
Google™
10

Building context-aware extensions for Gmail

Dan Holevoet
May 20, 2010

Got questions?

We're using Wave and Google Moderator:
http://bit.ly/bcu4jE

Agenda

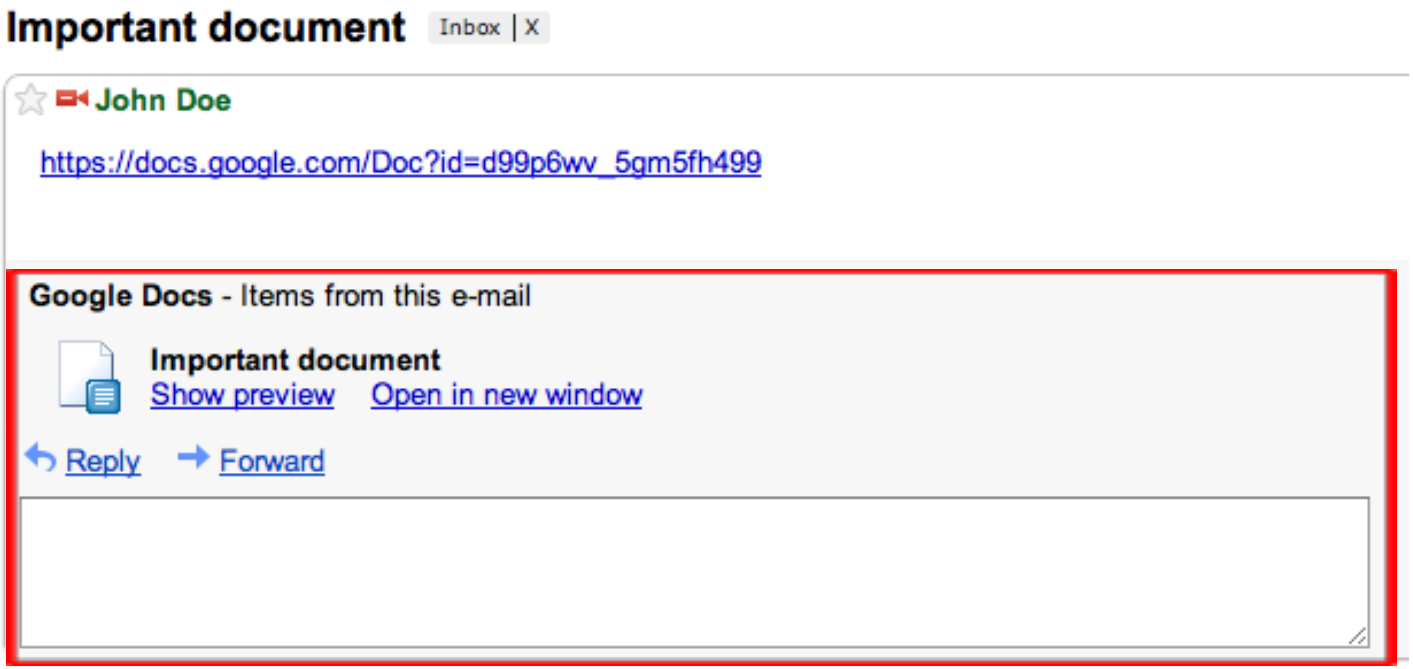
- Why contextual gadgets?
- What makes a contextual gadget?
  - Gadgets
  - Extractors
  - The manifest
- How do I test and distribute contextual gadgets?
- Common tasks and best practices
- An example application
- How do I get started?

Why contextual gadgets?

Why contextual gadgets?

Example gadgets

- YouTube gadget, matches YouTube links and displays the video
- Docs/Spreadsheets gadget, matches Docs and Spreadsheets links and gives an inline preview



Why contextual gadgets?

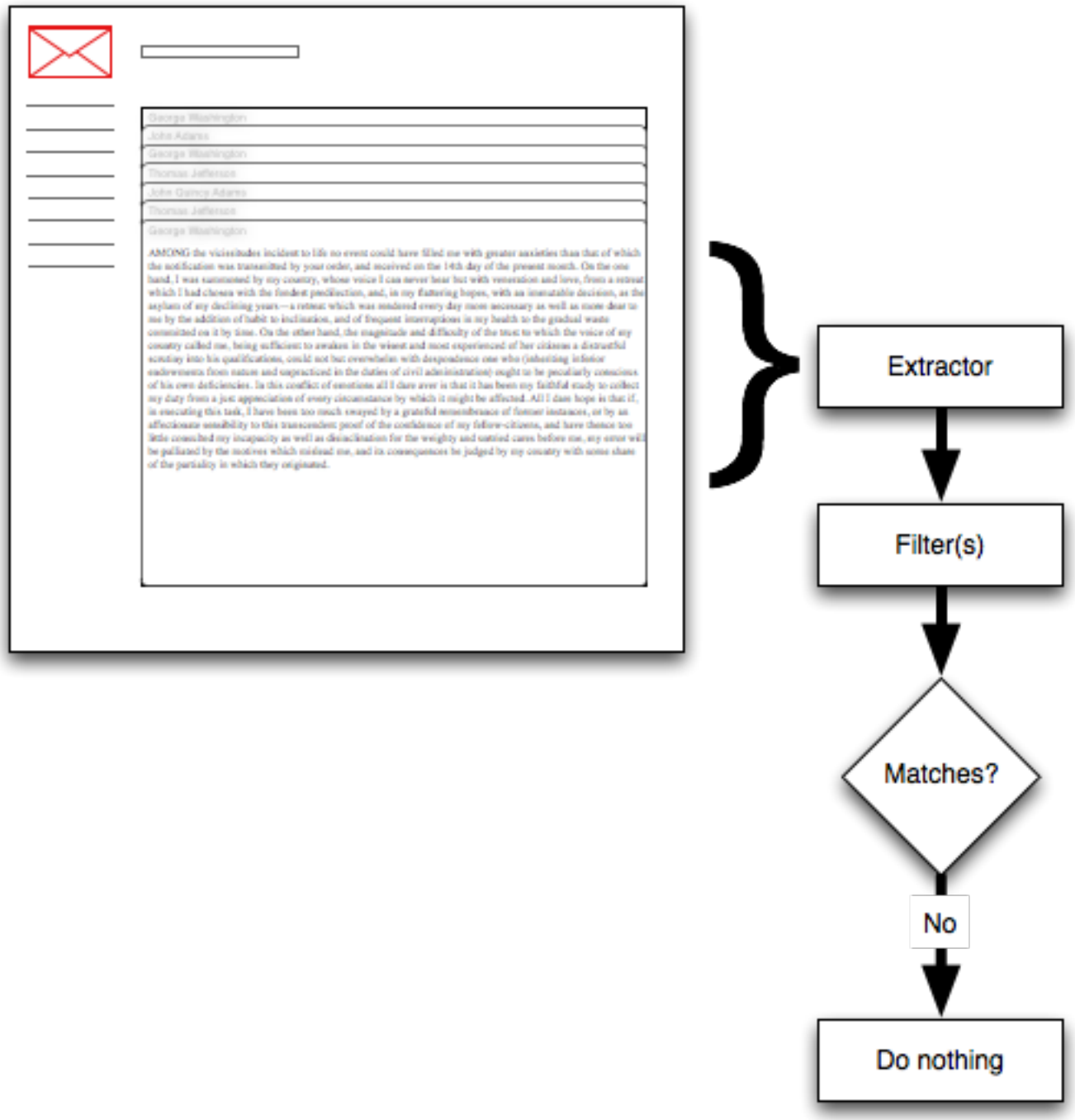
What do contextual gadgets do?

- Match on fields of an incoming email, eg:
  - To
  - From
  - Subject
  - Message body
- Matches trigger a gadget to display, and provide context

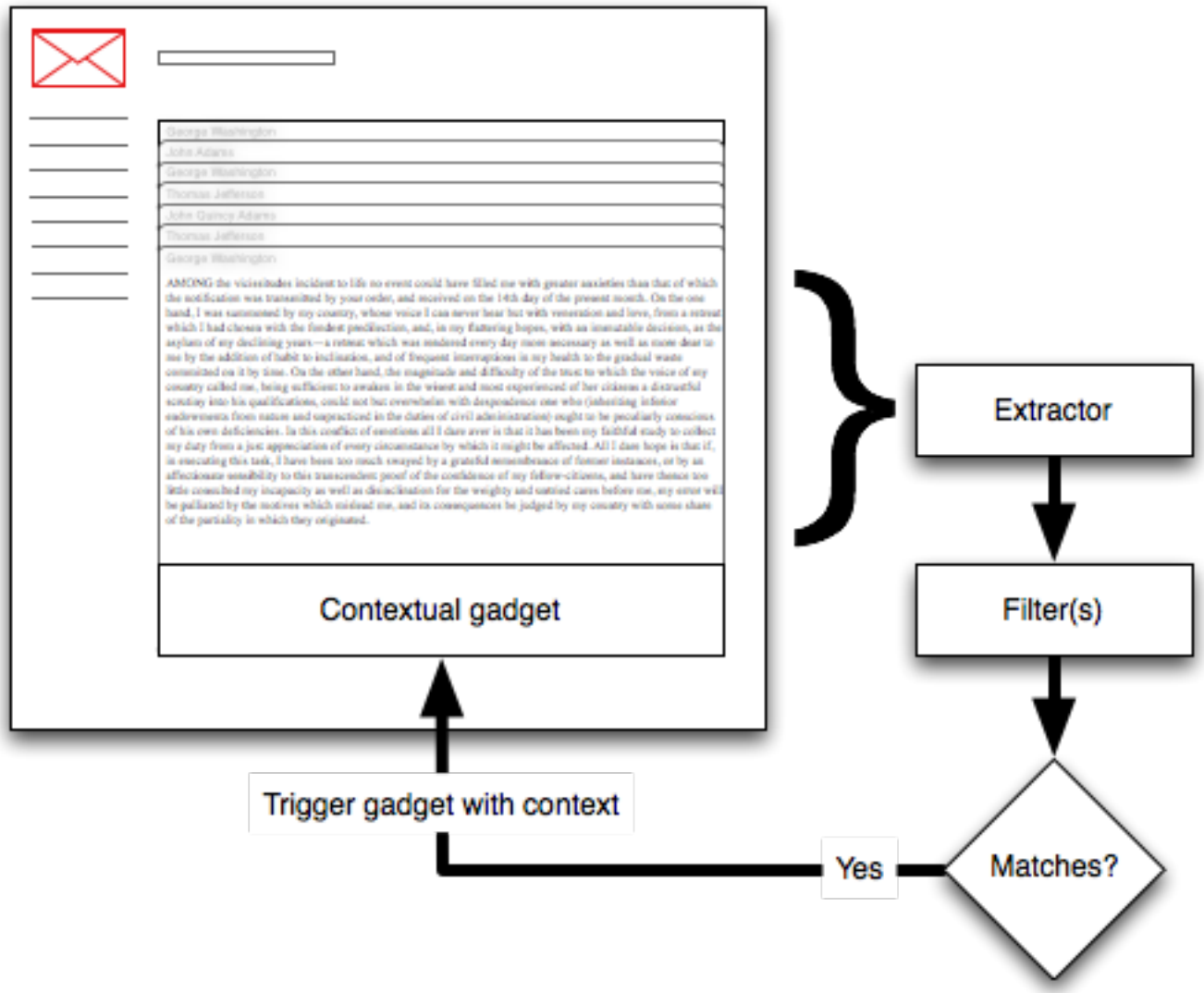
What makes a contextual gadget?

What makes a contextual gadget?

What makes a contextual gadget?



What makes a contextual gadget?



What makes a contextual gadget?

- Each contextual gadget is made up of:
  - A gadget
  - One or more extractors
  - Associated descriptions in the application manifest

What makes a contextual gadget?

Gadgets are XML, HTML, and JavaScript

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs title="hello world example" />
  <Content type="html">
    <![CDATA[
      Hello, world!
    ]]>
  </Content>
</Module>
```

What makes a contextual gadget?

Gadget-specific features

- 'dynamic-height' allows the gadget to grow and shrink depending on content
- osapi.http allows remote data access
  - Provides OAuth signing
  - Supports multiple HTTP methods
  - Supports JSON parsing of responses
- google.contentmatch returns matches from an email message

What makes a contextual gadget?

Extractors

- Extractors are essentially PCREs (Perl-compatible regular expressions)
- Two varieties:
  - Pre-canned (we do the work for you)
  - Custom (coming soon!)

What makes a contextual gadget?

Pre-canned extractors

- Google provides a set of pre-canned extractors that developers can use in their gadgets
- The list of pre-canned extractors:
  - EmailAddressExtractor
  - EmailBodyExtractor
  - EmailTimeExtractor
  - HelloExtractor
  - HttpLinkExtractor
  - RecipientCCEmailExtractor
  - RecipientEmailExtractor
  - RecipientToEmailExtractor
  - SenderEmailExtractor
  - SubjectExtractor
  - USStockTicker

What makes a contextual gadget?

Limiting pre-canned extractors

- Pre-canned extractors may be filtered to apply only on a subset of emails
- An example:
  - SubjectExtractor will match on all emails
  - Filtering allows only a subset of subjects to trigger the gadget
  - Filters are applied by using a regular expression on a field of the extractor

```
<Extension id="SubjectExtractor"
 type="contextExtractor">
 <Url>google.com:SubjectExtractor</Url>
 <Param name="subject"
  value="Hello World.*"/>
 <Triggers ref="MyGadget"/>
 <Container name="mail"/>
</Extension>
```

What makes a contextual gadget?

Custom extractors: combining results

- There are pre-canned extractors for every field of the email
- Custom extractors let you combine results for multiple fields into a single match

What is an extractor?

Example extractor

```
<ExtractorSpec platform="gmail" language="en">
 <Response platform="gmail" format="cardgadget">
   <Output name="time_sent">
    {@__DATE_SENT__}</Output>
   <Output name="sender_email">
    {@__FROM_ADDRESS__}</Output>
 </Response>
</ExtractorSpec>
```

What makes a contextual gadget?

The application manifest

- The manifest describes the components of the contextual gadget:
  - The extractors
  - Filters on pre-canned extractors
  - The gadgets to trigger
  - Scopes for data access

What is an extractor?

Example manifest: extractors and filters

```
...
<Extension id="testExtractor"
  type="contextExtractor">
  <Url>TestExtractorId</Url>
  <Param name="extractedField" value="regexp"/>
  <Triggers ref="exampleGadget"/>
  <Scope ref="exampleScope"/>
  <Container name="mail"/>
</Extension>

<Extension id="exampleGadget" type="gadget">
  <Url>http://example.com/mygadget.xml</Url>
  <Container name="mail"/>
</Extension>
...
```

What is an extractor?

Example manifest: triggers and gadgets

```
...
<Extension id="testExtractor"
  type="contextExtractor">
  <Url>TestExtractorId</Url>
  <Param name="extractedField" value="regexp"/>
  <Triggers ref="exampleGadget"/>
  <Scope ref="exampleScope"/>
  <Container name="mail"/>
</Extension>

<Extension id="exampleGadget" type="gadget">
  <Url>http://example.com/mygadget.xml</Url>
  <Container name="mail"/>
</Extension>
...
```

What is an extractor?

Example manifest: scopes

```
...
<Extension id="testExtractor"
  type="contextExtractor">
  <Url>TestExtractorId</Url>
  <Param name="extractedField" value="regexp"/>
  <Triggers ref="exampleGadget"/>
  <Scope ref="exampleScope"/>
  <Container name="mail"/>
</Extension>

<Extension id="exampleGadget" type="gadget">
  <Url>http://example.com/mygadget.xml</Url>
  <Container name="mail"/>
</Extension>
...
```

What is an extractor?

Example manifest: scopes

```
...
<Scope id="exampleScope">
  <Url>tag:google.com,2010:
    auth/contextual/extractor/SUBJECT</Url>
  <Reason>This gadget uses the subject line to
    determine context</Reason>
</Scope>
...
```

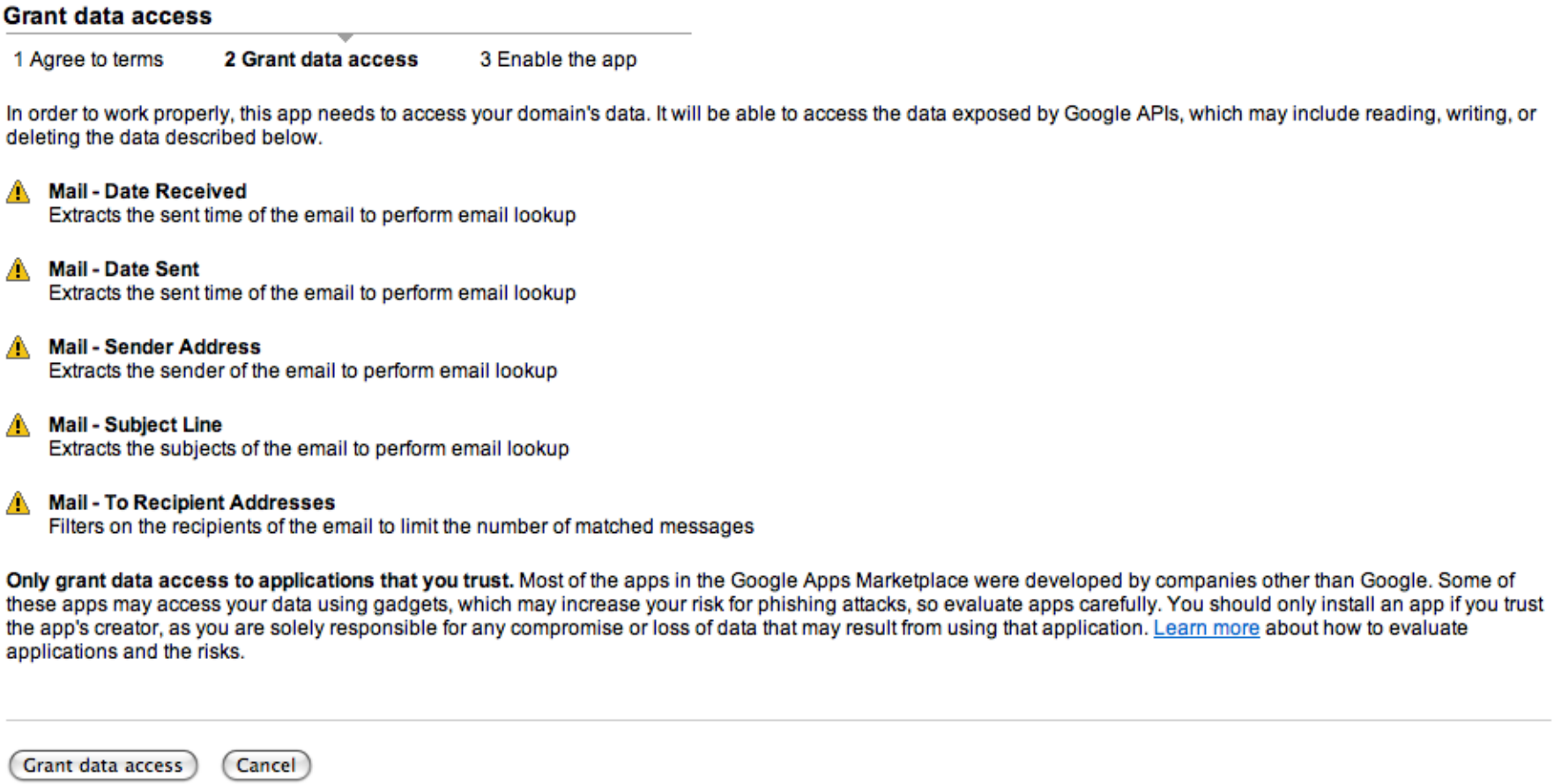
What is an extractor?

Example manifest: scopes

```
...
<Scope id="exampleScope">
  <Url>tag:google.com,2010:
    auth/contextual/extractor/SUBJECT</Url>
  <Reason>This gadget uses the subject line to
    determine context</Reason>
</Scope>
...
```

What is an extractor?

Scopes during installation



Grant data access

1 Agree to terms **2 Grant data access** 3 Enable the app

In order to work properly, this app needs to access your domain's data. It will be able to access the data exposed by Google APIs, which may include reading, writing, or deleting the data described below.

⚠ **Mail - Date Received**
Extracts the sent time of the email to perform email lookup

⚠ **Mail - Date Sent**
Extracts the sent time of the email to perform email lookup

⚠ **Mail - Sender Address**
Extracts the sender of the email to perform email lookup

⚠ **Mail - Subject Line**
Extracts the subjects of the email to perform email lookup

⚠ **Mail - To Recipient Addresses**
Filters on the recipients of the email to limit the number of matched messages

Only grant data access to applications that you trust. Most of the apps in the Google Apps Marketplace were developed by companies other than Google. Some of these apps may access your data using gadgets, which may increase your risk for phishing attacks, so evaluate apps carefully. You should only install an app if you trust the app's creator, as you are solely responsible for any compromise or loss of data that may result from using that application. Learn more about how to evaluate applications and the risks.

Grant data access Cancel

How do I test and distribute contextual gadgets?

How do I test and distribute contextual gadgets?

Development flow

- Log into the Google Apps Marketplace: http://google.com/enterprise/marketplace/
- Create a vendor profile if you don't have one
- Create a listing
- Enter an application manifest
- Deploy to your domain

How do I test and distribute contextual gadgets?

Developing for own domain

- Developers writing contextual gadgets for their own domain can create unpublished listings to install on their domain

How do I test and distribute contextual gadgets?

Show me the money

- Publish the listing to the Marketplace with the push of a button

Common tasks and best practices

Common tasks and best practices

Using extractors

- Use pre-canned extractors if possible
- Select the smallest necessary scope when matching data
- Match on as few emails as possible

Common tasks and best practices

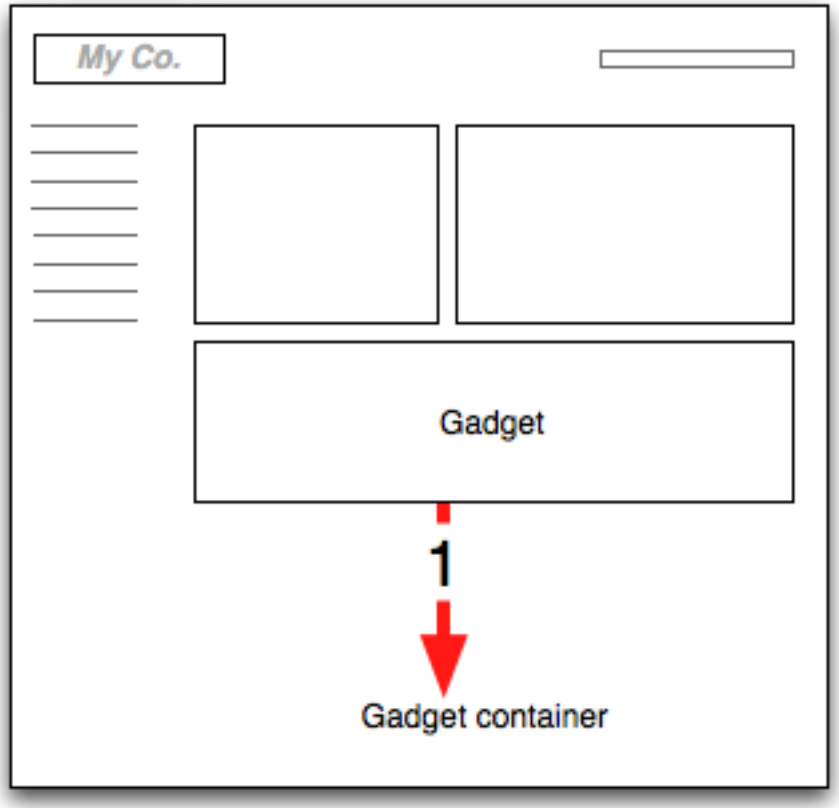
Writing your gadget UI

- If your gadget doesn't have anything to show, handle it gracefully
- Provide an option to expand or collapse your gadget UI, and remember the user's preference
- Use gadgets.window.adjustHeight() to return unused space back to the user

Common tasks and best practices

Fetching data

1. Gadget sends content request to parent page (container) using osapi.http.get (or .post)

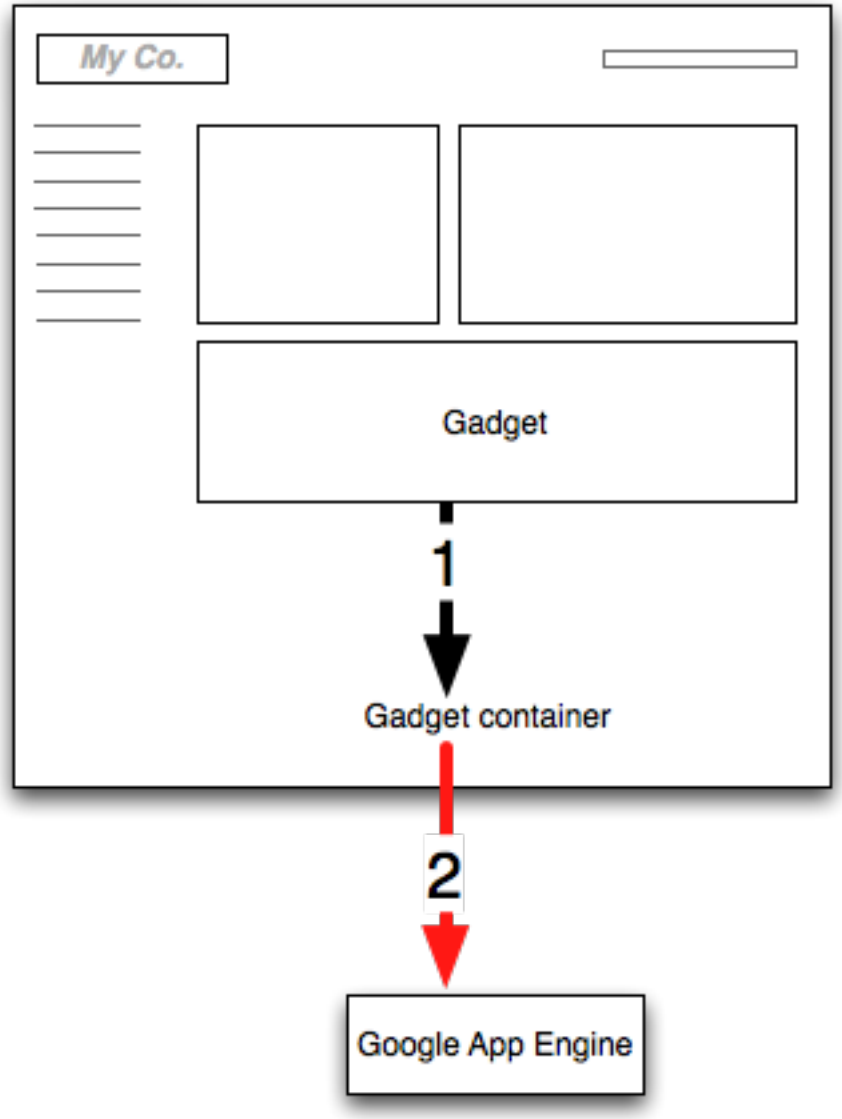



```
osapi.http.get({
 'href': 'http://example.com',
 'format': 'json',
 'authz': 'signed'
}).execute(handleResponse);
```

Common tasks and best practices

Fetching data

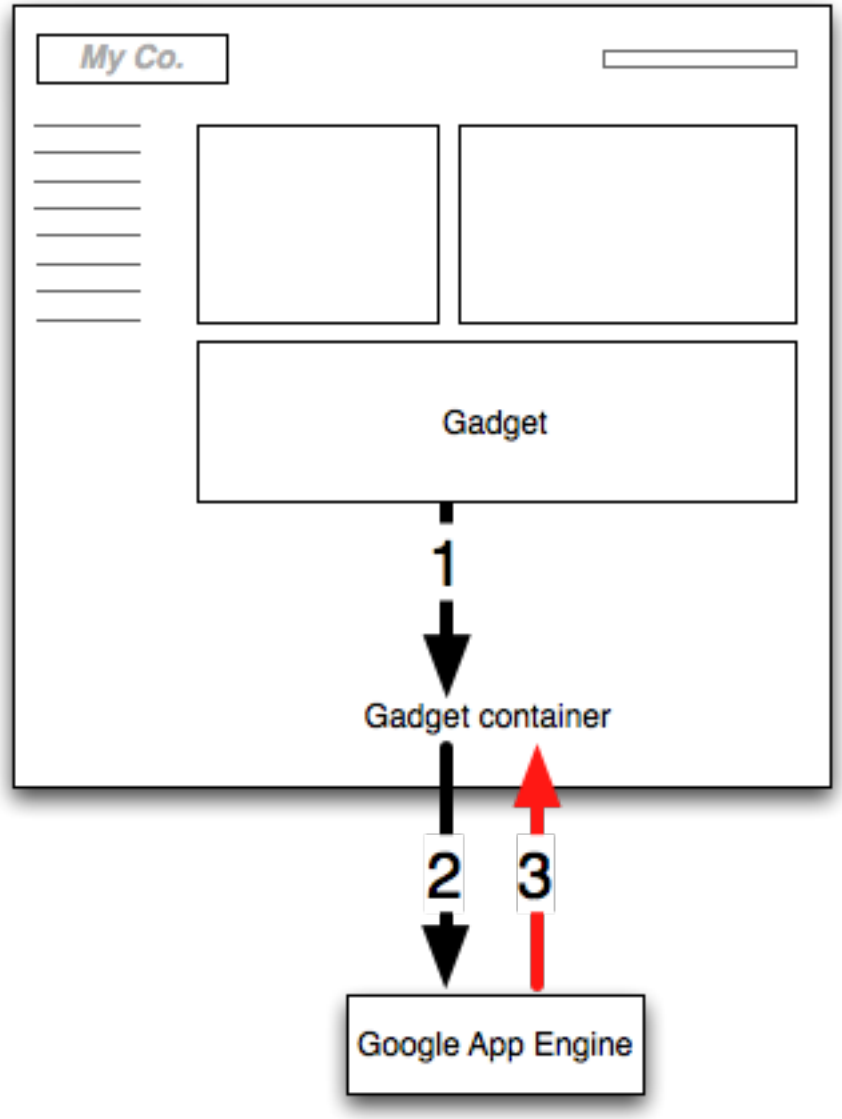
1. Gadget sends content request to parent page (container) using osapi.http.get (or .post)
2. Container proxies request to remote server



Common tasks and best practices

Fetching data

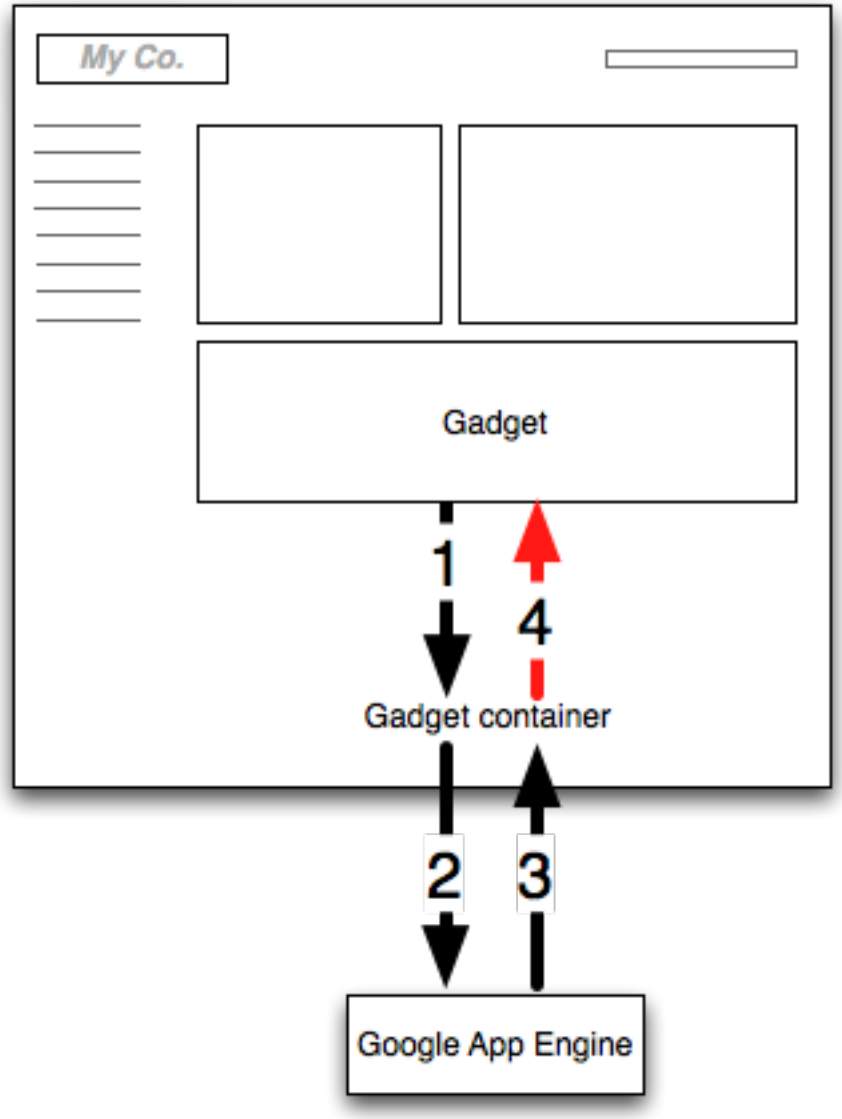
1. Gadget sends content request to parent page (container) using osapi.http.get (or .post)
2. Container proxies request to remote server
3. Server receives request and returns data



Common tasks and best practices

Fetching data

1. Gadget sends content request to parent page (container) using osapi.http.get (or .post)
2. Container proxies request to remote server
3. Server receives request and returns data
4. Data is passed to the gadget

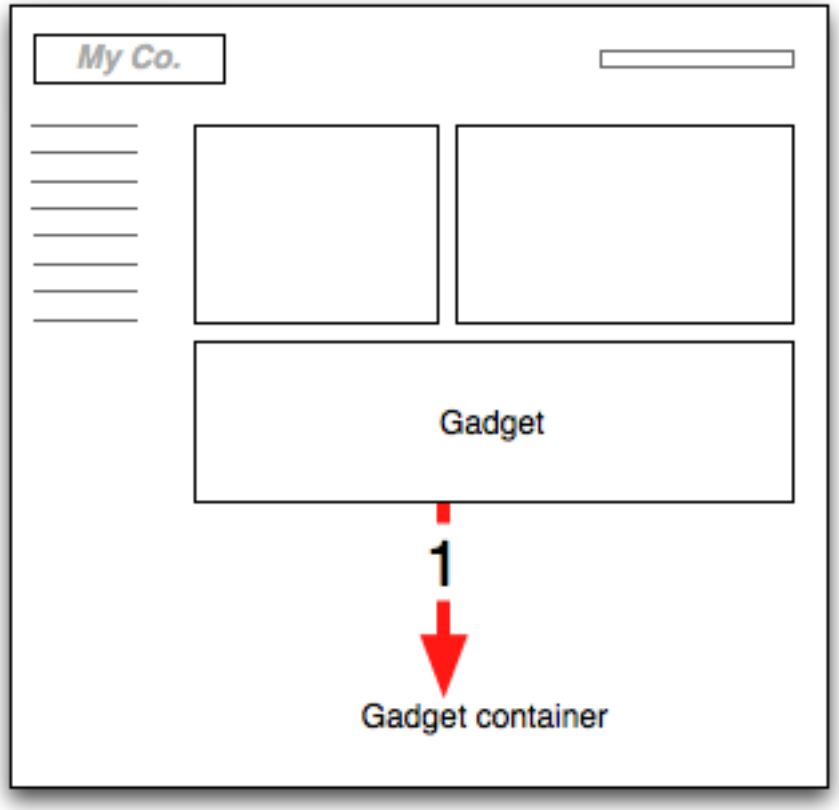


Common tasks and best practices

Authentication

1. Gadget sends content request to parent page (container) using osapi.http.get (or .post)

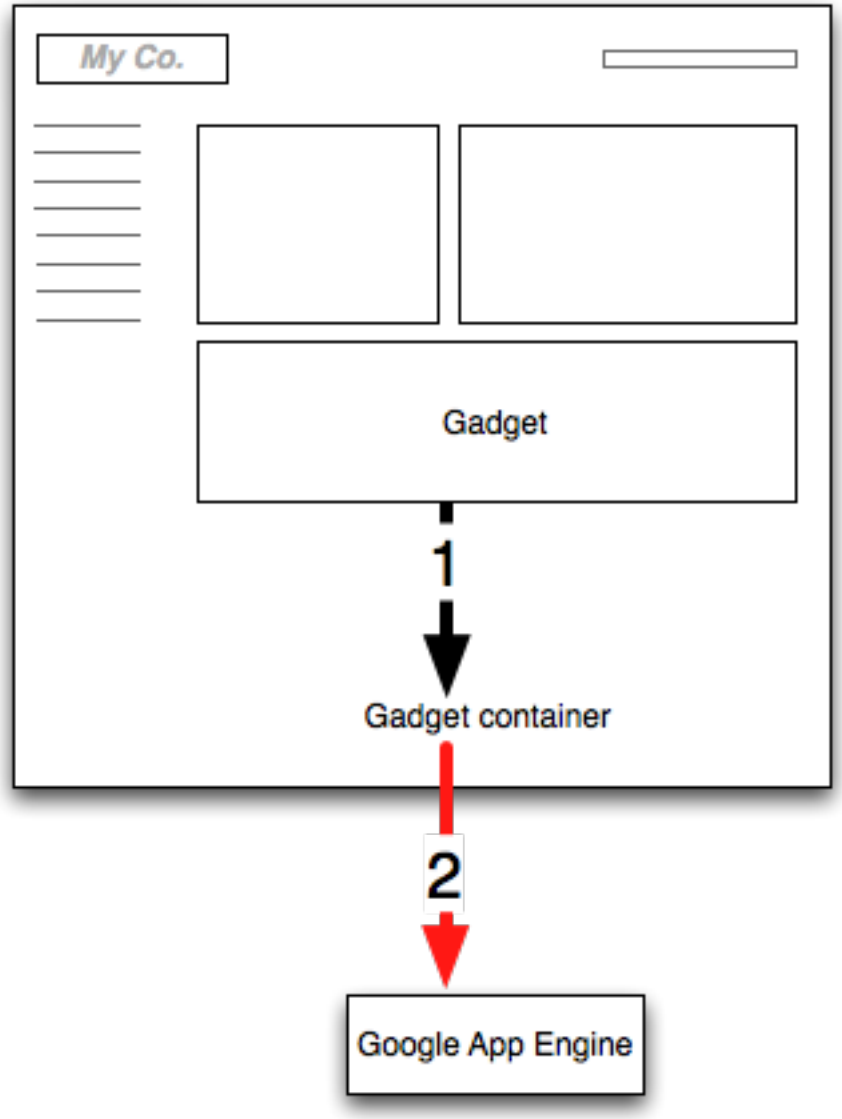
```
osapi.http.get({
 'href': 'http://example.com',
 'format': 'json',
 'authz': 'signed'
}).execute(handleResponse);
```



Common tasks and best practices

Authentication

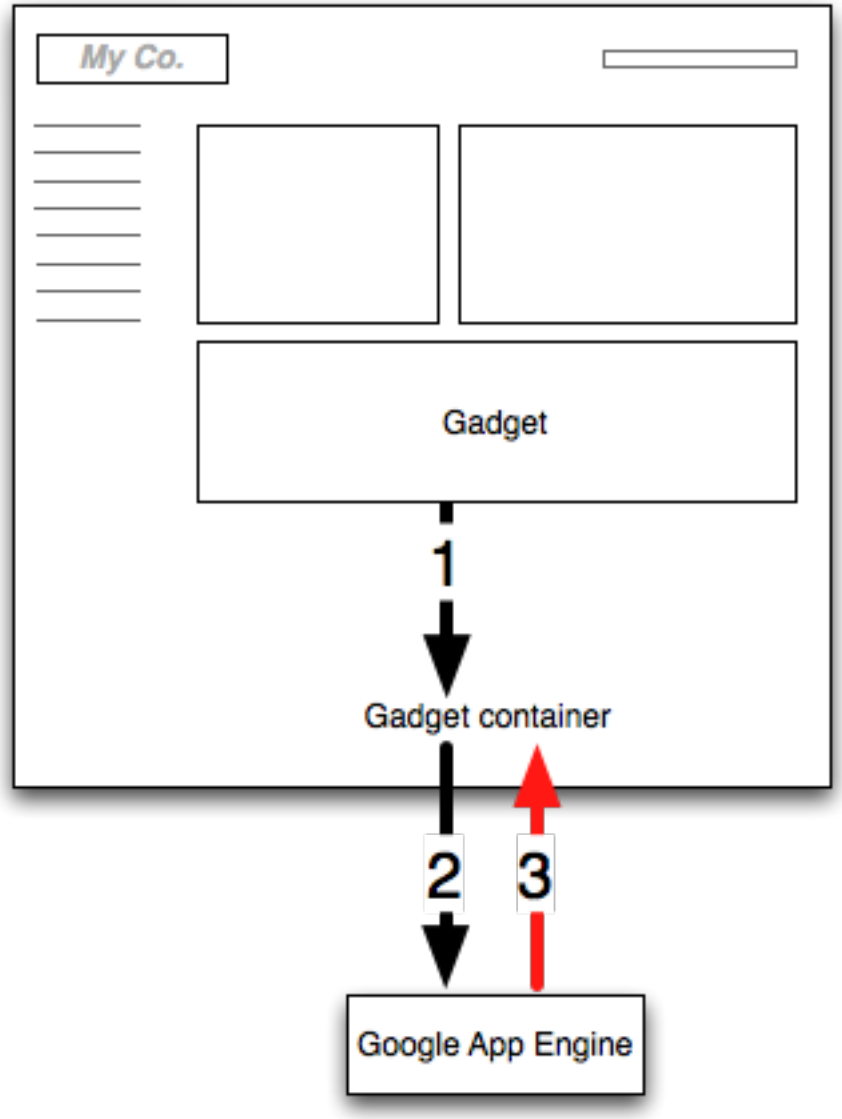
1. Gadget sends content request to parent page (container) using osapi.http.get (or .post)
2. Container appends unique user ID (OpenSocial ID) and signs request, before transmitting to remote server



Common tasks and best practices

Authentication

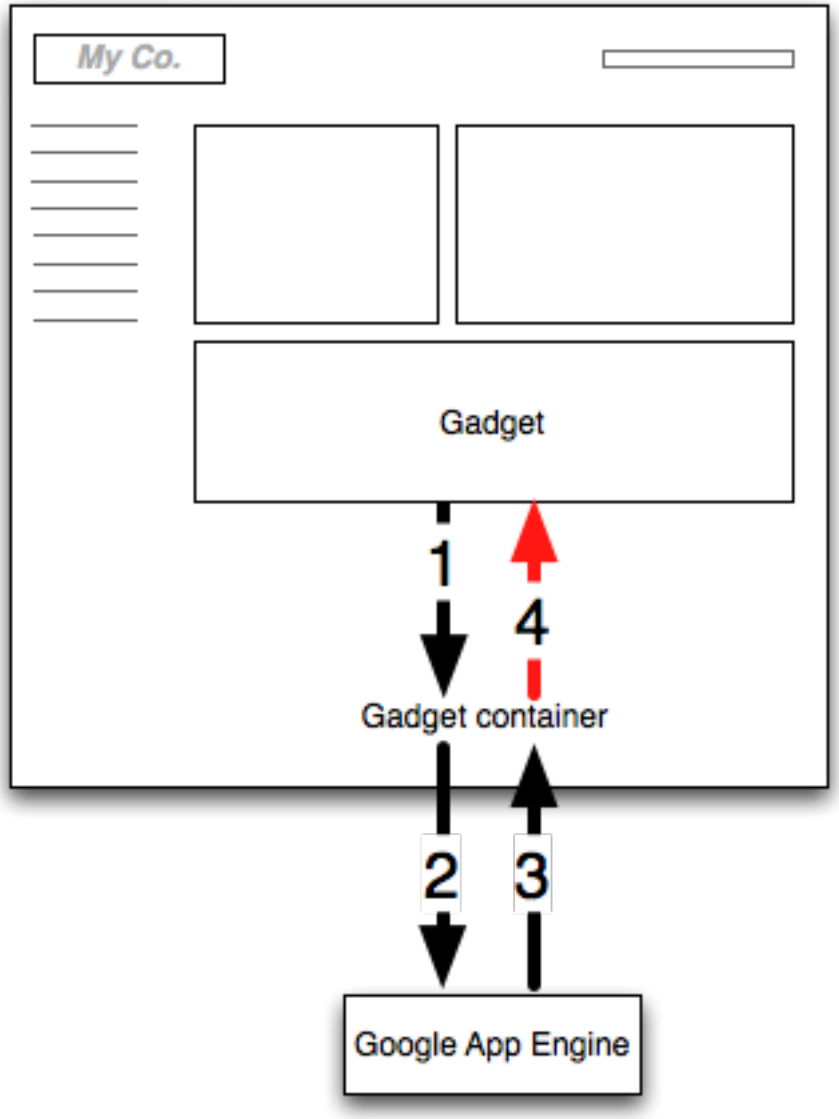
1. Gadget sends content request to parent page (container) using osapi.http.get (or .post)
2. Container appends unique user ID (OpenSocial ID) and signs request, before transmitting to remote server
3. Server receives request and returns data



Common tasks and best practices

Authentication

1. Gadget sends content request to parent page (container) using osapi.http.get (or .post)
2. Container appends unique user ID (OpenSocial ID) and signs request, before transmitting to remote server
3. Server receives request and returns data
4. Data is passed to the gadget



Common tasks and best practices

What if I need more than an ID?

- Existing applications can add an OpenSocial ID to their 'users' table
- When signed requests come in, see if that OpenSocial ID exists in the database — if it does, you're all set
- When the ID doesn't exist, create a one-time use token, and return that to the gadget
- Have the gadget open a window to your server (append the token to the request!) and log in, using OpenID — save the OpenSocial ID stored with that token in the database

Common tasks and best practices

Authentication (in code)

```
osapi.http.get({
  'href': 'http://example.com/json',
  'format': 'json',
  'authz': 'signed'
}).execute(handleResponse);
```

Common tasks and best practices

Authentication (in code)

```
user = Users.gql('WHERE opensocial_id = :1',
        self.request.get('viewer_id')).get()

if not user:
  temp = TempUser(opensocial_id =
                  self.request.get('viewer_id'),
                  token = session_token)
  return {'status': 'not found',
          'token': session_token}
else:
  return {'status': 'found'}
```

Common tasks and best practices

Authentication (in code)

```
var popup = window.open(openid_url, 'OpenID',
  'width=300,height=300');

finishedInterval = setInterval(function() {
 if (popup.closed) {
  osapi.http.get({
   'href': 'http://example.com/json',
   'format': 'json',
   'authz': 'signed'
  }).execute(handleResponse);

  clearInterval(finishedInterval);
 }
}, 100);
```

Common tasks and best practices

Writing safe gadgets

- Stick to HTML and CSS specs
- Use only safe DOM operations
- Use supported event handlers and timers
- Use supported frameworks:
  - Google Web Toolkit
  - OpenSocial Templates
  - JQuery
- Gmail contextual gadgets *will* use Caja!*
- More at http://code.google.com/p/google-caja/
- Caja Playground: http://caja.appspot.com

*(at some point)

Common tasks and best practices

The good, the bad, and the ugly

Good:

```
jQuery('#content').append('hello world!');
document.getElementById('content').
  appendChild(document.createTextNode(
    'hello world!'));
```

Bad:

```
document.write('hello world!');
```

Ugly:

```
window.parent.location = myApi.get(url);
```

An example application

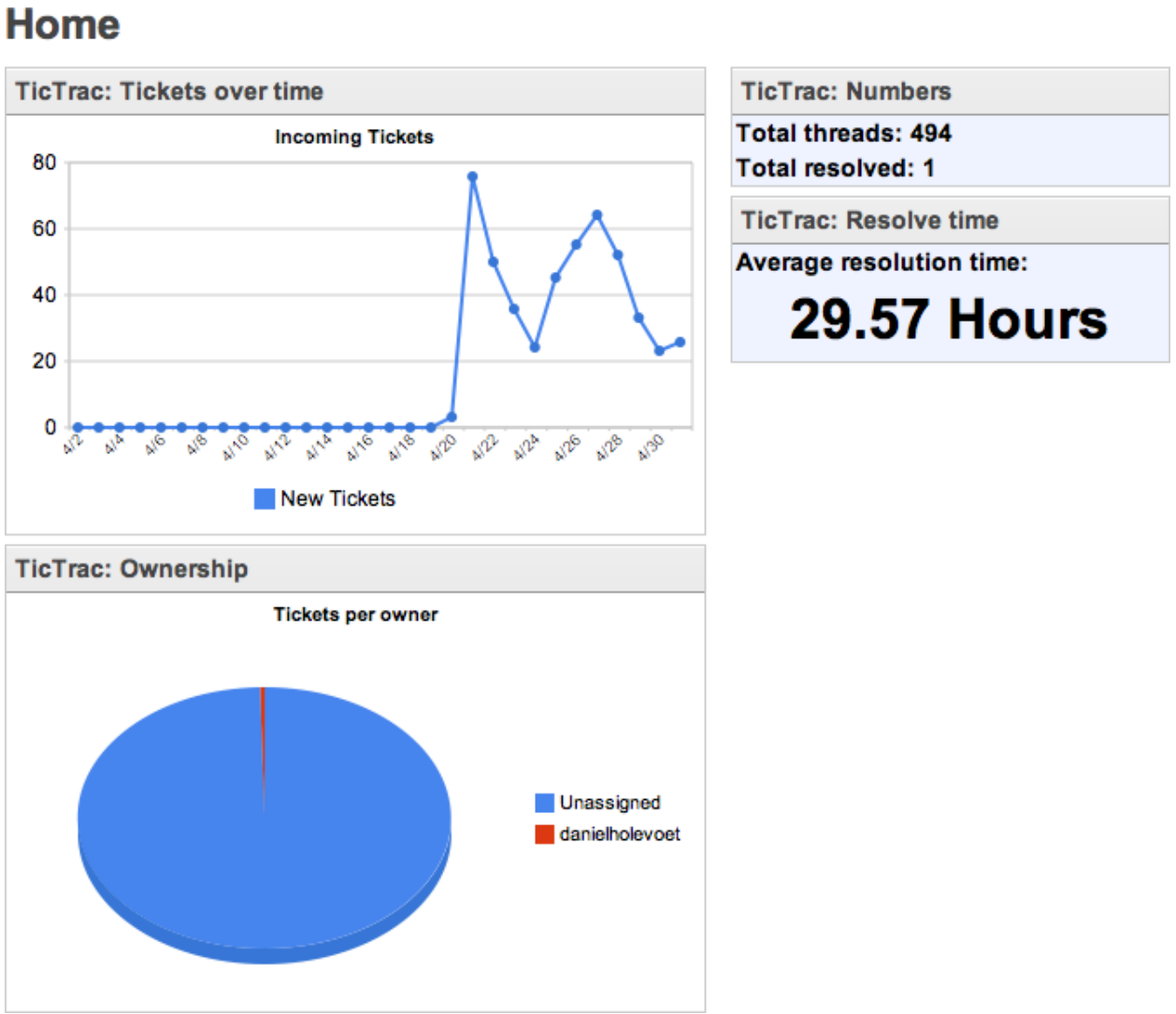
An example application

Feature summary

- Ticket tracking system
- Provides options to assign owner and status to email threads (in Gmail)
- Gadgets provide dashboard views embeddable in Google Apps

An example application

Examples of gadget usage



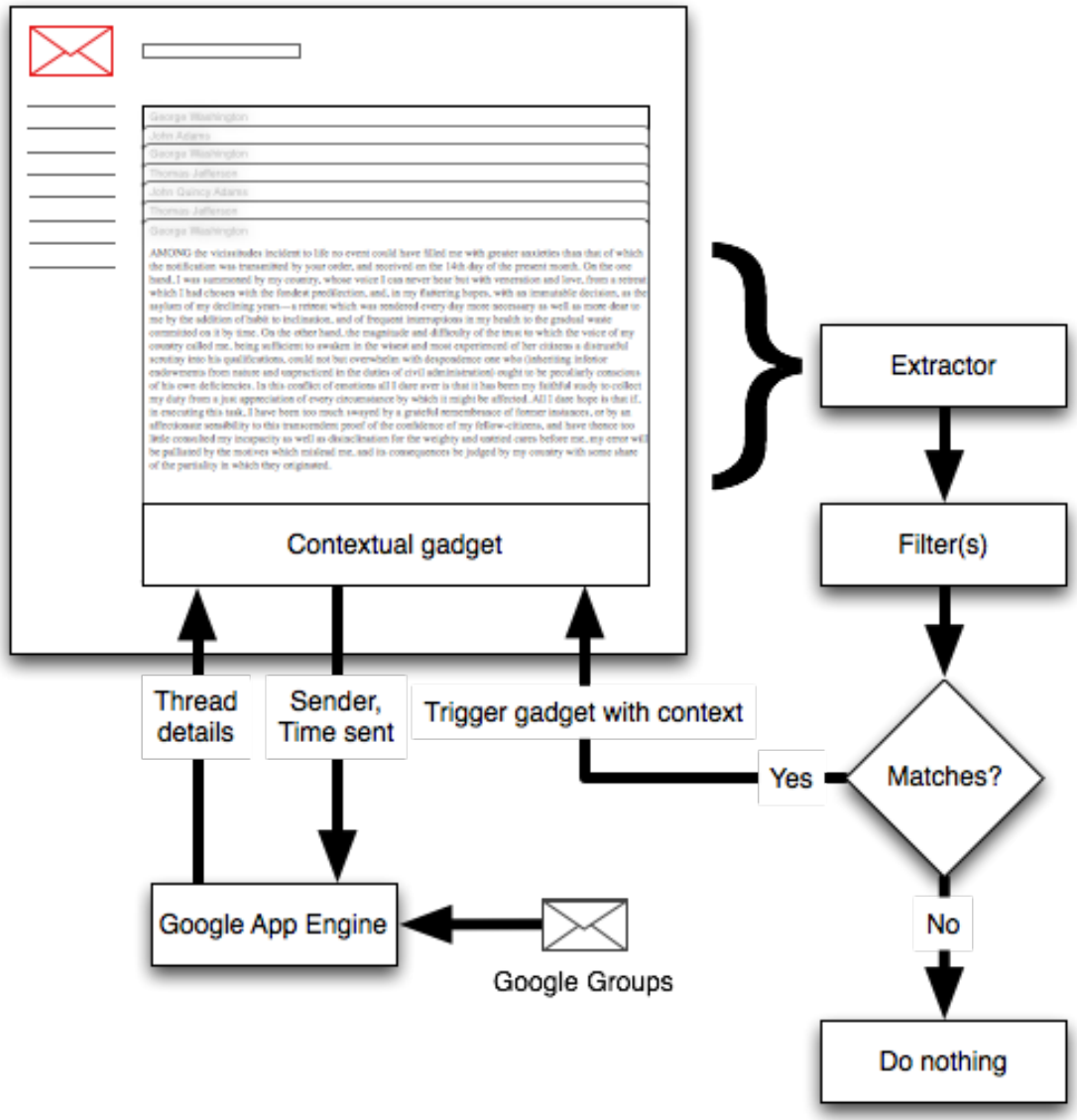
An example application

How TicTrac uses contextual gadgets

- Incoming messages are filtered by the recipient list (a set of mailing lists trigger a match)
- The sender and time sent are extracted
- These fields are sent to a Google App Engine service, to return thread details

An example application

Architecture diagram



An example application

Using the matches

```
var matches = google.contentmatch.getContentMatches();
var context = {};
for (var match in matches) {
  for (var key in matches[match]) {
    context[key] = matches[match][key];
  }
}

// Gives: context.time_sent, context.sender
```

An example application

Retrieving the thread from App Engine

```
getThreadFromDetails(context.time_sent,
                     context.sender_email,
                     function(threadid) {
  displayStatusBar(threadid);
  displayThreadActions(threadid);
  gadgets.window.adjustHeight();
});

/* Constructs a payload like:
payload = {
  'method':'retrieve_thread_from_details',
  'sent':context.time_sent,
  'sender':context.sender_email
};
*/
```

An example application

The application manifest

```
...
<Extension id="RecipientExtractor" type="contextExtractor">
  <Url>885565299:RecipientToEmailExtractor</Url>
  <Triggers ref="TicTracGadget"/>
  <Param name="recipient_to_email" value="my-mailing-
list@example.com"/>
  <Scope ref="timeSentScope" />
  <Scope ref="timeReceivedScope" />
  <Scope ref="recipientScope"/>
  <Container name="mail"/>
</Extension>
...
```

An example application

The application manifest

```
...
<Extension id="TicTracGadget" type="gadget">
  <Url>
    http://tictrac-test.appspot.com/gadget.xml
  </Url>
  <Container name="mail"/>
</Extension>
...
```

An example application

The application manifest

...

```
<Scope id="timeSentScope">
  <Url>tag:google.com,2010:auth/contextual/
    extractor/DATE_SENT</Url>
  <Reason>Extracts the sent time of the email to perform email
lookup</Reason>
</Scope>
```

...

How do I get started?

Getting started, today

- Developer documentation:
  - Contextual gadgets: http://code.google.com/apis/gmail/gadgets/contextual/
  - Marketplace: http://code.google.com/googleapps/marketplace/
- Developer forum:
  - http://bit.ly/bqPXht
- Check out the other Enterprise talks online when they are posted: http://code.google.com/events/io/2010/

Thanks to contextual gadget authors!

http://bit.ly/get-contextual

Google 10 I/O

Got questions?

Got questions?

We're using Wave and Google Moderator:
http://bit.ly/bcu4jE


Google™
10