Web APIs

Browsing the Web

- Can launch the built-in browser from an app
  - switches to full-featured browser
  - user returns to app via back button
  - no special coding to handle web browsing
- Use a web browser control
  - WebView
  - WebKit HTML rendering engine
  - easily add rich content to app
    - from web
    - locally generated
    - help pages

Uri uri1Uri = Uri.parse("http://androidbook.blogspot.com/");
Intent launchBrowser = new Intent(Intent.ACTION_VIEW, uri1Uri);
startActivity(launchBrowser);
WebView

- android.webkit.WebView
- requires INTERNET permission
- is a UI control
  - include in layout resource
  - needs reasonable amount of room
- content
  - from specific website
  - from server
  - from local storage
  - generated HTML
- scaling for mobile device
  - setInitialScale

```
final WebView wr = (WebView) findViewById(R.id.webview);
wr.loadUrl("http://www.periprol.org");

wr.loadUrl("file:///android_asset/webxy.html");

String strPageTitle = "The Last Words of Oscar Wilde";
String strPageContent = "<h1>" + strPageTitle + "</h1>
&quo; + strPageContent + "</body/>

wr.loadDataWithBaseURL("","", "text/html", "utf-8");
```

- WebView state
  - should be handled
  - onPause
    - stops unnecessary processing activity (e.g. JavaScript, plug-ins)
    - should be called at least when Activity pauses
  - onResume
    - to restart
- Flash
  - Flash 11 supported if Flash Player for Android installed from Android Market
WebSettings

- Default WebView lacks features of full browser
  - following link launches built-in browser
  - no title or controls
  - no JavaScript
- `getSettings`
  - WebView method
  - returns WebSettings object
  - used to change settings
  - settings e.g.
    - `setJavaScriptEnabled`
    - `setSupportZoom`

WebViewClient

- Helper class
  - handle navigation in WebView
  - override callbacks to handle rendering events
    - page loading start/finished
    - form submission
    - resource (URL) about to be loaded
    - handle errors during load
- setWebViewClient
- e.g. set title when page loaded
WebChromeClient

- Adding "browser chrome"
  - Features outside the web content region
- Helper class
  - Override callbacks for browser UI events
    - JavaScript calls
    - Progress updates
    - Title changes
- E.g. handling change in title

```java
WebChromeClient webChrome = new WebChromeClient();

@Override
public void onReceivedTitle(View view, String title) {
    super.onReceivedTitle(view, title);
    setTitle(title);
}
```

E.g. WebViewProject

- Manifest
  - use-permission
- Layout
  - WebView widget
- WebViewClient
  - Handling navigational clicks
- Loading initial address
  - onConsole, onKey
  - single line
- Back button
  - Handle click to implement back
  - onKeyUp (Activity callback)
  - Navigation methods
WebViewFragment

- Fragment with built-in WebView
  - subclass and override methods as required
  - onActivityCreated called when activity is created
- onActivityCreated
  - called when activity is created
  - initialization
  - adjust settings for WebView
  - can attach a WebViewClient or WebChromeClient

Content Providers
Data Sharing

- Content providers
  - mechanism for inter-app data sharing
  - share data by
    - implementing the ContentProvider interface
    - registering as a content provider in manifest
- Built-in content providers
  - android.provider
    - helper classes for using built-in providers
- Content URI
  - <standard_prefix>://<authority><data_path>/id
- Should not access from UI thread
Contacts

- Contact list
- Content provider
  - Contacts
    - legacy content provider
    - ContactsContract
      - since API 5, enhanced in API 14
- Permissions
  - READ_CONTACTS, WRITE_CONTACTS
  - android.provider.Contacts
  - android.provider.ContactsContract
    - column names

---

Table 14.3 Commonly Used ContactsContract Data Column Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContactsContract.CommonStatus</td>
<td>Defines a number of frequently used contact columns such as name, number, phone, and photo.</td>
</tr>
<tr>
<td>ContactsContract.Contacts</td>
<td>Defines the raw data associated with a single contact.</td>
</tr>
<tr>
<td>ContactsContract.Data</td>
<td>Defines the raw data associated with a single contact.</td>
</tr>
<tr>
<td>ContactsContract.HomeLookup</td>
<td>Defines the home phone number and can be used to quickly look up a phone number for caller identification purposes.</td>
</tr>
<tr>
<td>ContactsContract.StatusUpdate</td>
<td>Defines the social networking names and can be used to check the instant messaging status of a contact.</td>
</tr>
</tbody>
</table>

---

E.g. ContactsAccessProject

- Present a list of contacts from the Contacts content provider
- URI
  - no id is entire collection
- Column names
- CursorLoader
  - Helper class
    - queries a content provider receiving a cursor over result set
  - loadInBackground
    - load of cursor on background thread
- Build list of contacts as strings
  - iterate through cursor
- Create ArrayAdapter on list
- Set adapter for ListView
Modifying Content

- Permissions
  - may need a WRITE permission
  - can add/update and delete
- Use ContentResolver
  - from Context
  - insert
  - update
  - delete
- e.g.

```java
ContentValues values = new ContentValues();
//values.put(Contacts.People.NAME, "Eugene Swan");
Uri uri = getContentResolver().insert(
    Contacts.People.CONTENT_URI, values);
Uri接触 = Contacts.getContactDetails().content;
Contacts.People update (NAME = "Eugene Swan");
//values.clear();
values.put(Contacts.Homes.HOME_ADDRESS, "2101101212121212");
values.put(Contacts.Homes.TEL1, Contacts.Homes.TEL2);
getContentResolver().insert(phone304, values);
values.clear();
values.put(Contacts.Homes.HOME_ADDRESS, "3101101112121212");
values.put(Contacts.Homes.TEL1, Contacts.Homes.TEL2);
getContentResolver().insert(phone304, values);
```