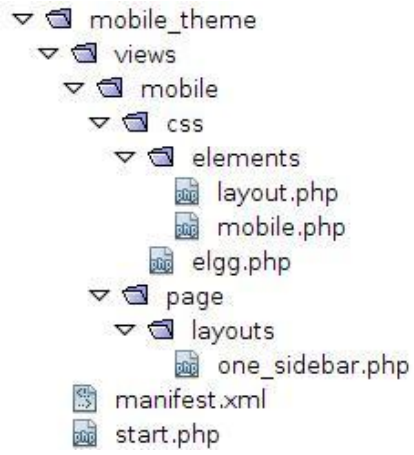
Creating a mobile theme

We close the chapter with a short guide to creating a mobile theme. Elgg's view system makes it easy to create a theme for mobile devices. A mobile theme creates another viewtype that is used whenever a mobile browser is detected. In Chapter 7, we discussed how Elgg uses a default viewtype to serve web pages to a browser and an rss viewtype to serve RSS feeds to a feed aggregator. The logic and data are the same for different viewtypes, but the rendered output is different. For a mobile theme, we create a mobile viewtype and switch to this viewtype whenever a mobile browser is detected.

Plugin structure

Our mobile theme is structured like the preceding theme except that instead of putting the views in `views/default/`, they are added to `views/mobile/`, as follows:



In the start script, we detect mobile browsers by analyzing the user agent of the request. There are toolkits available to do this for you such as**WURFL** (http://wurfl.sourceforge.net/nphp/) and http://detectmobilebrowser.com/. We leave it up to you to select one, or code the plugin to always serve the mobile theme. The code in the start script looks like the following code snippet:

```
<?php
/**
* Mobile theme
*
* An example of detecting mobile browsers and changing the viewtype
*/
elgg_register_event_handler('init', 'system', 'mobile_theme_init');
function mobile_theme_init() {
elgg_register_viewtype_fallback('mobile');
mobile_theme_set_viewtype();
// do not want the more menu
elgg_unregister_plugin_hook_handler('prepare', 'menu:site', 'elgg_site_menu_setup');
}
function mobile_theme_set_viewtype() {
$user_agent = $_SERVER['HTTP_USER_AGENT'];
// your code here
elgg_set_viewtype('mobile');
}
```

In the initialization function, we register our mobile viewtype as one that falls back to default. Doing this causes Elgg to use a view from the default viewtype if one does not exist in the mobile viewtype. That way we do not need to create a mobile view for every single

default view, but only for the ones that need to be structured differently for mobile browsers.

The `mobile_theme_set_viewtype()` function checks the mobile devices based on the user agent string sent by the browser. If it detects a mobile device, then it switches Elgg into the mobile viewtype. Of course, in the preceding code it is always set to the mobile viewtype.

Layout

The only change we are making to the layout is removing the sidebar. We do this by adding a `page/layouts/one_sidebar` view in the mobile viewtype. After we copy the code from the original view and remove the section for the sidebar, we now have a single column site.

CSS

It is best that a mobile theme have a fluid layout to accommodate smaller screen resolutions. We convert the default theme to a fluid theme by overriding the layout CSS view. This is accomplished by removing the fixed widths on `divs`. When we are done, the layout CSS is pared down to the following:

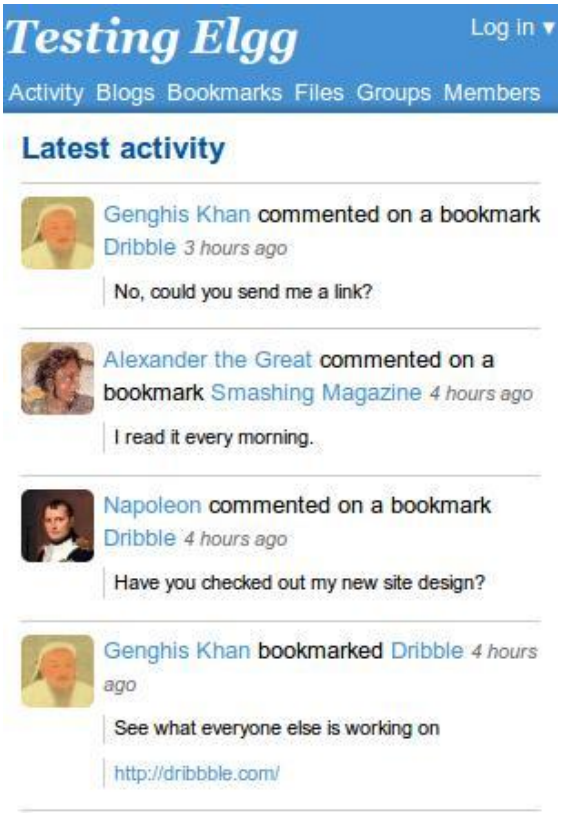
```
/***** TOPBAR ******/
.elgg-page-topbar {
background: #333333 url(<?php echo elgg_get_site_url(); ?>_graphics/toptoolbar_background.gif) repeat-x top left
border-bottom: 1px solid #000000;
position: relative;
height: 24px;
z-index: 9000;
}
/***** PAGE MESSAGES ******/
.elgg-system-messages {
position: fixed;
top: 24px;
right: 20px;
z-index: 2000;
}
.elgg-system-messages li {
margin-top: 10px;
}
.elgg-system-messages li p {
margin: 0;
}
/***** PAGE HEADER ******/
.elgg-page-header {
position: relative;
background: #4690D6 url(<?php echo elgg_get_site_url();
?>_graphics/header_shadow.png) repeat-x bottom left;
}
.elgg-page-header > .elgg-inner {
position: relative;
height: 60px;
}
/***** PAGE BODY LAYOUT ******/
.elgg-layout {
min-height: 360px;
}
.elgg-main {
position: relative;
padding: 10px;
}
.elgg-main > .elgg-head {
padding-bottom: 3px;
border-bottom: 1px solid #CCCCCC;
```

```
margin-bottom: 10px;
}
/***** PAGE FOOTER ******/
.elgg-page-footer {
position: relative;
}
.elgg-page-footer {
color: #999;
}
.elgg-page-footer a:hover {
color: #666;
}
```

We also override the primary CSS view, copy the original code into it, and add a line near the bottom to include a new mobile CSS view.

```
echo elgg_view('css/elements/mobile', $vars);
```

This mobile CSS view is used to override specific elements of the default theme without overriding complete views. We decrease the font sizes and simplify the site menu. When we are done, we have a site that looks like the following screenshot:



Because much of Elgg's CSS uses fluid layouts, there does not need to be much restyling of the content. Most of the work in creating a mobile theme is with navigation. For example, we did not theme the topbar. It takes up too much space without changes for a mobile display.