Cloud Storage for App Developers

Steve Marx (@smarx)
Developer Advocate, Dropbox

Storage for apps

- All apps have storage
- Local storage lacks cross-platform, cross-device access
- Cloud storage has challenges:
  - Identity / authentication
  - Billing
  - Abuse

Solutions

- Use a proper identity system
- Establish a billing relationship
- Create and enforce policies
- Invent a conflict resolution scheme

Use Dropbox (or Box or Drive or SkyDrive or …iCloud?)

- Authentication
  - Email verification, password reset
  - Multi-factor authentication
- Billing
  - Billing helps avoid space abuse
  - Users pay for space, not you
- Existing policies and enforcement
  - Malware distribution
  - Illegal content

Dropbox platform

- Files
  - Drop-ins
  - Core API
  - Sync API
- Datastores
  - Datastore API

Challenges one at a time

- Existing content
- Authentication / identity
- Caching / offline
- Conflict resolution

Challenges one at a time

- **Existing content**
- Authentication / identity
- Caching / offline
- Conflict resolution

Drop-ins: Chooser

- “Opens” a file
- Preview and direct links
- JS, iOS, Android
- No app authentication

DEMO: Inserting an image

```
Dropbox.choose({
  success: function (files) {
    var file = files[0];
    var text = file.name.substring(0,
      file.name.lastIndexOf('.'));
    var link = file.link.replace('www.dropbox.com',
      'dl.dropboxusercontent.com');
    $('#markdown').insertAtCaret('![' + name + ']' +
      '(' + link + ')');
  },
  extensions: ['.jpg', '.png', '.jpeg', '.gif']
});
```

Drop-ins: Saver

- Saves a file
- Pull model
- Web-only (for now)
- No app authentication

DEMO: Saver

```
<a href=".../koala_transparent.png"
class="dropbox-saver"></a>
```

Challenges one at a time

- Existing content
- **Authentication / identity**
- Caching / offline
- Conflict resolution

Core API

- Simple web API
- HTTP (RESTish), JSON, OAuth 2.0
- Libraries in Python, Ruby, Java, PHP, iOS, and Android
- Community-supported libraries in ActionScript, C#, JavaScript, Perl, …

HTTP, JSON, OAuth

```
GET https://api.dropbox.com/1/metadata/auto/hello.txt
Authorization: Bearer ZWp2fW...

HTTP/1.1 200 OK
Content-Type: application/json
{
    "path": "/hello.txt",
    "bytes": 11,
    "modified": "Thu, 20 Jun 2013 00:06:49 +0000",
    "rev": "2118670b5",
    "thumb_exists": false,
    ...
}
```

Permissions

- **App folder**
  permission to a specific folder
- **File types**
  permission to a certain class of files
- **Full Dropbox**
  permission to everything

Authenticating in Python

```
def get_flow():
  return DropboxOAuth2Flow(APP_KEY, APP_SECRET,
    url_for('dropbox_auth', _external=True),
    session, 'dropbox-csrf-token')

@app.route('/')
def index():
  if 'token' not in session:
    return redirect(get_flow().start())
  # Do stuff...

@app.route('/dropbox-auth')
def dropbox_auth():
  token, uid, extras = get_flow().finish(request.args)
  session['token'] = token
  return redirect(url_for('index'))
```

Listing files

```
# Instantiate client with token
client = DropboxClient(token)

# List files
files = client.metadata('/')['contents']
```

Writing a file

```
client.put_file(
  name+'.md', # path
  request.form['markdown'], # contents
  overwrite=True)
```

Sharing a file

```
share_url = client.share(path, short_url=False)['url']
```

Challenges one at a time

- Existing content
- Authentication / identity
- **Caching / offline**
- Conflict resolution

Sync API

- Filesystem view of Dropbox
- Offline caching and syncing
- Change notification
- Core API at the bottom
- iOS and Android

Listing files on Android

```
@Override
public List<DbxFileInfo> loadInBackground() {
  DbxFileSystem fs = getDbxFileSystem();
  ArrayList<DbxFileInfo> entries =
    new ArrayList<DbxFileInfo>();

  for (DbxFileInfo entry : fs.listFolder(mPath)) {
    if (!entry.isFolder
        && entry.path.toString().endsWith(".md")) {
      entries.add(entry);
    }
  }

  return entries;
}
```

Caching in the Sync API

- All reads are from cache
- Changes downloaded for open files
- Notification when new version ready
- `.update()` to use new version

Listening for file changes

```
private final DbxFile.Listener mChangeListener =
  new DbxFile.Listener() {

  @Override
  public void onFileChange(DbxFile file) {
    if (file.getNewerStatus().isCached) {
      contents = mFile.readString();
      // ...
    }
  }
};
```

Challenges one at a time

- Existing content
- Authentication / identity
- Caching / offline
- **Conflict resolution**

Core/Sync conflict resolution

- Core API has three strategies:
  - Overwrite
  - Rename on collision (“myfile (1).txt”)
  - Optimistic concurrency (pass a rev)
- Sync uses optimistic concurrency
- Neither attempts to merge

Datastore API

- Syncing for structured data
- Cache content when offline
- Exchange deltas when online
- Automatically merge changes

Datastores

- Data model hierarchy:
  Account  datastore  table 
  record  field  value
- NoSQL: no relational model
- Everything cached locally
- Change notification
- Call *sync* to send/receive changes
- Per-field conflict resolution

Datastore conflicts

- Each delta sent w/ "parent revision"
- Deltas with bad parent rev: conflicts
- Conflicts are rejected by the server
- Client updates and retries

Datastore conflict resolution

- Changes to different fields "commute"
- Changes to the same field are merged
- Conflict resolution strategies:
  - Server wins (default)
  - Client wins
  - Maximum
  - Minimum
  - Sum (increases and decreases combined)
- Lists are merged using operational transformation (OT)

Datastores on Android

```
@Override
public void onResume() {
  DbxAccount acct = NotesAppConfig
    .getAccountManager(getActivity()).getLinkedAccount();
  datastore = DbxDatastoreManager.forAccount(acct)
    .openDefaultDatastore();
  datastore.addSyncStatusListener(this);
  datastore.sync();
  updateSlide();
}
@Override
public void onPause() {
  datastore.removeSyncStatusListener(this);
  datastore.close();
}
```

Handling changes

```
@Override
public void onDatastoreStatusChange(DbxDatastore datastore) {
  if (datastore.getSyncStatus().hasIncoming) {
    datastore.sync();
    updateSlide();
  }
}

private void updateSlide() {
  ...
  DbxRecord record = datastore.getTable("decks").get(id);
  ...
}
```

Making updates

```
private void incrementSlideBy(int i) {
  ...
  DbxRecord record =
    datastore.getTable("decks").getOrInsert(id);
  if (record.hasField("currentSlide")) {
    record.set("currentSlide",
      (int)record.getDouble("currentSlide") + i);
  } else {
    record.set("currentSlide", 2);
  }
  datastore.sync();
  ...
}
```

Summary

- Storage in the cloud isn't enough
- Identity/authentication is important
- Abuse prevention is important
- Caching, offline access important
- Use a service that's built for this

Thanks!

- https://www.dropbox.com/developers
- Email me: smarx@dropbox.com
- Complain about me: @smarx