } {
    append page "
There is an error in the regular expression match pattern syntax:
<blockquote><pre>$errmsg</pre></blockquote>
[wbm_footer]"
    ns_return 200 text/html $page
    return
} else {
    if { $match_p == 0 } {
        append page "
The regexp didn't match the page. Confirm the full expression:
<blockquote><pre>[ns_quotehtml "$the_regexp"]</pre></blockquote>
[wbm_footer]"
        ns_return 200 text/html $page
        return
    }
}

append page "
Continue to modify the regexp until the matches capture your intent.
<br>
Then select the matches you wish to include in your mobile deck.
<form action=edit-3>
[export_form_vars bookmark_id bookmark_title complete_url the_regexp
multiple_p private_p]
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>"

if { $n_match_vars > 1 } {
    if { $multiple_p == "t" } {
        append page "
```

```
      <tr>
        <td>"

        for {set i 1} {$i <= $n_match_vars} {incr i} {
            if { [lsearch $match_vars "var_$i"] != -1 } {
                set chk "checked"
            } else {
                set chk ""
            }
            append page "
          <input type=checkbox name=\"match_vars\" value=\"var_$i\" $chk>
          Match Var $i"
        }
        append page "
        </td>
      </tr>"
    }
} else {
    append page "
      <input type=hidden name=\"match_vars\" value=\"var_1\">"
}

# This regsub destructively matches the full page text, removing the match.

set n_match 1
set mult_regsub "regsub $sw {$pattern} \$url_text {\1} url_text"

# Execute the following code once if not multiple_p. Otherwise, loop until
# we find all matches in the url_text or reach a max number of iterations.

while { [eval $mult_regsub] && $multiple_p == "t" &&
        $n_match <= $n_max_match || $n_match == 1 } {

    if { $n_match_vars > 1 && $multiple_p == "t" } {
        append page "
      <tr>
        <td>
          &nbsp;
        </td>
      </tr>"
    }

    for {set i 1} {$i <= $n_match_vars} {incr i} {
        append page "
      <tr>
        <td>"

        if { $n_match_vars > 1 } {
            if { $multiple_p == "t" } {
                append page "
          $i:"
            } else {
                if { [lsearch $match_vars "var_$i"] != -1 } {
                    set chk "checked"
                } else {
                    set chk ""
                }
                append page "
          <input type=checkbox name=match_vars value=var_$i $chk>"
            }
        } else {
            append page "
          <img src=pics/dot.gif border=0>"
        }
        append page "
          [ns_striphtml [set var_$i]]
        </td>
      </tr>"
    }
    incr n_match
    eval $the_regexp
}

append page "
      <tr>
        <td align=left>
          <input type=submit value=\"Select Matches\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/edit-3.tcl

```
# By sirkin@mit.edu on 07/12/00

# Third page in a sequence to update HTML bookmark in the database. Prompts
# user for page text to appear in mobile display (such as header or footer),
# one-line and detailed descriptions. Then directs to edit-4.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
    bookmark_title
    complete_url
    the_regexp
    {match_vars -multiple-list}
    multiple_p
    private_p
}

if { [empty_string_p $match_vars] } {
    ad_return_complaint 1 "<li>You did not select any match vars."
    return
}

# The max number of iterations through page if all occurrences is selected.

set n_max_match 20

set title "Layout \"$bookmark_title\""

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "edit?[export_url_vars
bookmark_id]" "Edit Bookmark"] [list "edit-2?[export_url_vars bookmark_id
bookmark_title complete_url multiple_p private_p]" "Specify"] "Layout"]
<hr>"

set sql_query "
    select preceding_text,
           following_text,
           one_line_descrip,
           detailed_descrip
    from   wap_bookmarks
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"

if { [catch {db_1row $sql_query} errmsg] } {
    ad_return_error "Database Select Failed" "There was an error
    making this select from the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

if { [catch {ns_httpget $complete_url} url_text] } {
    append page "
<h3>Host Unavailable</h3>
Unable to reach host, try using the Select Match Vars button once more.
[wbm_footer]"
    ns_return 200 text/html 200 $page
    return
} else {
    if { ![eval $the_regexp] } {
        append page "
<Expression Match Failed</h3>
Unable to match page, try using the Select Match Vars button once more.
[wbm_footer]"
        ns_return 200 text/html 200 $page
        return
    }
}

append page "
<form action=edit-4>
```

```
[export_form_vars bookmark_id bookmark_title complete_url the_regexp
match_vars multiple_p private_p]
  <h3>Additional Text</h3>
  Revise any brief additional text to appear on the mobile page.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          Preceding text:
        </td>
        <td align=left>
          <input size=30 name=preceding_text value=\"$preceding_text\">
        </td>
      </tr>"

regexp {regexp {(.*)} \$url_text} $the_regexp match pattern

# Patterns may begin with -text, which would cause the regexp or regsub
# match to fail. In that case, use -- to indicate the end of switches.

if { [string first "-" $pattern] == 0 } {
    set sw "-- "
} else {
    set sw ""
}

# This regsub destructively matches the full page text, removing the match.

set n_match 1
set mult_regsub "regsub $sw{$pattern} \$url_text {\1} url_text"

# Execute the following code once if not multiple_p. Otherwise, loop until
# we find all matches in the url_text or reach a max number of iterations.

while { [eval $mult_regsub] && $multiple_p == "t" &&
        $n_match <= $n_max_match || $n_match == 1 } {

    foreach var $match_vars {
        append page "
      <tr>
        <td>
        </td>
        <td>
          <img src=pics/dot.gif border=0> [ns_striphtml [set $var]]
        </td>
      </tr>"
    }
    incr n_match
    eval $the_regexp
}

append page "
      <tr>
        <td>
```

```
          Following text:
        </td>
        <td align=left>
          <input size=30 name=following_text value=\"$following_text\">
        </td>
      </tr>
    </table>
  </blockquote>
  <h3>Bookmark Description</h3>
  A few words to describe the bookmark and any extended details on its use.
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          Brief Description:
        </td>
        <td align=left>
          <input size=30 name=one_line_descrip
          value=\"[ns_quotehtml $one_line_descrip]\">
        </td>
      </tr>
      <tr>
        <td>
          Extended Details:
        </td>
        <td align=left>
          <textarea rows=4 cols=30 name=detailed_descrip>
              [ns_quotehtml $detailed_descrip]
          </textarea>
        </td>
      </tr>
      <tr>
        <td>
        </td>
        <td align=left>
          <input type=submit value=\"Save All Changes\">
        </td>
    </table>
  </blockquote>
</form>
[wbm_footer]"

ns_return 200 text/html $page


# /www/wap/bookmark/edit-4.tcl
# By sirkin@mit.edu on 07/12/00

# Updates HTML bookmark, defined in edit, edit-2 and edit-3, in the database.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
    bookmark_title
    complete_url
    the_regexp
    match_vars
    multiple_p
    private_p
    preceding_text
    following_text
    one_line_descrip
    detailed_descrip
}

if { [regexp {([^:']+://[^/]+)} $complete_url host_url] } {
    set host_url "$host_url/"
} else {
    set host_url ""
}

set update "
    update wap_bookmarks
    set    bookmark_title = '$QQbookmark_title',
           host_url = '[DoubleApos $host_url]',
           complete_url = '$QQcomplete_url',
           the_regexp = '$QQthe_regexp',
           match_vars = '$QQmatch_vars',
           multiple_p = '$QQmultiple_p',
           private_p = '$QQprivate_p',
           preceding_text = '$QQpreceding_text',
           following_text = '$QQfollowing_text',
           one_line_descrip = '$QQone_line_descrip',
           detailed_descrip = '$QQdetailed_descrip',
           modification_date = sysdate
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"

if { [catch {db_dml $update} errmsg] } {
    ad_return_error "Database Update Failed" "There was an error
    making this update into the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

ad_returnredirect ""
```

Edit WML Bookmark

```
# /www/wap/bookmark/wap-edit.tcl
# By sirkin@mit.edu on 08/02/00

# First page in a sequence to update WML bookmark in the database. Prompts
# user for the target deck url and bookmark title, and whether bookmark is
# public or private. Then directs to wap-edit-2.
```

```
ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
}

set sql_query "
    select bookmark_title,
           complete_url,
           private_p,
           one_line_descrip
    from   wap_bookmarks
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"

if { [catch {db_1row $sql_query} errmsg] } {
    ad_return_error "Database Select Failed" "There was an error
    making this select from the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

if { $private_p == "f" } {
    set pub_chk "checked"
    set prv_chk ""
} else {
    set pub_chk ""
    set prv_chk "checked"
}

set title "Edit \"$bookmark_title\""

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] "Edit Bookmark"]
<hr>
<form action=wap-edit-2>
[export_form_vars bookmark_id]
  <h3>Target Page URL</h3>
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          URL:
        </td>
        <td>
          <input size=30 name=complete_url value=$complete_url>
        </td>
      </tr>
      <tr>
        <td>
          Title:
```

```
        </td>
        <td align=left>
          <input size=30 name=bookmark_title value=\"$bookmark_title\">
        </td>
      </tr>
    </table>
  </blockquote>
  <h3>Bookmark Description</h3>
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          Text:
        </td>
        <td>
          <input size=30 name=one_line_descrip
          value=\"[ns_quotehtml $one_line_descrip]\">
        </td>
      </tr>
    </table>
  </blockquote>
  <h3>Privacy</h3>
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          <input type=radio name=private_p value=f $pub_chk> Public
          <input type=radio name=private_p value=t $prv_chk> Private
        </td>
      </tr>
      <tr>
        <td>
          <input type=submit value=\"Edit Bookmark\">
        </td>
      </tr>
    </table>
  </blockquote>
  <ul>
    <li>
      <a href=delete?[export_url_vars bookmark_id]>Delete this bookmark</a>
    </li>
  </ul>
</form>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/wap-edit-2.tcl

```
# By sirkin@mit.edu on 08/02/00

# Updates WML bookmark, defined in wap-edit, in the database.

ad_maybe_redirect_for_registration
```

```
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
    bookmark_title
    complete_url
    private_p
    one_line_descrip
}

if { [regexp {([^:\"]+://[^/]+)} $complete_url host_url] } {
    set host_url "$host_url/"
} else {
    set host_url ""
}

set update "
    update wap_bookmarks
    set    bookmark_title = '$QQbookmark_title',
           host_url = '[DoubleApos $host_url]',
           complete_url = '$QQcomplete_url',
           private_p = '$QQprivate_p',
           one_line_descrip = '$QQone_line_descrip',
           modification_date = sysdate
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"

if { [catch {db_dml $update} errmsg] } {
    ad_return_error "Database Update Failed" "There was an error
    making this update into the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

ad_returnredirect ""
```

Delete Bookmark

```
# /www/wap/bookmark/delete.tcl
# By sirkin@mit.edu on 07/16/00

# First page in sequence to delete bookmark from the database. Checks for
# confirmation the user intends to delete. It doesn't matter if it is Web
# page or WAP site bookmark, only that the current user owns it.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
}
set n_deletable_links [db_string "
    select count (*)
    from   wap_bookmarks
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"]

if { $n_deletable_links == 0 } {
    ad_return_complaint 1 "
    <li>You do not own this bookmark and cannot delete it."
    return
}

set bookmark_title [db_string "
    select bookmark_title
    from   wap_bookmarks
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"]

set title "Delete \"$bookmark_title\""

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "edit?[export_url_vars
bookmark_id]" "Edit Bookmark"] "Delete"]
<hr>
<form action=delete-2>
[export_form_vars bookmark_id]
  <h3>Confirm Action</h3>
  Are you certain you want to delete \"$bookmark_title\"?
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          <input type=submit value=\"Confirm Delete\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/delete-2.tcl

```
# By sirkin@mit.edu on 07/16/00

# Deletes bookmark confirmed in delete from the database.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
}
```

```
set delete "
    delete from wap_bookmarks
    where  bookmark_id = $bookmark_id
    and    owner_id = $user_id"

if { [catch {db_dml $delete} errmsg] } {
    ad_return_error "Database Delete Failed" "There was an error
    making this delete from the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

ad_returnredirect ""
```

Check Bookmark Status

```
# /www/wap/bookmark/links-check.tcl
# By sirkin@mit.edu on 07/14/00

# Checks the status of bookmark references and regexps in one user's menu.
# Also fetches the page's meta-tag keywords and description. For bookmarks
# whose references cannot be reached or regexps don't work, provide a form
# checkbox to delete. For those whose regexp doesn't work, also provide a
# link to edit the bookmark's regexp.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

set title "Check Bookmark Links"

ad_return_top_of_page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] "Check Bookmark Links"]
<hr>
To delete bookmarks that are not okay, mark select checkboxes and submit.
<br>
Please be patient as it take a while for some pages to fully load and check.
<form action=links-delete>
[export_form_vars delete_list]
  <table cellpadding=2 cellspacing=0 border=0 width=100%>
    <tr bgcolor=#dddddd>
      <td align=left>
        <img src=pics/folderopen.gif border=0>
      </td>
      <td align=left width=100%>
        <b>Checked WAP Bookmark Links</b>
      </td>
      <td align=right nowrap>
         Status
      </td>
    </tr>
  </table>"
```

```
set sql_query "
    select   bookmark_id,
             bookmark_title,
             complete_url,
             the_regexp
    from     wap_bookmarks
    where    owner_id = $user_id
    order by sort_key"

set links_list [db_list_of_lists $sql_query]
db_release_unused_handles

set return_text "
<ul>
  <li>
    <a href=\"\">Return home</a>
  </li>
</ul>"

if { [llength $links_list] == 0 } {
    ns_write "
  You have no bookmarks to check.
</form>
$return_text
[wbm_footer]"
    return
}

set n_dead 0
foreach link $links_list {
    set meta_clause ""
    set live_clause ""
    set regexp_clause ""

    set bookmark_id [lindex $link 0]
    set bookmark_title [lindex $link 1]
    set complete_url [lindex $link 2]
    set the_regexp [lindex $link 3]

    # Follow the ACS style for checking HTTP status, first using HEAD as it
    # is quick, second using GET if the first try fails. Some program-backed
    # servers return a no status (empty) or 404 to a HEAD but not to a GET.

    if { [catch {get_http_status $complete_url 0} response] } {
        set bullet "
        <input type=checkbox name=delete_list value=$bookmark_id>"
        set result_text "
        No response (1st try)"
        incr n_dead
    }

    if { $response == 404 || $response == 405 || $response == 500 ||
         [empty_string_p $response] } {
```

```
        if { [catch {get_http_status $complete_url 1} response] } {
            set bullet "
            <input type=checkbox name=delete_list value=$bookmark_id>"
                set result_text "
                No response (2nd try)"
                incr n_dead
        }
    }

    if { $response != 200 && $response != 302 } {
        set bullet "
        <input type=checkbox name=delete_list value=$bookmark_id>"
        set result_text "
        Page not found ($response)"
        incr n_dead
    } else {

        # Reached the page content. Now get meta-tags and run regexp.

        set descrip ""
        set keywords ""
        regexp -nocase {<meta name="description" content="([^"]*)">}
        $url_text match descrip
        if { [string length $descrip] > 990 } {
            set descrip "[string range $keywords 0 990]..."
        }
        regexp -nocase {<meta name="keywords" content="([^"]*)">}
              $url_text match keywords
        if { [string length $keywords] > 990 } {
            set keywords "[string range $keywords 0 990]..."
        }
        set meta_clause ", meta_descrip = '[DoubleApos $descrip]'"
        append meta_clause ", meta_keywords = '[DoubleApos $keywords]'"

        if { [empty_string_p $the_regexp] } {
            set live_clause ", last_live_date = sysdate"
            set bullet "
        <img src=pics/dot.gif border=0>"
            set result_text "
        Bookmark okay"
        } else {
            if { ![catch {ns_httpget $complete_url} url_text] } {
                set live_clause ", last_live_date = sysdate"
                if { ![eval $the_regexp] } {
                    set bullet "
        <input type=checkbox name=delete_list value=$bookmark_id>"
                    set result_text "
        Regexp failed (<a href=edit?[export_url_vars bookmark_id]>edit</a>)"
                    incr n_dead
                } else {
                    set regexp_clause ", last_regexp_date = sysdate"
                    set bullet "
        <img src=pics/dot.gif border=0>"
                    set result_text "
        Bookmark okay"
                    }
                }
            }
        }

    ns_write "
  <table cellpadding=2 cellspacing=0 border=0 width=100%>
    <tr bgcolor=#eeeeee>
      <td>
        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
      </td>
      <td align=right>$bullet
      </td>
      <td align=left width=100%>
        <a href=view?[export_url_vars bookmark_id]>$bookmark_title</a>
      </td>
      <td align=right nowrap>
        <font size=-1>$result_text
        </font>
      </td>
    </tr>
  </table>"

    set update "
        update wap_bookmarks
        set    last_check_date = sysdate$meta_clause$live_clause$regexp_clause
        where  bookmark_id = $bookmark_id
        and    owner_id = $user_id"

    db_dml $update
}

if { $n_dead > 0 } {
    ns_write "
  <blockquote>
    <table cellpadding=2 cellspacing=0 border=0>
      <tr>
        <td align=left>
          <input type=submit value=\"Delete Marked Links\">
        </td>
      </tr>
    </table>
  </blockquote>"
}

ns_write "
</form>
$return_text
[wbm_footer]"
```

Delete Bookmarks with Bad Status

```
# /www/wap/bookmark/links-delete.tcl
# By sirkin@mit.edu on 07/16/00

# First page in sequence to delete bookmarks with bad status from the database.
# Checks for confirmation the user intends to delete. It doesn't matter if they
# are Web page or WAP site bookmarks, only that the current user owns them.

# Their bad status was determined in links-check because ither the host was not
# reachable or the regexp failed to match the target page.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    {delete_list -multiple-list}
}

if { [empty_string_p $delete_list] } {
    ad_return_complaint 1 "
    <li>You did not select any links to delete."
    return
}

set deletable_list [db_list_of_lists "
    select bookmark_id,
           bookmark_title
    from   wap_bookmarks
    where  bookmark_id in ([join $delete_list ", "])
    and    owner_id = $user_id"]

set n_deletable_links [llength $deletable_list]
if { $n_deletable_links == 1 } {
    set title "Delete \"[lindex [lindex $deletable_list 0] 1]\""
} else {
    set title "Delete Select Bookmarks"
}

set page "
[wbm_header "$title"]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "links-check" "Check
Bookmark Links"] "Delete"]
<hr>
<form action=links-delete-2>
  <h3>Confirm Action</h3>
  Are you certain you want to delete"

foreach link $deletable_list {
    set bookmark_id [lindex $link 0]
    set bookmark_title [lindex $link 1]
    lappend link_list "
  \"$bookmark_title\"
  <input type=hidden name=delete_list value=$bookmark_id>"
}

set link_text [join $link_list " and "]

append page "
$link_text
  <blockquote>
    <table cellpadding=0 cellspacing=0 border=0>
      <tr>
        <td>
          <input type=submit value=\"Confirm Delete\">
        </td>
      </tr>
    </table>
  </blockquote>
</form>
[wbm_footer]"

ns_return 200 text/html $page
```

# /www/wap/bookmark/links-delete-2.tcl

```
# By sirkin@mit.edu on 07/14/00

# Deletes bookmarks confirmed in links-delete from the database.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    {delete_list -multiple-list}
}

set delete "
    delete from wap_bookmarks
    where  bookmark_id in ([join $delete_list ", "])
    and    owner_id = $user_id"

if { [catch {db_dml $delete} errmsg] } {
    ad_return_error "Database Delete Failed" "There was an error
    making this delete from the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

ad_returnredirect ""
```

B: Share and Copy Public Bookmarks

Users view public bookmarks using several summaries, all accessed in the Web portion of the site. `public-view` lists all bookmarks ordered by their recency (when they were saved) or their frequency (how many copies). `group-view` does not list bookmarks individually, but as they are grouped, either by host or owner. Following a link on this page takes the user to `host-view` or `user-view`, where bookmarks are listed accordingly.

From `public-view`, users can search for bookmarks by keywords appearing in their title, host or complete address, or owner-provided or meta-tag descriptions. From `view`, users can copy any public bookmark.

Public Summary View

```
# /www/wap/bookmark/public-view.tcl
# By sirkin@mit.edu on 07/30/00

# Lists all public bookmarks as links along with their one-line descriptions.
# Following a link takes the user to view, where that bookmark can be edited
# by the owner or copied into a personal menu view by anyone else.

ad_maybe_redirect_for_registration

ad_page_variables {
    {sort {bookmark_id}}
    {num_page {0}}
}

if { $sort == "bookmark_id" } {
    set sort_menu "recency | <a href=public-view?sort=n_copies&
    num_page=$num_page>popularity</a>"
} else {
    set sort_menu "<a href=public-view?sort=bookmark_id&
    num_page=$num_page>recency</a> | popularity"
}

set links_list [db_list_of_lists "
    select   count (*) as n_copies,
             max (bookmark_id) as bookmark_id,
             max (bookmark_title) as bookmark_title,
             nvl (max (one_line_descrip), '&nbsp;') as one_line
    from     wap_bookmarks
    where    private_p = 'f'
    group by complete_url, the_regexp
    order by $sort desc"]

set n_total_links [llength $links_list]
```

```
# Provide scroll options so not all bookmarks have to be listed on one page.

set n_view 10
set min_link [expr $n_view * $num_page + 1]
if { $n_total_links > [expr $min_link + $n_view - 1] } {
    set max_link [expr $min_link + $n_view - 1]
} else {
    set max_link $n_total_links
}

set num_text "$min_link - $max_link of $n_total_links"

set prev_link "<a href=public-view?sort=$sort&
num_page=[expr $num_page - 1]><img src=pics/lf.gif border=0></a>"
set more_link "<a href=public-view?sort=$sort&
num_page=[expr $num_page + 1]><img src=pics/rt.gif border=0></a>"

set num_menu "&nbsp;"
if { $n_total_links > 0 } {
    if { $min_link > $n_view } {
        append num_menu "$prev_link&nbsp;"
    }
    append num_menu $num_text
    if { $max_link < $n_total_links } {
        append num_menu "&nbsp;$more_link"
    }
}

set title "Public Bookmarks"

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] "Public Bookmarks"]
<hr>
Sort by $sort_menu
<p>
<table cellpadding=2 cellspacing=0 border=0 width=100%>
  <tr bgcolor=#dddddd>
    <td align=left>
      <img src=pics/folderopen.gif border=0>
    </td>
    <td align=left width=100% nowrap>
      <b>Public WAP Bookmarks</b>
    </td>
    <td align=right nowrap>
      $num_menu
    </td>
  </tr>
</table>
<table cellpadding=2 cellspacing=0 border=0 width=100%>"

if { $n_total_links == 0 } {
    append page "
  There are no public bookmarks stored in the database."
} else {
    set return_url [ad_conn url]
    for {set i $min_link} {$i <= $max_link} {incr i} {
        set link [lindex $links_list [expr $i - 1]]

        set n_copies [lindex $link 0]
        set bookmark_id [lindex $link 1]
        set bookmark_title [lindex $link 2]
        set one_line [lindex $link 3]

        if { $n_copies > 1 } {
            set n_text "($n_copies)"
        } else {
            set n_text ""
        }

        append page "
  <tr bgcolor=#eeeeee>
    <td>
      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    </td>
    <td align=right>
      <img src=pics/dot.gif border=0>
    </td>
    <td align=left nowrap>
       <a href=view?[export_url_vars bookmark_id return_url]>
       $bookmark_title</a> $n_text
    </td>
    <td align=left width=100% nowrap>
      <font size=-1>
        $one_line
      </font>
    </td>
  </tr>"
    }
}

append page "
</table>
<p>
View public bookmarks <a href=group-view>grouped by host or user</a>.
<p>
<form action=search>
  <table cellpadding=0 cellspacing=0 border=0>
    <tr>
      <td>
        Search for:
        <input size=30 name=srch_text>
        <input type=submit value=\"Search\">
      </td>
    </tr>
  </table>
</form>
```

```
[wbm_footer]"

ns_return 200 text/html $page
```

Group Summary View

```
# /www/wap/bookmark/group-view.tcl
# By sirkin@mit.edu on 07/20/00

# Groups public bookmarks by host_url and owner_id and lists the hosts and
# owner names with the number of bookmarks in each group. Users then follow
# links in each list to view all of the bookmarks in that group.

ad_maybe_redirect_for_registration

ad_page_variables {
    {hosts_page {0}}
    {users_page {0}}
}

set title "Public Bookmarks"

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] "Public Bookmarks"]
<hr>
<h3>View Bookmarks</h3>"

set hosts_list [db_list_of_lists "
    select   count (*) as n_bookmarks,
             host_url
    from     wap_bookmarks
    where    private_p != 't'
    group by host_url
    order by n_bookmarks desc"]

set n_total_hosts [llength $hosts_list]

# Provide scroll options so not all bookmarks have to be listed on one page.

set n_hosts_view 10
set min_host [expr $n_hosts_view * $hosts_page + 1]
if { $n_total_hosts > [expr $min_host + $n_hosts_view - 1] } {
    set max_host [expr $min_host + $n_hosts_view - 1]
} else {
    set max_host $n_total_hosts
}

set num_hosts_text "$min_host - $max_host of $n_total_hosts"

set prev_link "<a href=group-view?hosts_page=[expr $hosts_page - 1]&
users_page=$users_page><img src=pics/lf.gif border=0></a>"
```

```
set more_link "<a href=group-view?hosts_page=[expr $hosts_page + 1]&
users_page=$users_page><img src=pics/rt.gif border=0></a>"

set num_hosts_menu "&nbsp;"
if { $min_host > $n_hosts_view } {
    append num_hosts_menu "$prev_link&nbsp;"
}
append num_hosts_menu $num_hosts_text
if { $max_host < $n_total_hosts } {
    append num_hosts_menu "&nbsp;$more_link"
}

for {set i $min_host} {$i <=$max_host} {incr i} {
    set host [lindex $hosts_list [expr $i - 1]]

    set n_bookmarks [lindex $host 0]
    set host_url [lindex $host 1]

    regsub {^http://([^/]*)/?} $host_url {\1} host_name
    append host_list "
          <li>
            <a href=host-view?[export_url_vars host_url]>
            $host_name</a> ($n_bookmarks)
          </li>"
}

set users_list [db_list_of_lists "
    select   count (bookmark_id) as n_bookmarks,
             first_names || ' ' || last_name as viewed_user_name,
             owner_id as viewed_user_id
    from     users, wap_bookmarks
    where    owner_id = user_id
    and      private_p = 'f'
    group by first_names, last_name, owner_id
    order by n_bookmarks desc"]

set n_total_users [llength $users_list]

# Provide scroll options so not all bookmarks have to be listed on one page.

set n_users_view 10
set min_user [expr $n_hosts_view * $users_page + 1]
if { $n_total_users > [expr $min_user + $n_users_view - 1] } {
    set max_user [expr $min_user + $n_users_view - 1]
} else {
    set max_user $n_total_users
}

set num_users_text "$min_user - $max_user of $n_total_users"

set prev_link "<a href=group-view?hosts_page=$hosts_page&
users_page=[expr $users_page - 1]><img src=pics/lf.gif border=0></a>"
set more_link "<a href=group-view?hosts_page=$hosts_page&
users_page=[expr $users_page + 1]><img src=pics/rt.gif border=0></a>"
```

```
set num_users_menu "&nbsp;"
if { $min_user > $n_users_view } {
    append num_users_menu "$prev_link&nbsp;"
}
append num_users_menu $num_users_text
if { $max_user < $n_total_users } {
    append num_users_menu "&nbsp;$more_link"
}

for {set i $min_user} {$i <=$max_user} {incr i} {
    set user [lindex $users_list [expr $i - 1]]

    set n_bookmarks [lindex $user 0]
    set viewed_user_name [lindex $user 1]
    set viewed_user_id [lindex $user 2]

    append user_list "
          <li>
            <a href=user-view?[export_url_vars viewed_user_id]>
            $viewed_user_name</a> ($n_bookmarks)
          </li>"
}

if { $n_total_hosts == 0 || $n_total_users == 0 } {
    append page "
There are no public bookmarks to view at this time.
} else {
    append page "
<blockquote>
  <table cellpadding=2 cellspacing=0 border=0>
    <tr>
      <td align=left width=100%>
        Grouped by host
      </td>
      <td align=right nowrap>
        $num_hosts_menu
      </td>
    </tr>
    <tr>
      <td colspan=2>
        <ul>$host_list
        </ul>
      </td>
    </tr>
    <tr>
      <td colspan=2>
        &nbsp;
      </td>
    </tr>
    <tr>
      <td align=left width=100%>
        Grouped by user
      </td>
      <td align=right nowrap>
```

```
        $num_users_menu
      </td>
    </tr>
    <tr>
      <td colspan=2>
        <ul>$user_list
        </ul>
      </td>
    </tr>
  </table>
</blockquote>"
}

append page "
View public bookmarks <a href=public-view>listed by name</a>.
[wbm_footer]"

ns_return 200 text/html $page
```

Group Summary View by Host

```
# /www/wap/bookmark/host-view.tcl
# By sirkin@mit.edu on 07/20/00

# One of two target pages from group-view, which only shows bookmark groups,
# not individual bookmarks. This script lists bookmarks grouped by host_url.

ad_maybe_redirect_for_registration

ad_page_variables {
    host_url
}

if { ![regsub {^http://([^/]*)/?} $host_url {\1} viewed_host] } {
    set viewed_host ""
}

set title "Bookmarks for $viewed_host"

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "group-view" "Public
Bookmarks"] "One Host"]
<hr>
<table cellpadding=2 cellspacing=0 border=0 width=100%>
  <tr bgcolor=#dddddd>
    <td align=left>
      <img src=pics/folderopen.gif border=0>
    </td>
    <td align=left width=100% nowrap>
      <b>Public WAP Bookmarks</b>
    </td>
  </tr>
</table>
<table cellpadding=2 cellspacing=0 border=0 width=100%>"

set db_query "
    select   first_names || ' ' || last_name as owner_name,
             owner_id,
             bookmark_id,
             bookmark_title,
             nvl (one_line_descrip, '&nbsp;') as one_line
    from     users, wap_bookmarks
    where    user_id = owner_id
    and      host_url = '$QQhost_url'
    and      private_p = 'f'
    order by owner_name, sort_key"

set current_name ""
set return_url [ad_conn url]
db_foreach $db_query {
    if { $current_name != $owner_name } {
set current_name $owner_name
append page "
  <tr bgcolor=#eeeeee>
    <td>
      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    </td>
    <td colspan=3>
      <a href=/shared/community-member?user_id=$owner_id>$owner_name</a>
    </td>
  </tr>"
    }

    append page "
  <tr bgcolor=#eeeeee>
    <td>
      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    </td>
    <td align=right>
      <img src=pics/dot.gif border=0>
    </td>
    <td align=left nowrap>
      <a href=view?[export_url_vars bookmark_id]>$bookmark_title</a>
    </td>
    <td align=left width=100% nowrap>
      <font size=-1>
        $one_line
      </font>
    </td>
  </tr>"

} if_no_rows {
    append page "
  $viewed_host has no public bookmarks stored in the database."
}
```

```
append page "
</table>
[wbm_footer]"

ns_return 200 text/html $page
```

Group Summary View by User

```
# /www/wap/bookmark/user-view.tcl
# By sirkin@mit.edu on 07/20/00

# One of two target pages from group-view, which only shows bookmark groups,
# not individual bookmarks. This script lists bookmarks grouped by owner_id.

ad_maybe_redirect_for_registration

ad_page_variables {
    viewed_user_id
}

set viewed_name [db_string "
    select   first_names || ' ' || last_name
    from     users
    where    user_id = $viewed_user_id"]

set title "Bookmarks for $viewed_name"

set page "
[wbm_header $title]
<h2>$title</h2>
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "group-view" "Public
Bookmarks"] "One User"]
<hr>
Community member page for <a href=/shared/community-member?
user_id=$viewed_user_id>$viewed_name</a>
<p>
<table cellpadding=2 cellspacing=0 border=0 width=100%>
  <tr bgcolor=#dddddd>
    <td align=left>
      <img src=pics/folderopen.gif border=0>
    </td>
    <td align=left width=100% nowrap>
      <b>Public WAP Bookmarks</b>
    </td>
  </tr>
</table>
<table cellpadding=2 cellspacing=0 border=0 width=100%>"

set db_query "
    select   bookmark_id,
             bookmark_title,
             nvl (one_line_descrip, '&nbsp;') as one_line
    from     wap_bookmarks
    where    owner_id = $viewed_user_id
    and      private_p = 'f'
    order by sort_key"

set return_url [ad_conn url]
db_foreach $db_query {
    append page "
  <tr bgcolor=#eeeeee>
    <td>
      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    </td>
    <td align=right>
      <img src=pics/dot.gif border=0>
    </td>
    <td align=left nowrap>
      <a href=view?[export_url_vars bookmark_id return_url]>
      $bookmark_title</a>
    </td>
    <td align=left width=100% nowrap>
      <font size=-1>
        $one_line
      </font>
    </td>
  </tr>"

} if_no_rows {
    append page "
  $viewed_name has no public bookmarks stored in the database."
}

append page "
</table>
[wbm_footer]"

ns_return 200 text/html $page
```

Search for Bookmark

```
# /www/wap/bookmark/search.tcl
# By sirkin@mit.edu on 08/20/00

# Search through all public bookmarks for keywords within their titles, host or
# complete urls, one-line or detailed descriptions, and meta-tag descriptions.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    srch_text
    {num_page {0}}
}

set srch_text [string trim $srch_text]
set srch_ptrn "'%[string toupper $QQsrch_text]%'"

set links_list [db_list_of_lists "
    select   max (bookmark_id) as bookmark_id,
             max (bookmark_title) as bookmark_title,
             nvl (max (one_line_descrip), '&nbsp;') as one_line
    from     wap_bookmarks
    where    owner_id != $user_id
    and      private_p = 'f'
    and      (upper (bookmark_title) like $srch_ptrn
    or        upper (complete_url) like $srch_ptrn
    or        upper (one_line_descrip) like $srch_ptrn
    or        upper (detailed_descrip) like $srch_ptrn
    or        upper (meta_keywords) like $srch_ptrn
    or        upper (meta_descrip) like $srch_ptrn)
    group by complete_url, the_regexp
    order by bookmark_title"]

set n_total_links [llength $links_list]

# Provide scroll options so not all bookmarks have to be listed on one page.

set n_view 10
set min_link [expr $n_view * $num_page + 1]
if { $n_total_links > [expr $min_link + $n_view - 1] } {
    set max_link [expr $min_link + $n_view - 1]
} else {
    set max_link $n_total_links
}

set num_text "$min_link - $max_link of $n_total_links"

set prev_link "<a href=search?srch_text=$srch_text&
num_page=[expr $num_page - 1]><img src=pics/lf.gif border=0></a>"
set more_link "<a href=search?srch_text=$srch_text&
num_page=[expr $num_page + 1]><img src=pics/rt.gif border=0></a>"

set num_menu "&nbsp;"
if { $n_total_links > 0 } {
    if { $min_link > $n_view } {
        append num_menu "$prev_link&nbsp;"
    }
    append num_menu $num_text
    if { $max_link < $n_total_links } {
        append num_menu "&nbsp;$more_link"
    }
}

set title "Search for \"$srch_text\""

set page "
[wbm_header $title]
<h2>$title</h2>
```

```
[ad_context_bar_ws [list "" "WAP Bookmark Home"] [list "public-view" "Public
Bookmarks"] "Search"]
<hr>
<p>
<table cellpadding=2 cellspacing=0 border=0 width=100%>
  <tr bgcolor=#dddddd>
    <td align=left>
      <img src=pics/folderopen.gif border=0>
    </td>
    <td align=left width=100% nowrap>
      <b>Matching Public WAP Bookmarks</b>
    </td>
    <td align=right nowrap>
      $num_menu
    </td>
  </tr>
</table>
<table cellpadding=2 cellspacing=0 border=0 width=100%>"

if { $n_total_links == 0 } {
    append page "
  There are no public bookmarks that matched your search term."
} else {
    set return_url [ad_conn url]
    for {set i $min_link} {$i <= $max_link} {incr i} {
        set link [lindex $links_list [expr $i - 1]]

        set bookmark_id [lindex $link 0]
        set bookmark_title [lindex $link 1]
        set one_line [lindex $link 2]

append page "
  <tr bgcolor=#eeeeee>
    <td>
      &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
    </td>
    <td align=right>
      <img src=pics/dot.gif border=0>
    </td>
    <td align=left nowrap>
       <a href=view?[export_url_vars bookmark_id return_url]>
       $bookmark_title</a>
    </td>
    <td align=left width=100% nowrap>
      <font size=-1>
        $one_line
      </font>
    </td>
  </tr>"
    }
}

append page "
</table>
```

```
[wbm_footer]"

ns_return 200 text/html $page
```

Copy Public Bookmark

```
# /www/wap/bookmark/copy.tcl
# By sirkin@mit.edu on 07/18/00

# Inserts a new bookmark in the table with the same attributes as the bookmark
# pointed to by bookmark_id, except for new_bookmark_id and copying user_id.

ad_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
    new_bookmark_id
}

set sql_query "
    select *
    from   wap_bookmarks
    where  bookmark_id = $bookmark_id"

if { [catch {db_1row $sql_query} errmsg] } {
    ad_return_error "Database Select Failed" "There was an error
    making this select from the database. The error message was:
    <p><blockquote><pre>$errmsg</pre></blockquote>"
    return
}

set insert "
    insert into wap_bookmarks
        (bookmark_id,
         bookmark_title,
         host_url,
         complete_url,
         the_regexp,
         match_vars,
         multiple_p,
         private_p,
         preceding_text,
         following_text,
         owner_id,
         sort_key,
         one_line_descrip,
         detailed_descrip,
         creation_date)
    values
        ($new_bookmark_id,
         '[DoubleApos $bookmark_title]',
         '[DoubleApos $host_url]',
        '[DoubleApos $complete_url]',
        '[DoubleApos $the_regexp]',
        '[DoubleApos $match_vars]',
        '[DoubleApos $multiple_p]',
        '[DoubleApos $private_p]',
        '[DoubleApos $preceding_text]',
        '[DoubleApos $following_text]',
        $user_id,
        $new_bookmark_id,
        '[DoubleApos $one_line_descrip]',
        '[DoubleApos $detailed_descrip]',
        sysdate)"

# Protect against a double-click on the Web form inserting two copies
# into the database. The new_bookmark_id value was generated in view,

if { [catch {db_dml $insert} errmsg] } {
    if { [db_string "select count (*) from wap_bookmarks
         where bookmark_id = $new_bookmark_id"] == 0 } {

        ad_return_error "Database Insert Failed" "There was an error
        making this insert into the database. The error message was:
        <p><blockquote><pre>$errmsg</pre></blockquote>"
        return
    }
}

ad_returnredirect ""
```

C: Serve Content to WAP Browsers

Only two scripts are needed for the WAP portion of the site. `wap-home` displays the personal bookmark menu summary view for mobile devices. `serve` actually runs the regular expression against the target page and returns the matches.

Personal Menu Summary View

```
# /www/wap/bookmark/wap-home.tcl
# By sirkin@mit.edu on 07/06/00

# Front page viewed from WAP browser. Presents a select list of bookmarks the
# user has indicated as currently visible to the device. If more than ten are
# visible, the last option is a More..., directing the user to a new card.

wap_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

set deck "
<?xml version=\"1.0\"?>
<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"
\"http://www.wapforum.org/DTD/wml_1.1.xml\">

<wml>
  <head>
    <meta forua=\"true\" http-equiv=\"Cache-Control\"
    content=\"max-age=0\"/>
  </head>

  <card title=\"ACS WAP\" id=\"c_1\">
    <p>"

# Where value in the_regexp is null, the bookmark references a native WAP site.

set links_list [db_list_of_lists "
    select   bookmark_id,
             bookmark_title,
             complete_url,
             decode (the_regexp, null, 't', 'f') as wap_p
    from     wap_bookmarks
    where    owner_id = $user_id
    and      hidden_p = 'f'
    order by sort_key"]

set n_links 0
set n_cards 1
set n_all_links [llength $links_list]

set deck_footer "
    </p>
  </card>
```

```
</wml>"

if { $n_all_links == 0 } {
    append deck "
      No stored bookmarks.
$deck_footer"
    ns_return 200 text/vnd.wap.wml $deck
    return
} else {
    append deck "
      Select Bookmark:
      <select>"
}

foreach link $links_list {
    incr n_links
    set bookmark_id [lindex $link 0]
    set bookmark_title [lindex $link 1]
    set complete_url [lindex $link 2]
    set wap_p [lindex $link 3]

    # If there are 10 links, present them inline. Otherwise require a More...
    # link in the 10th slot of each card. That is, when n_links = 10, 19, 28,
    # 37, etc. Find this as (n_links - 1) == (9 * n_cards) (in real numbers).

    if { [expr $n_links - 1.0] == [expr 9.0 * $n_cards] } {
        set more_slot_p "t"
    } else {
        set more_slot_p "f"
    }

    if { $more_slot_p && $n_all_links > $n_links} {
        incr n_cards
        append deck "
        <option onpick=\"#c_$n_cards\">More...</option>
      </select>
    </p>
  </card>

  <card title=\"ACS WAP\" id=\"c_$n_cards\">
    <p>
      Select Bookmark:
      <select>"
    }

    # WML treats $text as a variable, so if a text string includes a $, the
    # page will fail to load. $$ escapes the substitution to print out a $.

    regsub -all {\$} $bookmark_title "$$" bookmark_title

    # If bookmark is a WAP site, link directly to it. Otherwise use serve.

    if { $wap_p } {
        append deck "
        <option onpick=\"$complete_url\">$bookmark_title</option>"
    } else {
        append deck "
        <option onpick=\"serve?bookmark_id=$bookmark_id\">$bookmark_title</
option>"
}

append deck "
      </select>
$deck_footer"

ns_return 200 text/vnd.wap.wml $deck
```

Serve Bookmark

```
# /www/wap/bookmark/serve.tcl
# By sirkin@mit.edu on 07/06/00

# Target page viewed from WAP browser. Presents the bookmark selected by the
# user in wap-home. Rather straightforward, it matches the regexp against the
# target page once or multiple times, removing the match each time. The match
# variables indicated by the user are then placed into a list, wrapped with a
# WML template, and served.

wap_maybe_redirect_for_registration
set user_id [ad_verify_and_get_user_id]

ad_page_variables {
    bookmark_id
}

# Use meta value must-revalidate rather than max-age=0 so the deck will be
# forced to reload when navigating to it backward (i.e., during a refresh).

set deck "
<?xml version=\"1.0\"?>
<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"
\"http://www.wapforum.org/DTD/wml_1.1.xml\">

<wml>
  <head>
    <meta forua=\"true\" http-equiv=\"Cache-Control\"
    content=\"must-revalidate\"/>
  </head>

  <template>
    <do type=\"accept\" label=\"Back\">
      <prev/>
    </do>
    <do type=\"accept\" label=\"Reload\">
      <go href=\"serve?bookmark_id=$bookmark_id\"/>
    </do>
  </template>
  <card title=\"ACS WAP\">
    <p>"

set db_query "
    select  bookmark_title,
            complete_url,
            the_regexp,
            match_vars,
            multiple_p,
            preceding_text,
            following_text
    from    wap_bookmarks
    where   bookmark_id = $bookmark_id
    and     owner_id = $user_id"

set deck_footer "
    </p>
  </card>
</wml>"

if { [catch {db_1row $db_query} errmsg] } {
    append deck "
      Not your bookmark.
$deck_footer"
    ns_return 200 text/vnd.wap.wml $deck
    return
}

if { [catch {ns_httpget $complete_url 10} url_text] } {
    append deck "
      Unable to reach host.
$deck_footer"
    ns_return 200 text/vnd.wap.wml $deck
    return
}

# WML treats $text as a variable, so if a text string includes a $, the
# page will fail to load. $$ escapes the substitution to print out a $.

regsub -all {\$} $bookmark_title "$$" bookmark_title

append deck "
      $bookmark_title
      <br/>"

if { ![eval $the_regexp] } {
    append deck "
      Unable to apply bookmark."
} else {
    if { ![empty_string_p $preceding_text] } {
        regsub -all {\$} $preceding_text "$$" preceding_text
        append deck "
      $preceding_text
      <br/>"
    }

    regexp {regexp .*{(.*)} \$url_text} $the_regexp match pattern

    # Patterns may begin with -text, which would cause the regexp or regsub
    # match to fail. In that case, use -- to indicate the end of switches.

    if { [string first "-" $pattern] == 0 } {
        set sw "-- "
    } else {
        set sw ""
    }

    set n_match 1
    set n_max_match 20
    set mult_regsub "regsub $sw{$pattern} \$url_text {\1} url_text"

    # Execute the following code once if not multiple_p. Otherwise, loop until
    # we find all matches in the url_text or reach a max number of iterations.

    while { [eval $mult_regsub] && $multiple_p == "t" &&
            $n_match <= $n_max_match || $n_match == 1 } {

        foreach var $match_vars {

            # Here we want to regsub on each match_var, named var_1 etc.
            # Note that $var references var_1, which points to the text.

            regsub -all {\$} [set $var] "$$" var_1
            append deck "
      o [ns_striphtml $var_1]
      <br/>"
        }
        incr n_match
        eval $the_regexp
    }

    if { ![empty_string_p $following_text] } {
        regsub -all {\$} $following_text "$$" following_text
        append deck "
      $following_text"
    }
}

append deck "
$deck_footer"

ns_return 200 text/vnd.wap.wml $deck
```

[illegible]