How to implement your own OpenSocial Container on the Java Platform

Chris Schalk, Google Developer Advocate
Paul Lindner, hi5 Architect

ID# TS-6574

Java

Sun microsystems

Learn the specific steps required to implementing your own OpenSocial container on the Java™ platform

GOAL

Agenda

- OpenSocial Overview
- Options for hosting OpenSocial applications
- Introducing Apache Shindig
- How to develop with Shindig
- Integrating external data with Shindig
- Putting it all together
- RealWorld Experience with Shindig - hi5

Agenda

- OpenSocial Overview
- Options for hosting OpenSocial applications
- Introducing Apache Shindig
- How to develop with Shindig
- Integrating external data with Shindig
- Putting it all together
- RealWorld Experience with Shindig - hi5

“What is OpenSocial?”

- OpenSocial is a set of common APIs for building social applications across the web
- It is being developed by Google in conjunction with partners from the Web/Social Application development community

What is OpenSocial trying to solve?

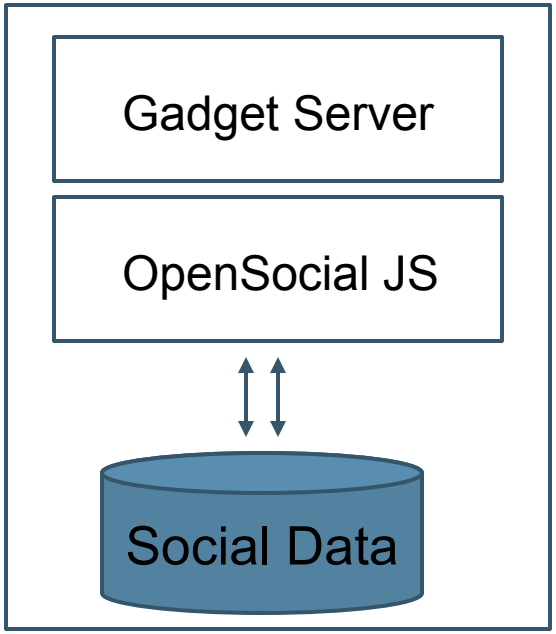
- As each social website opens its environment to developers, it exposes an API

  " … But what if you want to your application on multiple social websites?"

- Problem: Developers have to learn multiple APIs to publish in multiple environments

- Solution: OpenSocial allows developers to write applications to a common standard API that will run on multiple websites

OpenSocial Architecture at a glance



OpenSocial Site

JSON
(application communication)

Client

OpenSocial Architecture at a glance

Client

OpenSocial Client

- Web page with embedded OpenSocial Applications
- Applications are comprised of XML documents with JavaScript™ technology
  - Similar to iGoogle Gadget
  - Has additional functionality
    - OpenSocial JavaScript APIs

OpenSocial JavaScript APIs

- OpenSocial Application Developers essentially develop on the Gadget platform but with 3 core OpenSocial services…

OpenSocial JavaScript APIs overview

The core OpenSocial services include:

- People and Friends Data API
  - Access friends information programmatically
- Activities Data API
  - See what you're friends are up to
  - Share what you are doing
- Persistence Data API
  - Share data with your friends, the world

OpenSocial JavaScript APIs overview

Additional Gadgets services include:

- **Gadgets Core**
  Utilities handling gadget preferences, IO, JSON

- **Gadgets Feature-Specific**
  Utilities for working with flash, window management, tabs, rpc, MiniMessage

People & Friends API Code Example

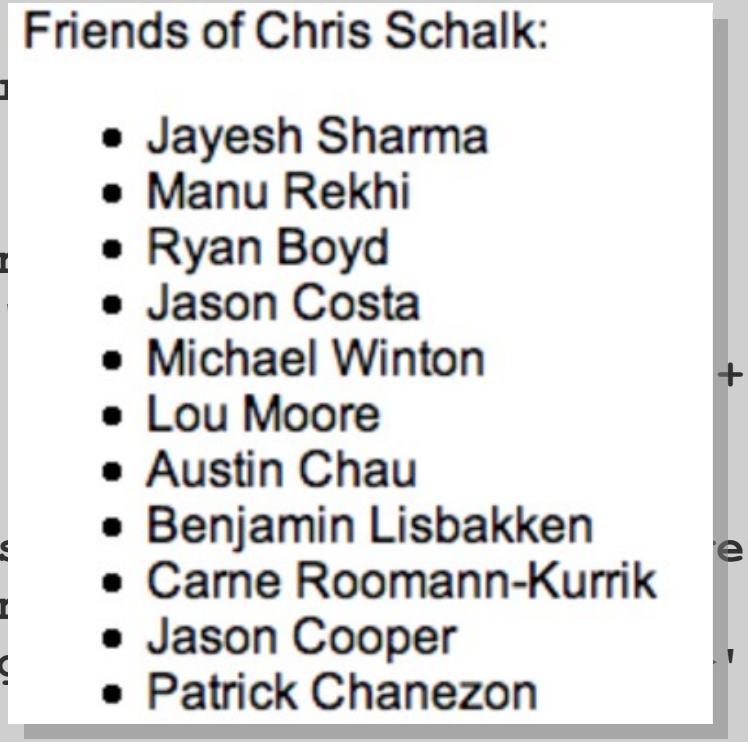
Getting info on **you** and **your friends**:

```
/**
* Request for friend info when the page loads.
*/

function getFriendData() {
   var req = opensocial.newDataRequest();
   req.add(req.newFetchPersonRequest(VIEWER), 'viewer');
   req.add(req.newFetchPeopleRequest(VIEWER_FRIENDS),


     'viewerFriends');
   req.send(onLoadFriends);
};
```

Core Services - People

Getting info on you and **your friends**:

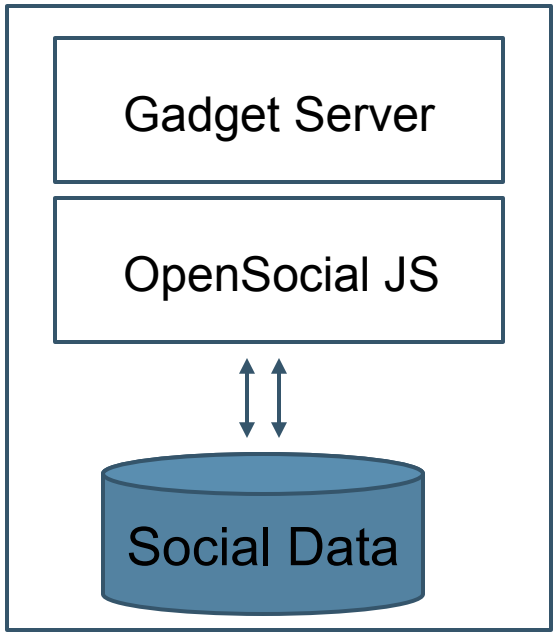
Generated output:




```
/**
* Callback function for retu
*/

function onLoadFriends(respo
  var viewer = response.get(
  var html = 'Friends of ' +                                    +
      ':<br><ul>';

  var viewerFriends = respons                                   etData();
  viewerFriends.each(functio
    html += '<li>' + person.g                                   ';});
  html += '</ul>';
  document.getElementById('message').innerHTML = html;
};
```

OpenSocial Architecture at a glance

JSON Request/Responses Exposed



[{"type":**"FETCH_PEOPLE"**,"idSpec
":"VIEWER","profileDetail":["id","nam
e","thumbnailUrl"]
{"type":"FETCH_PERSON_APP_DA
TA","idSpec":"VIEWER","…

{**"Responses"**:[{"response":{"Value":{"List":
[{"AboutMe":"","AllowSend":true,"FullName"

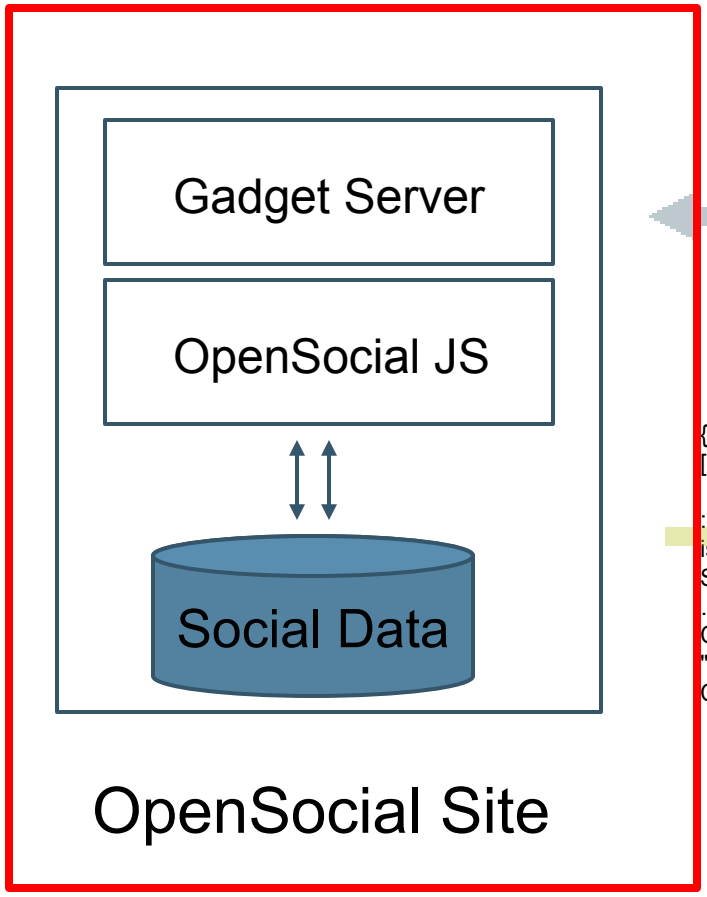
:{"FamilyName":"Schalk","GivenName":"Chr
is","Unstructured":"Chris
Schalk"},"Id":"05047698136432048591"
…
Chanezon"},"Id":"01515881069164444926",
"Location":"","Name":"Patrick
Chanezon","Nickname":"","Occupation"

Client

OpenSocial Site

Getting familiar with OpenSocial Application Development

OpenSocial Architecture at a glance

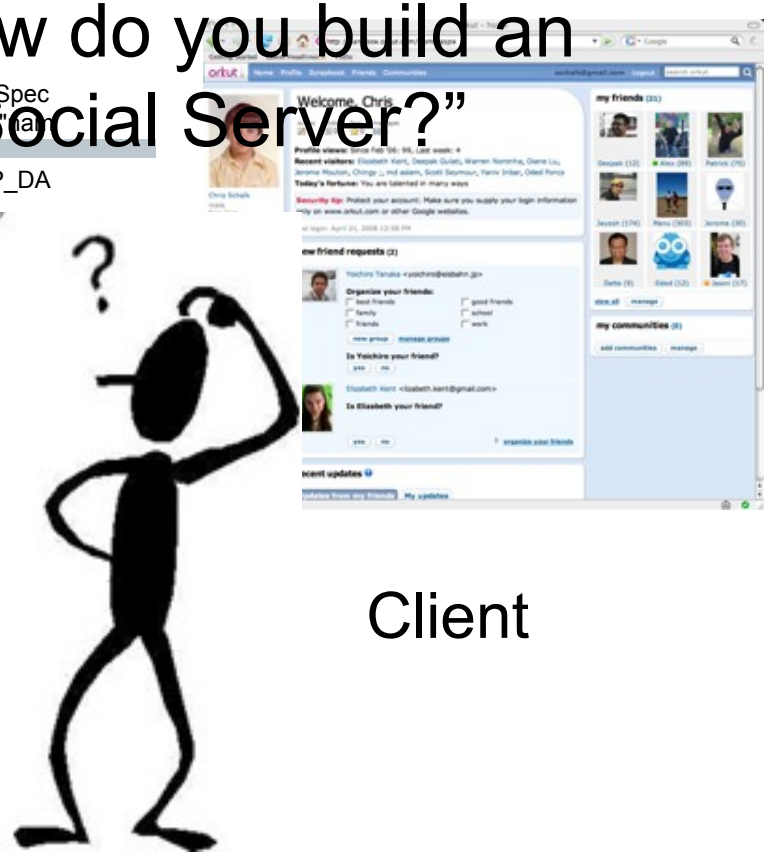


"So how do you build an OpenSocial Server?"

[{"type":"FETCH_PEOPLE","idSpec
":"VIEWER","profileDetail":["id","nam
e","thumbnailUrl"]}
{"type":"FETCH_PERSON_APP_DA
TA","idSpec":"VIEWER",”…

{"Responses":[{"response":{"Va
[{"AboutMe":"","AllowSend":true,'

:{"FamilyName":"Schalk","Given
is","Unstructured":"Chris
Schalk"},"Id":"050476981364320
…
Chanezon"},"Id":"015158810691
"Location":"","Name":"Patrick
Chanezon","Nickname":"","Occu



Client

Agenda

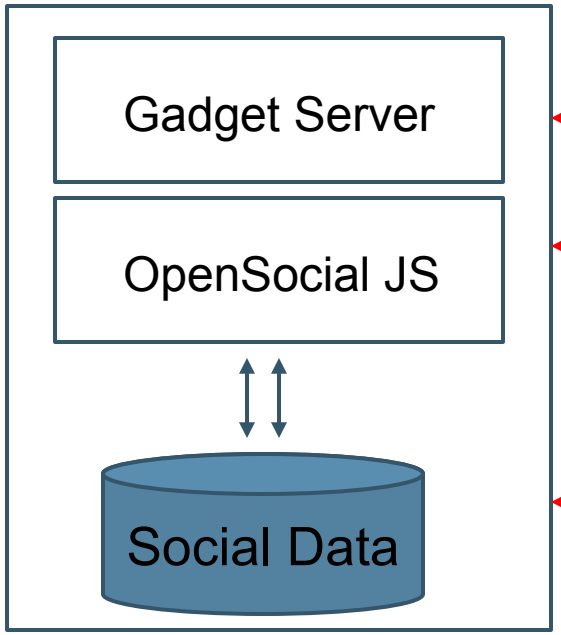
- OpenSocial Overview
- Options for hosting OpenSocial applications
- Introducing Apache Shindig
- How to develop with Shindig
- Integrating external data with Shindig
- Putting it all together
- RealWorld experience with Shindig - hi5

Options for hosting OpenSocial Apps

- Option #1: Write your own spec compliant code in whatever language you want…
  - Other early “pioneer” OpenSocial containers went this route.
- Option #2: Utilized existing OpenSource container code.
  - **The Apache incubator project “Shindig” serves this purpose!**

OpenSocial Architecture at a glance

OpenSocial container site requirements



- Container must support Gadgets API
- Container must support OpenSocial JS API
- Container must have ability to tie into external social data

OpenSocial Site

Agenda

- OpenSocial Overview
- Options for hosting OpenSocial applications
- Introducing Apache Shindig
- How to develop with Shindig
- Integrating external data with Shindig
- Putting it all together
- RealWorld experience with Shindig - hi5

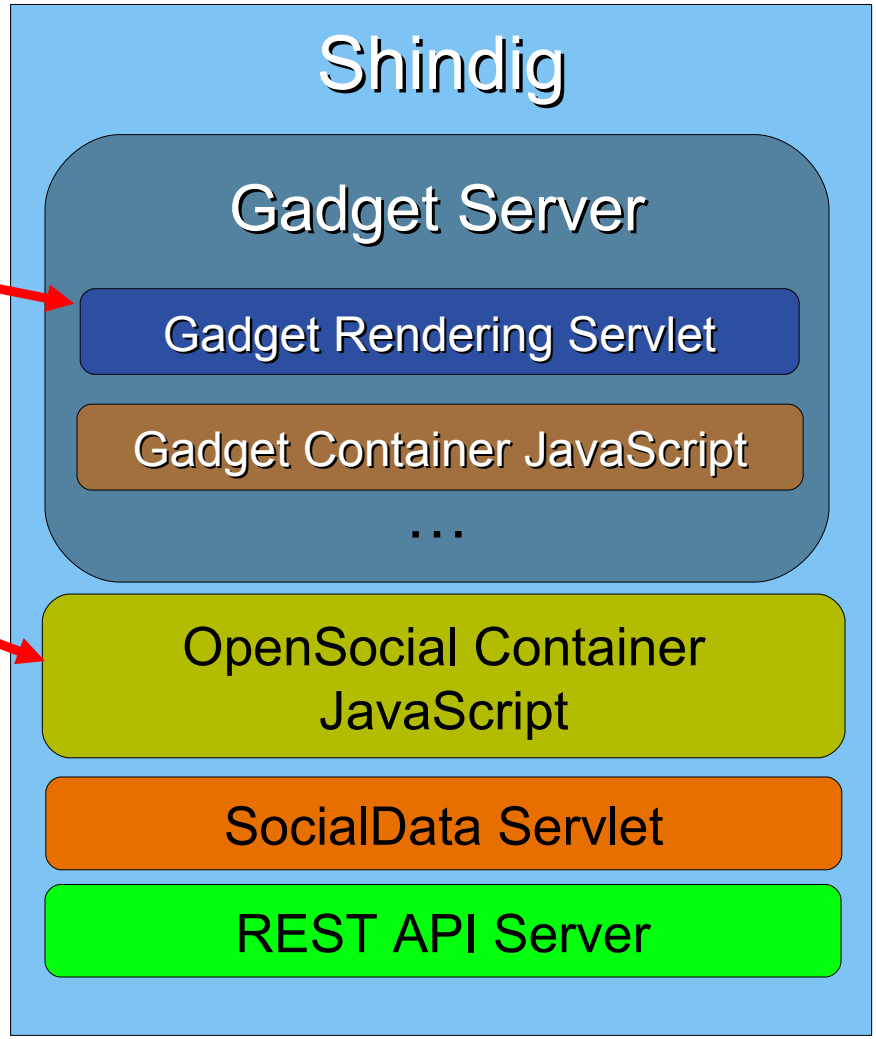
OpenSocial's Container - Shindig

- What is Shindig?
  - "OpenSource software that allows you to *serve* OpenSocial applications"
- Is currently an Apache Software Incubator project
  - Heavy partner involvement (Ning, hi5 …)
  - Serves as open source reference implementation of OpenSocial & Gadgets technologies
- It's Goal:

  "To serve as an easy to use OpenSocial "container in a box"

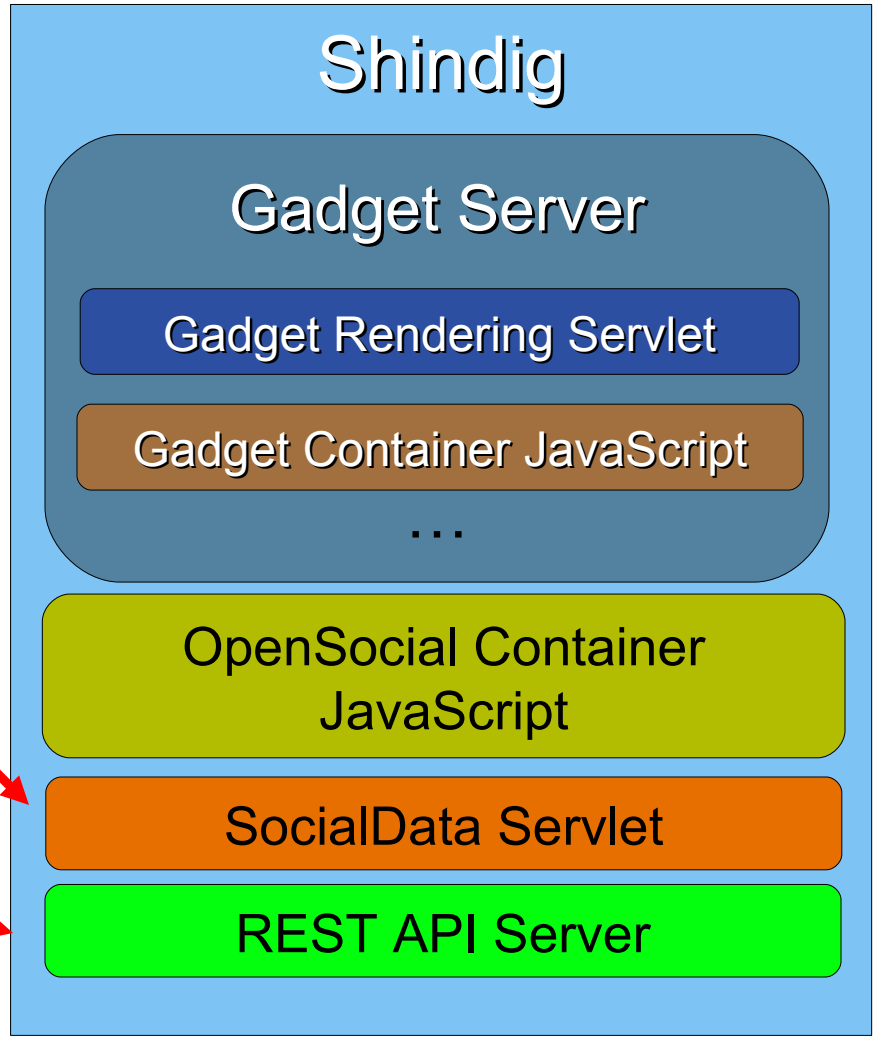
Shindig Components

- Gadget Server
  - Renders gadget XML (i.e. from gmodules.com)
  - Gadget Container JavaScript environment
- OpenSocial Container JavaScript environment
  - JavaScript environment for people, activities, persistence

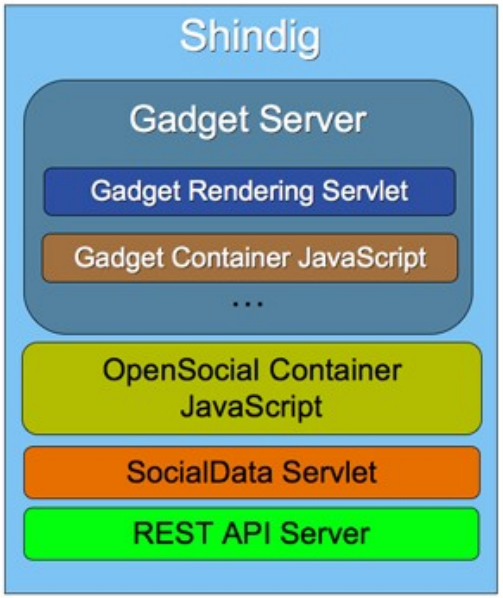


Shindig Components

- SocialData Servlet
  - Handles requests for data and sends to data handlers
- OpenSocial REST APIs
  - Provide lightweight REST access to social data *(currently being implemented)*



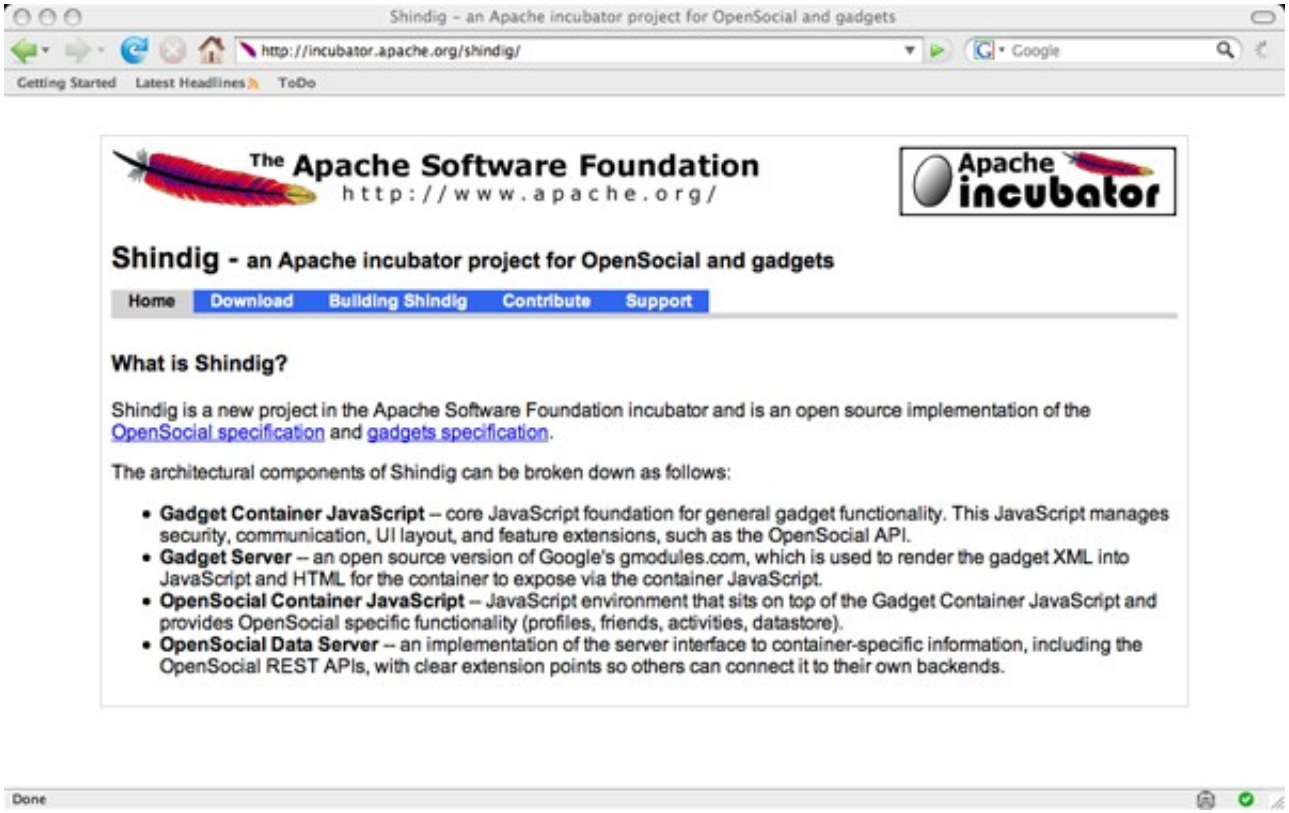
Shindig in Action



- **Application Installation**
  Gadget XML is loaded and cached on OpenSocial Container

- **Running the application**
  Request is made from Client
  Data is returned and rendered

Shindig Website



http://incubator.apache.org/shindig

Agenda

- OpenSocial Overview
- Options for hosting OpenSocial applications
- Introducing Apache Shindig
- How to develop with Shindig
- Integrating external data with Shindig
- Putting it all together
- RealWorld experience with Shindig - hi5

Developing with Shindig

- Easy to get started…
  - Follow instructions on website building Shindig
    - You'll need a 1.5 Java™ Development Kit (JDK™ software/Java Runtime Environment (JRE™) software version
    - A svn client to check out the code
    - Apache Maven to build Shindig
      - *Automated build downloads are not yet available*
- Once code is checked out, follow instructions in README files

Checking out, building and running Shindig *(in about a minute!)*

Working with Shindig in an IDE

- Shindig can be worked on from and IDE
- There's guidance on the Shindig website on how to setup Eclipse to work with Shindig

Setting up Eclipse to build and debug Shindig

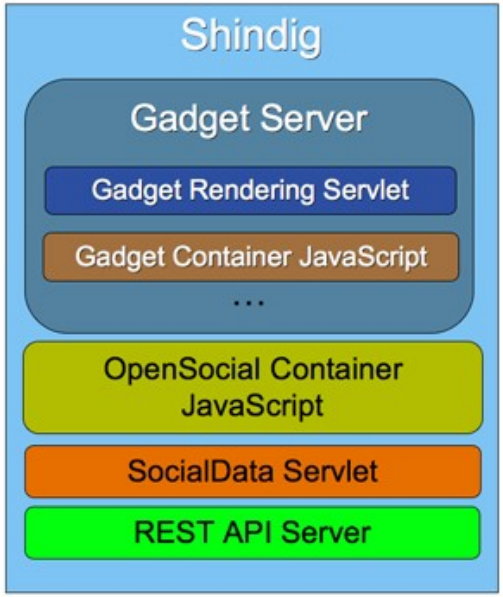
DEMO

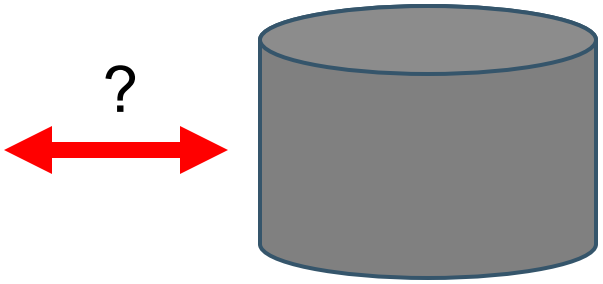
Agenda

- OpenSocial Overview
- Options for hosting OpenSocial applications
- Introducing Apache Shindig
- How to develop with Shindig
- Integrating external data with Shindig
- Putting it all together
- RealWorld experience with Shindig - hi5

Integrating external data with Shindig

"So how can you make Shindig talk to your data?"





Integrating external data with Shindig

Solution!

Implement these interfaces and tie in to your own datasources:

- PeopleService
- ActivityService
- DataService

- Can use JDBC directly or Hibernate ... or other persistence frameworks…

Integrating external data with Shindig

PeopleService Interface

```
public interface PeopleService {
   public ResponseItem<ApiCollection<Person>> getPeople(List<String> ids,

     SortOrder sortOrder, FilterType filter, int first, int max,
     Set<String> profileDetails, GadgetToken token);


   public List<String> getIds(IdSpec idSpec, GadgetToken token)
     throws JSONException;

   public ResponseItem<ApiCollection<Person>> getPeople(List<String> ids,

     SortOrder sortOrder, FilterType filter, int first, int max,
     Set<String> profileDetails, GadgetToken token);
   …
}
```

Integrating external data with Shindig

Example: A SQL Implementation

```
public class SQLPeopleService implements PeopleService {
      public ResponseItem<ApiCollection<Person>>

   getPeople(List<String> ids, SortOrder sortOrder,
          FilterType filter, int first, int max,
          Set<String> profileDetails, GadgetToken token) {

    Map<String,Person> allPeople = SQLDataLayer.get().getAllPeople();

    return allPeople;
   }
   …
}
```

Integrating external data with Shindig

After creating your own implementations for People, Data and Activities, you bind your implementations using a provided Guice module

```
public class SocialApiGuiceModule extends AbstractModule {
  @Override
  protected void configure() {
bind(PeopleService.class).to(SQLPeopleService.class);
    bind(DataService.class).to(SQLDataService.class);
    bind(ActivitiesService.class).to(SQLActivitiesService.class);
    bind(new TypeLiteral<List<GadgetDataHandler>>() {})
       .toProvider(GadgetDataHandlersProvider.class);  }
…
}
```

Running the SampleContainer with custom data from a SQL database

Agenda

- OpenSocial Overview
- Options for hosting OpenSocial applications
- Introducing Apache Shindig
- How to develop with Shindig
- Integrating external data with Shindig
- Putting it all together
- RealWorld Experience with Shindig - Hi5

How to put it all together…

- Map your social data to OpenSocial data standards
- Security options
  - Can create your own implementations or use built in security OAuth with Shindig
- To render the gadgets, use JavaScript to embed gadgets onto your page

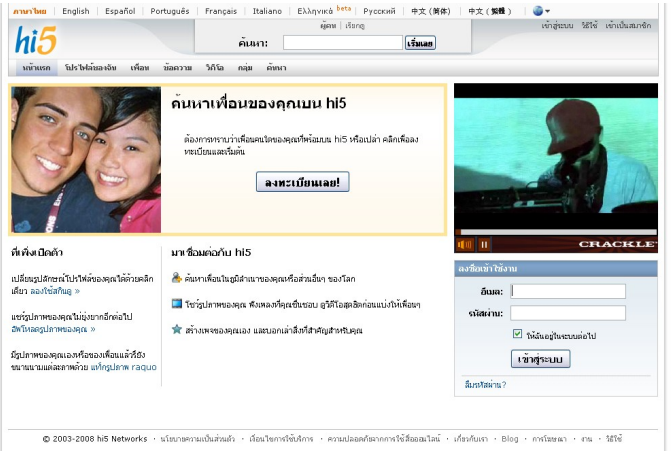
Putting it all together – Embedding OpenSocial Apps into your website

Agenda

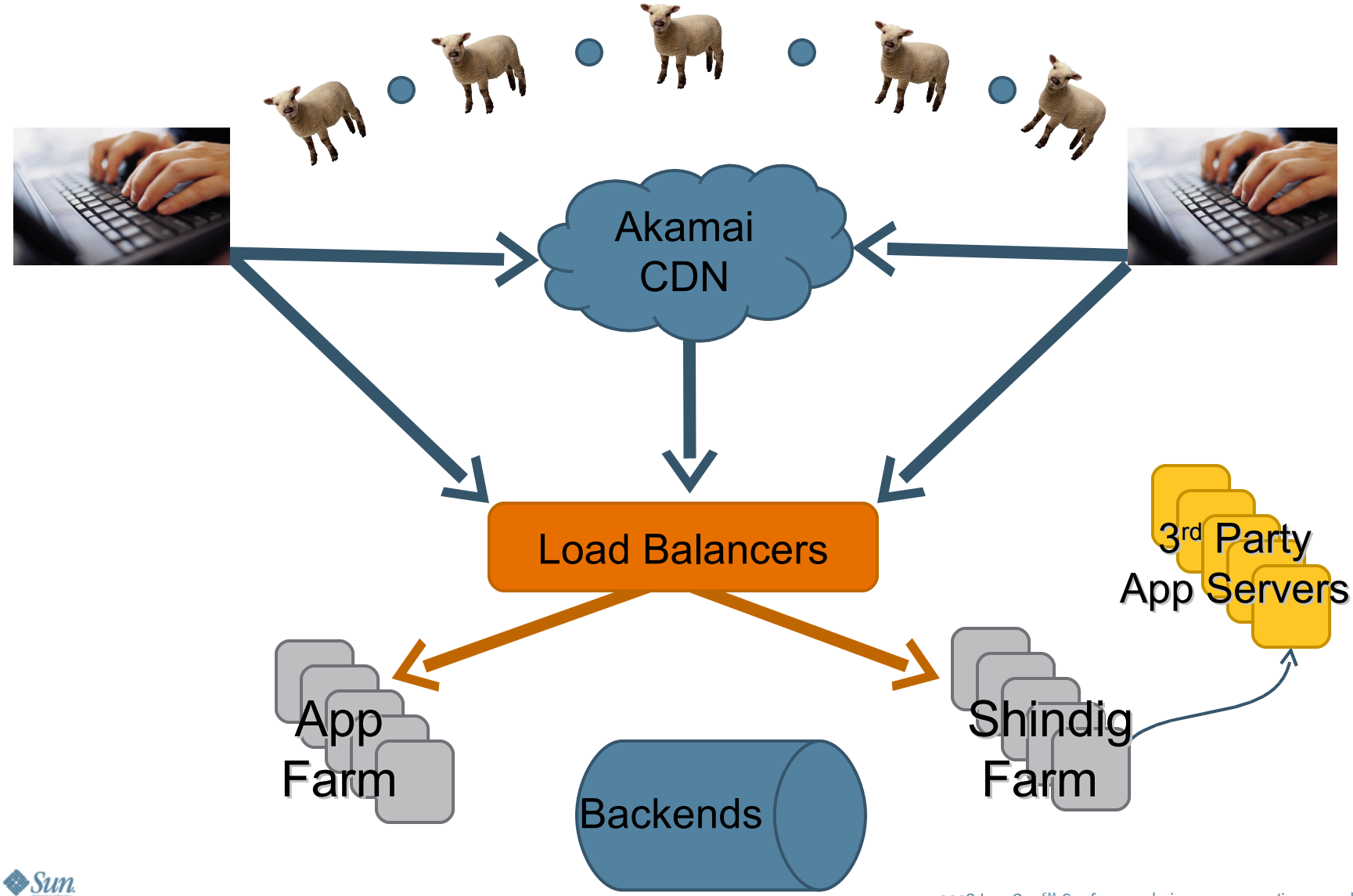
- OpenSocial Overview
- Options for hosting OpenSocial applications
- Introducing Apache Shindig
- How to develop with Shindig
- Integrating external data with Shindig
- Putting it all together
- RealWorld experience with Shindig - hi5

Putting it All Together - Shindig at hi5

- History
  - Started work January, 2008
  - In Production Since April 1st
  - 100% Deployed April 6th
  - http://developer.hi5.com/
  - 30-40m users per month
- Architecture…
- Lessons Learned…
- Scaling…



Architecture

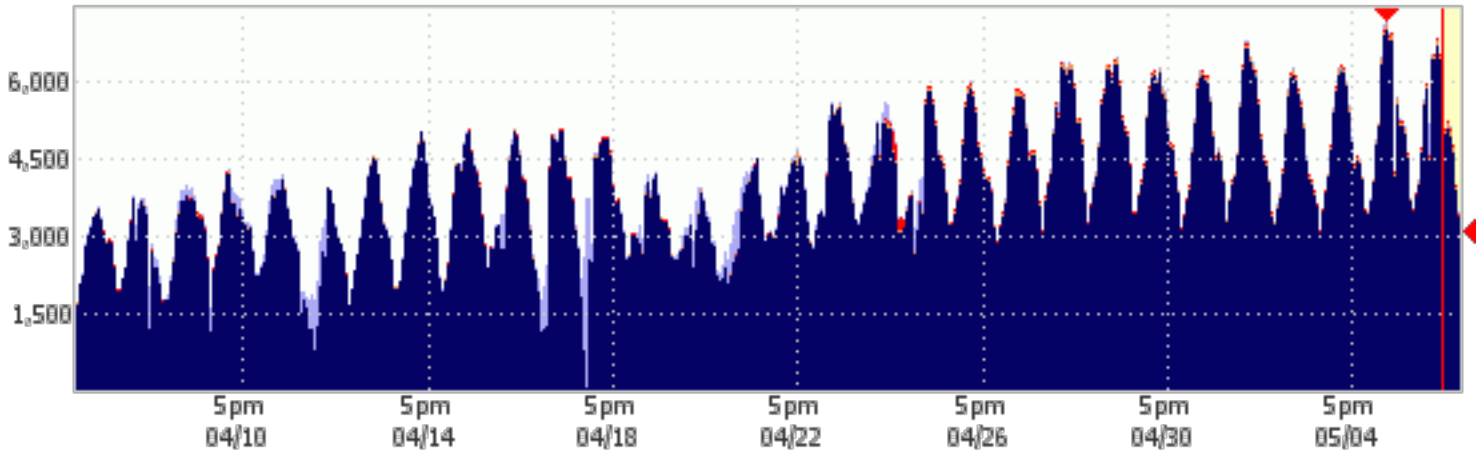


Architecture
Server Internals

- App Servers
  - Apache 2.2
  - Resin 2.1
  - “Old Reliable”
  - Hundreds and Hundreds
- Backends
  - Postgres Databases
  - Memcache, Memcache, Memcache (1TB+ !)
  - Graph Servers
  - Storage, Mail, DNS, etc
- Shindig Servers
  - Tomcat 6.0.16
  - Enunciate API Server
  - 4GB, dual 4-Core Xeon
- 3rd Party App Servers
  - Wide Variety
  - Rails, Python and PHP most popular

Traffic – we have…

- 10k req/sec (Edge)
- 6.5k req/sec (Origin)



Lessons Learned

- Be prepared for Viral Growth
  - We expanded Shindig Servers 50% in the first week
  - Insure that requests are O(1)
- Supporting OpenSocial is a lot more than Shindig
  - Developer Tools for publishing and managing Apps
  - Application Gallery and Metadata Management
  - Notifications Database
  - Monitoring and Controls – Essential!

How to Scale

- Fix Your Concurrency Issues
- Work Around Other People's Concurrency Issues
- Target the requests on the Critical Path
  - Friend/Activities/Data used all the time
  - Must be fast, this is Ajax
- Factor out the Disk
  - Server instances should never swap
  - Database usage should be minimal, if at all – use pre-populated caches.
- Cache, Cache, Cache
  - In the CDN, in the Browser, Everywhere

How to Scale – JVM

- Fast, Consistent Response Time is Required
- Use latest Java 1.6 beta 10
- ParNew and CMS are your friend
  - `-XX:+DisableExplicitGC`
  - `-XX:+UseParNewGC`
  - `-XX:+UseConcMarkSweepGC`
  - `-XX:+AggressiveOpts`
  - `-XX:+DoEscapeAnalysis`
  - `-XX:+PrintGCDetails`
  - `-Xss320k -Xms2300m -Xmx2300m`

How to Scale – Concurrency

- High CPU + spin locks == slow
- How to Fix? – Fire Drills.
  - Push all traffic to a single host
  - Use Performance Monitoring tools
  - Or just do lots of Stack Traces
  - Remove Concurrency Problem
  - … repeat …

How to Scale – Gotchas

- JCE Crypto Providers
  - Use ThreadLocal Cipher Cache
- Character Sets
  - Use more than 2? You die…
  - "String".getBytes() ends up being synchronized.
  - UrlConnection will request charsets utf-8 and UTF-8!
  - Make static specialized CharsetEncoders
- Log4j shared data structures
  - Just reduce logging
- synchronizedMap(Hashmap) bad
  - ConcurrentHashMap good!

Results

- Fast
  - Cached at the Edge
  - Scales with more hosts/cores
- Cheap
  - All commodity hardware
  - Open Source
- Good
  - Quality Apps that our Users love
  - 10+ billion req/month
  - 85+ TB of data used

THANK YOU

How to Implement your own OpenSocial Container on the Java Platform

Chris Schalk, Paul Lindner

TS-6574