

Index

A

- Accelerators, Joyent**, 298
- access, defined (server-side programming)**, 173
- Access Key Identifiers**, 295
- action requests**, 232–233
- activities**, 130–139
 - activity data, 207–208
 - activity posting, 134–137
 - creating, 135–136
 - invitations to install, 137–139
 - media items in, 254–258
 - messages, 131–134
 - REST requests and, 202
- adjustHeight method**, 104, 106
- Adobe Express Install application**, 273
- advertising**, 318–320
- Akamai**, 290
- Amazon Machine Image (AMI)**, 293
- Amazon Simple Storage Server (S3)**, 296
- Amazon Web Services (AWS)**
 - Amazon Simple Storage Server (S3), 296
 - basics, 295
 - CloudFront, 296–297
 - defined, 292–293
 - EBS, 296
 - Elastic Compute Cloud (EC2), 293–295
 - grid/cluster computing, 296
 - Joyent, 298
 - pricing, 297–298
- analytics**, 222, 312–316
- AppData**
 - clearing, 258–259
 - data support, 208–210
 - requests, 202
- appearance, of applications**, 17–20
- application data, defined**, 46
- application infrastructures**, 304
- application servers**, 283–284
- Application/Client layer**, 281
- applications. See also specific applications**
 - appearance of, 17–20
 - brand-based, 39
 - communication and, 31. *See also* communications applications
 - control of. *See* control of applications
 - creating, 83–86
 - finding, 15–16
 - friend selector, 241–243
 - goals of. *See* goals of applications
 - identity of social networks, 48–50
 - improvement of, 53
 - installing, 16–17
 - JavaScript tools for, 274–275
 - making social and viral, 29
 - marketing, 50–53
 - person/people. *See* person/people applications
 - publishing (Zembyl), 373
 - retaining users, 53
 - size for different views, 19–20
 - social network applications, defined, 14
 - testing, 78, 79, 221–222
 - tips for development of, 53–54
 - trends in. *See* trends in applications
 - viral channels and features, 46–48
- Application Builder interface (MySpace)**, 69–71
- Architectural Styles and the Design of Network-Based Software Architectures**, 169, 172
- architecture**
 - OpenSocial RESTful API, 175–176
 - REST, 170
 - RESTful/RESTful-RPC hybrid, 171
 - Web system, 280–282
 - YIOS, 354–355
- architecture of OpenSocial**, 57–64
 - basics, 57–58, 63–64
 - client-based API, 58–59
 - server-based API, 58, 59–62
- asynchronicity (OpenSocial JavaScript API)**, 114, 116
- Atom**, 66
- attentiveness of application users**, 31
- attributes. See also specific attributes**
 - Content element, 93–94
 - ModulePrefs element, 88–91
 - UserPrefs element, 91–93
- authentication. See also OAuth (OpenSocial REST authorization and authentication)**
 - defined, 176
 - RESTful architecture and, 175
- authorization. See also OAuth (OpenSocial REST authorization and authentication)**
 - defined, 176
 - RESTful architecture and, 175
- authz**, 232
- AWS. See Amazon Web Services (AWS)**

Base URL, calls and

B

Base URL, calls and, 200

batching requests

basics, 63–64

in code for Friend Finder, 114

multiple requests, 306–307

before attribute, 347–348

blog services, 52

blurbs, creating, 140–144

brand-based applications, 39

browsing indexes (hi5), 15–16

bug-tracking software, 317

build communities, defined, 1

bundles, message, 250–254

buttons for email creation,

131–132

C

Cache-Control HTTP header, 286

caching

basics, 284–289

issues, 228–229

vs. requests for external files, 309

CAJA, 222, 358–359

calls

Base URL and, 200

calling templates, 331–332

to external resources, 227–228

Gadget API, 63

opensocial.getEnvironment() method call,
259

os.data.resendRequest ('key') method call,
349–350

signed makeRequest calls, 230

YQL, 365

Canvas view

application with contents for, 107

basics, 19, 22

CanvasWidget, 371, 372

case studies

Jambool, 299–300, 322

KlickNation, 299, 322

monetization, 321–322

RockYou, 299, 322

ShopIt, 322

Slide, 300, 322

social network architecture, 299–300

Watercooler, 300, 322

CDNs (Content Delivery Networks), 290

certificate caches, 231

check_signature method (OAuth), 188

Chou, Kevin, 173, 300

Cisco Content Delivery Engine/System, 287

client code performance monitoring, 309–311

client libraries, 353

Client URL library (curl), 171, 172

client-based API (OpenSocial), 58–59

cloning, 302

cloud computing, 291

CloudFront, 296–297

code

hosting on MySpace, 69–71

performance monitoring, 309–311

for signed requests, 231

troubleshooting, 119

uploading from external servers,
71–74

Coderunner, 274–275

collections, defined, 65

collocated hosting, 291

communication engagement, 31

communications applications, 246–258

gadget message bundles, creating,
250–251

media items in activities, 254–258

message and activity templates,
251–254

message summaries, 254

minimessages, creating, 249–250

signed requests, creating, 247–249

configuring Flash media, 273–274

consumers, defined (OAuth), 176

containers

application testing and, 221

compliance, 262–263

defined, 55

inquiring about capabilities on, 232

signed requests and, 231

support of OpenSocial, 56–57

container-specific extensions, 264–266

Content Delivery Networks (CDNs),
290

Content element, 93–94

control of applications, 25–29

developer control, 29

network control, 25–27

user control, 28

Core Gadget API, 95–102

cost calculator (Amazon), 298

Cost per Action (CPA), 319

CSS

image sprites, 312

style sheets in headers, 308

curl. See Client URL library (curl)

currency, virtual, 320–321

custom tags (OSML), 343

D

data

accessing with tags (OSML), 343

including encrypted, 229

passing to templates, 330–331

support, 203

data (OpenSocial REST API)

activity, 207–208

AppData, 208–210

group, 207

messaging, 210

person, 204
retrieving and storing, 202

data formats (OpenSocial), 65–66

data formatting (OpenSocial REST API), 203–210

- activity data, 207–208
- AppData data, 208–210
- group data, 207
- messaging data, 210
- person data, 204–207

data pipelining, 345–350

- data, getting via markup, 345
- data access, 346–347
- data conditional rendering, 347–348
- dynamic request for data, 349–350
- listener for data changes, 348–349

databases

- database sharding, 301–302
- design, 300–302
- distributed, 301

DataContext objects, 346–347

DataRequest extension, 266

DataRequest object, 306

debugging

- CAJA and, 359–360
- errors value and, 228
- Firebug and, 303, 360
- getErrorMessage() method and, 54, 262

definition files for templates, 333–334

delay, defined, 278

deployment, 67–83

- Freebar, 83
- Friendster, 80–83
- hi5, 74–75
- imeem, 78–80
- MySpace. *See* MySpace deployment
- Netlog, 78
- OpenSocial REST API, 212–213
- orkut, 75–78

destination applications, 39

developer dashboard (imeem), 78–79

developers

- control of applications, 29
- signed requests and, 230
- tips for developing applications, 53–54

development

- YAP, 360–361
- YAP OpenSocial applications, 361–364

discovery

- OpenSocial RESTful API, 213–216
- REST applications and, 175, 176

dismissible minimessage, 249–250

distributed cache, 287–289

distributed database systems, 300

distribution, defined, 31

DNS lookups, reducing number of, 307

Dojo Toolkit, 274

downloads. *See also* Web sites for downloading

- parallel, 309

dynamic evolution of applications, 32

dynamic requests for data, 349–350

E

EC2 (Elastic Compute Cloud), 293–295

edge servers, 290, 296

efficiency, defined, 278

Elastic Block Store (EBS), 296

Elastic Compute Cloud (EC2), 293–295

elements. *See also specific elements; sub-elements*

- Atom, 66
- Gadget XML, 87–95
- XML, 66
- XRDS Simple documents, 214

email

- example, 131–133
- services for marketing, 52

embedFlash method, 272

encrypted data, including, 229

engagement of application users, 30–32

environment (support and domain), 259–261

Environment class, 259

error codes

- HTTP, 213
- OpenSocial, 263

error handling, 262–263

errors (HTTP), 195, 213

“Evaluating the Scalability of Distributed Systems”, 278

examples

- activities, creating, 135–136
- blurbs, creating, 140–144
- buttons, to trigger email creation, 131–132
- email messages, 131–133
- Info, 122
- notification, 133–134
- OAuth in PHP, 190–195
- people (GetFriends), 119–122
- people (Info), 122
- people (ViewerData), 117–119
- PersonData, 130
- templates, 350–352
- Y!OS, 366–367

Expires HTTP header, 286

exposure of applications, 31

expressions

- defined, 329
- templates and, 329–330

Extensible Resource Descriptor Sequence (XRDS)

- RESTful architecture and, 176
- XRDS Simple, 213–216

extensions, container-specific, 264–266

external resources, 226–232

- caching issues, 228–229
- calls to, 227–228
- POST request, 229
- preload, 232
- signed request, 230–231

Facebook

F

Facebook

- application trends and, 34–35
- gifting feature, 45
- programming, 174–175

fans

- fan accounts, 317
- fan applications, 36–38

Feature-Specific Gadget API, 102–107

Fielding, Roy, 169, 172

fields

- individual person fields, 205
- to request about people, 123–125
- subfields for organizations, 207
- subfields for person fields, 206
- YAP supported, 361–362

fields extension, 266

files, reducing number of, 307

Firebug, 303, 360

Flash media, 271–275

Flixster hypothetical application, 60–62

flushing server responses, 309

foreign-based social networks, 9–10

forward proxy cache, 286

Freebar

- appearance of, 24
- basics, 9–11
- deployment, 83

Friend Blurb application, 140–143

Friend Finder

- batching requests, 114
- code for, 112–113
- on hi5, 75, 76
- makeRequest method, 227–228
- on MySpace, 74
- navigation tabs, 222–224
- on orkut, 77
- styled tabs, creating, 226

friends

- defined, 116
- friend selector application, 241–243
- of friends, finding, 246
- GetFriends example, 119–122
- list of, paginating, 238–239
- requesting, 236–238
- testing if users are, 243–244
- top friends, finding, 244–246

Friendster

- appearance of, 23–24
- basics, 8–9
- deployment, 80–83
- styled tabs, creating, 226
- traffic on, 48–50

Full view (YAP), 358, 359, 366, 367, 368

future tags (OSML), 339–340

G

Gadget API

- calls, 63
- core gadget API, 95–102
- Feature-Specific Gadget API, 102–107
- using, 271–272

gadget message bundles, creating, 250–251

Gadget XML, 87–95

- application stripped of, 362–363
- defined, 87
- elements and sub-elements, 87–95
- file for Friend Finder, 112
- OpenSocial JavaScript applications and, 87
- preload element, 232
- specifications, 71–72

gadgets

- basics, 87
- defined, 59
- lifecycle support, 109–110
- multiple views, 107–109

Gadgets i18n tool, 270

gadgets.flash object, 102, 106, 272

gadgets.io, 95–97

gadgets.io.getproxyUrl method, 312

gadgets.json, 95, 97

gadgets.Minimessage Gadget API class, 249

gadgets.Minimessage.dismissMessage application, 250

gadgets.Prefs, 95, 100

gadgets.skinn object, 102, 105–106

gadgets.util, 95, 97–100

gadgets.util.hasFeature method, 232

gadgets.views object, 102–104

gadgets.window object, 102, 104–105

gender-specific pronouns, creating, 240–241

general appeal applications, 35–36

GET method, 229

getDomain method, 259

getErrorMessage method, 263

GetFriends, people example, 119–122

globalization, 267–271

goals of applications, 29–33

- dynamic evolution, 32
- engagement, 30–32
- growth, 30
- look and feel, 32
- problem solving, 33
- relationship building, 33
- self expression, 32
- social exposure, 33

goods, virtual, 320–321

Google

- gadget documentation, 87
- Google Analytics, 313–315
- Google App Engine, 299
- Google Gadget XML. *See* Gadget XML

greetings message, 251

grid computing, 291

grid/cluster computing, 296

grouping requests, 63–64

groups

data groups, support of, 207
requests and, 201

growth of application use, 30**GUI design, 222–226**

look and feel, 224–225
navigation tabs, 222–224
social-network specific looks, 225–226

{**guid**}, 200

Guptas, Vikas, 299

GZIP, compressing content with, 308

H

hadError() method, 263

Hadoop, 296

Hansson, David Heinemeier, 171

HAS_APP, 245–246

headers

CSS style sheets in, 308
HTTP, cache and, 285–286

height of applications, adjusting, 106–107

Hello World

with Core Gadget API, 101
height, adjusting, 106–107
including lifecycle support, 110
with ModulePrefs, 94–95
OpenSocial, 64–65
template, constructing, 326–329
template that uses passed parameters, 331

hi5

appearance of, 22
basics, 5–7
browsing index, 15–16
deployment, 74–75
lifecycle extension, 265–266
OAuth and, 196–197
support for OpenSocial REST, 196–198
template library, 225, 266
traffic on, 48–50

history of OpenSocial, 55–56

Home view, 19, 20–21

horizontal partitioning (distributed databases), 301

horizontal scaling, 280

hosting, on MySpace, 69–71

hosting solutions

AWS. *See* Amazon Web Services (AWS)
developer comments/trends, 292
Google App Engine, 299
types of, 290–292

Hot Stocks YAP application, 366, 367

HTML. *See* HyperText Markup Language (HTML)

HTTP. *See* HyperText Transfer Protocol (HTTP)

hybrid applications

client/server OpenSocial, 61–62
OpenSocial JavaScript and RESTful API, 175

Hyper Text Caching Protocol (HTCP), 287

HyperText Markup Language (HTML)

as alternative to media items, 256–258
HTML validators, 222

HyperText Transfer Protocol (HTTP)

errors (OAuth), 195
headers, cache and, 285–286
method type, 210–211
OAuth request parameters and, 180
OAuth tokens and, 181
requests, minimizing number of, 306
REST and, 169, 171, 175
status codes (OpenSocial REST API), 213

I

i18n and L10n, 267

ICP (Internet Cache Protocol), 287

IdSpec, 122–130

IdSpec class, 246
PersonData example, 130
social data, application requesting, 126–129
social data, fields to request, 123–125

if attribute, 332

IFrames, hybrid applications and, 62

I-Jet Media Inc., 322

iLike

length of engagement and, 39–40
on MySpace, 18

image spriting, 312

imeem

appearance of, 24
basics, 9
deployment, 78–80

Info example, 122

information

retrieving, 140–144
storing, 140

Infrastructure layer, 281

init() function, user languages and, 100–101

init() method, 113–114

installing

applications, 16–17
invitations to install, 137–139

instance types (EC2), 293

internationalization

basics, 267–271
marketing applications and, 41–42
OSML and, 344

Internet Cache Protocol (ICP), 287

Internet Security and Acceleration (ISA) server, 287

invitations to install, 137–139

issue-tracking software, 317

iWidgets, 367–368

J

Jambool case study, 299–300, 322

JavaScript. *See also* OpenSocial JavaScript API

CAJA and, 222
code, for Friend Finder, 112–113
code, locating at bottom, 309
minifying, 308
SWFObject JavaScript library, 272–273

JavaScript (continued)

JavaScript (continued)

- tools for applications, 274–275
 - JavaScript Object Notation (JSON)
 - basics of, 65–66
 - REST responses and, 211–212
- Jogalekar, Prasad**, 278
- Joyent**, 298
- JSMin filtering program**, 308
- JSON**. *See* JavaScript Object Notation (JSON)

K

- “Keep Alive On” HTTP feature, 307
- key cache (OAuth)**, 195
- key chains (Zembyl)**, 371
- Kiss**
 - internationalization and, 42
 - search on, 17
- KlickNation case study**, 299, 322
- Kostarev, Alexey**, 45

L

- languages (computer)**. *See also* JavaScript; OpenSocial Markup Language (OSML)
 - language support (OAuth), 196
 - work in language of choice, defined, 173
 - YML, 365–366
- languages (spoken)**
 - tag for Spanish messages, 344
 - user languages, 100–102
- Levchin, Max**, 322
- Leymann, F.**, 172
- libraries**
 - client, 353
 - curl, 171, 172
 - hi5 template, 225, 266
 - OAuth, 196
 - SWFObject JavaScript, 272–273
- lifecycles, support for**, 109–110
- lifecycle events**, 109
- lighttpd**, 289
- listen attribute**, 348
- listener, for data changes**, 348–349
- load balancing**, 284
- loading times**, 312
- localization**
 - basics, 267–271
 - defined, 332
 - with templates, 332
- longer engagement applications**, 39
- look and feel**
 - of applications, 32
 - GUI design and, 224–225
 - social-network specific looks, 225–226
- looping**
 - repeat attribute for, 332
 - through lists, 345

M

- Maffia new**
 - use of media in, 41, 42
 - virtual currency and, 50–51
- makeRequest method**
 - caching and, 228
 - gadgets.io, 96, 100
 - refresh intervals, setting and, 229
 - simple invocation of, 227–228
- managed dedicated hosting**, 291
- marketing applications**, 50–53
- McNulty, Rhet**, 322
- media**
 - in applications, 39–41
 - HTML as alternative to, 256–258
 - using in activities, 254–258
- meetups**, 304
- Memcached system**, 287–289
- messages**
 - email message example, 131–133
 - gadget message bundles, 250–251
 - message and activity templates, 251–254
 - message bundles, 252–254
 - minimessages, 249–250
 - notification example, 133–134
 - as option, 202–203
 - summaries, 254
 - types of, 131
- messaging data**, 210
- methods**
 - Core Gadget API, 96–100
 - Feature-Specific API, 102–106
 - HTTP (OpenSocial REST requests), 210
 - OAuthRequest class, 182
 - OAuthSignatureMethod_RSA_SHA1 class, 190
 - OpenSocial, 144–167
 - optional parameters, 116
 - Y!OS, 354
- metrics**
 - performance, 278
 - scalability, 279
- micropayments/micro-transactions**, 321
- Microsoft Internet Security and Acceleration (ISA) server**, 287
- micro-transactions, defined**, 45
- Middleware layer**, 281
- minified JavaScript**, 308
- minimessages, creating**, 249–250
- mobile applications**, 45
- ModulePrefs**
 - attributes and sub-elements for, 88–91
 - defined, 87
 - Hello World with, 94–95
- monetization**, 318–322
- monitoring client code performance**, 309–311
- monitoring software**, 302–303
- multiple cache servers**, 287
- multiple content tags**, 108–109
- multiple views (applications)**, 107–109
- <myapp:HelloWorld> tag**, 342

OpenSocial REST API

MySpace

appearance of, 20–22
 application trends and, 34
 Application Builder interface, 69–71
 basics, 2–4
 rules for developers, 27
 traffic on, 48–50

MySpace deployment, 67–74

code, uploading from external servers,
 71–74
 getting started, 67–69
 hosting on MySpace, 69–71

N

`<namespace:templateName/>` tag, 342

naming

OpenSocial API and, 116
 templates, 328–329

navigation tabs, 222–224**Netlog**

appearance of, 24–25
 basics, 11
 deployment, 78

network control, 25–27**new App Install program**, 51**newFetchPersonAppDataRequest method**, 140**newUpdatePersonAppDataRequest method**,
 140, 143**Nginx**, 289**non-RESTful**, defined, 169**non-social networking applications**, 59–62**NOT_IMPLEMENTED error code**, 262–263**nothing shared cloning**, 302**notification example**, 133–134**O****OAuth (OpenSocial REST authorization and authentication)**, 176–198

basics, 176–177
 hi5 authentication scheme, 196–198
 HTTP errors, 195
 incorporating into OpenSocial REST API, 196
 libraries, 196
 OAuthSignatureMethod_RSA_SHA1, 189
 OpenID, 195
 parameters, 179
 in PHP. *See* OAuth in PHP
 requests, 179–180
 signing requests, 180
 simple PHP test program, 191–194
 steps, 177–178
 tokens, 181, 195

OAuth in PHP, 181–195

example, 190–195
 OAuthConsumer, 181
 OAuthRequest, 182–188
 OAuthSignatureMethod, 188–190
 OAuthToken, 181–182

objects

Core Gadget API, 95
 Feature-Specific API objects, 102–106
 OpenSocial, 144–167
 received by callback function, 228

Offerpal, 320**onLoadViewerResponse method**, 114**Open Group Architectural Framework (TOGAF) Method**,
 282**open sandbox access**, defined, 53**OpenID (OAuth)**, 195**OpenSocial**

AppData as container cache, 307
 application sample, 64–65
 applications, creating, 83–86
 architecture. *See* architecture of OpenSocial
 data formats, 65–66
 deployment. *See* deployment; MySpace deployment
 error codes, 263
 get from cache, 312
 history, 55–56
 introduction to, 1, 55
 object, 144–146
 performance. *See* performance
 programming. *See* OpenSocial programming
 supporting networks, 14
 tools, 54

OpenSocial JavaScript API, 111–168

activities. *See* activities
 architecture, 63–64
 basics, 58, 61
 client-based API and, 58–59
 details, 144–167
 features, 115–116
 Gadget XML and, 87
 people. *See* people
 persistence. *See* persistence
 simple OpenSocial application, 111–115

OpenSocial Markup Language (OSML), 334–345

basics, 334–335
 custom tags, 343
 data access with tags, 343
 future tags, 339–340
 internationalization and, 344
 looping through lists, 345
 partial content, rendering, 344
 proposed tags, 341–342
 specifying as required feature, 342
 standard tags, 335–338

OpenSocial programming, 221–233

action requests, 232–233
 application testing, 221–222
 external resources. *See* external resources
 GUI design. *See* GUI design
 inquiring about capabilities, 232

OpenSocial REST API

data formatting. *See* data formatting (OpenSocial REST API)

deployment, 212–213

discovery, 213–216

future versions of, 216

OpenSocial REST API (continued)

OpenSocial REST API (continued)

- HTTP status codes, 213
- incorporating OAuth into, 196
- vs. RPC protocol, 216–217
- security, 216
- specification, 200–203
- using, 198, 212

OpenSocial RESTful API, 169–219

- architecture, 175–176
- authorization and authentication. *See* OAuth (OpenSocial REST authorization and authentication)
- basics, 60–62
- data formatting. *See* data formatting (OpenSocial REST API)
- HTTP method type, 210–211
- OpenSocial REST API specification, 200–203
- responses, 211–212
- REST basics, 169–172
- REST request construction, 198–199
- RPC protocol and, 216–219
- server-side programming, 173–175

OpenSocial Tool/Harness, 222

- opensocial.Activity class**, 134
- opensocial.Activity object**, 146–147
- opensocial.Activity.Field object**, 147–148
- opensocial.Address object**, 148
- opensocial.Address.Field object**, 148–149
- opensocial.BodyType object**, 149
- opensocial.BodyType.Field object**, 149
- opensocial.Collection object**, 149–150
- opensocial.CreateActivityPriority object**, 150
- opensocial.DataRequest object**, 150–152
- opensocial.DataRequest.DataRequestFields object**, 152
- opensocial.DataRequest.FilterType object**, 152–153
- opensocial.DataRequest.newUpdatePersonAppData Request method**, 258
- opensocial.DataRequest.PeopleRequestFields object**, 153
- opensocial.DataRequest.PeopleRequestFields.FILTER key**, 244
- opensocial.DataRequest.PeopleRequestFields.FIRST parameter**, 236
- opensocial.DataRequest.SortOrder object**, 153
- opensocial.DataResponse object**, 153–154
- opensocial.Email object**, 154
- opensocial.Email.Field object**, 154
- opensocial.Enum object**, 154
- opensocial.Enum.Drinker object**, 154–155
- opensocial.Enum.Gender object**, 155
- opensocial.Enum.LookingFor object**, 155
- opensocial.Enum.Presence object**, 155–156
- opensocial.Environment object**, 156
- opensocial.Environment.ObjectType object**, 157
- opensocial.EscapeType object**, 157
- opensocial.getEnvironment() method call**, 259
- opensocial.IdSpec object**, 157–158
- opensocial.IdSpec.Field object**, 158
- opensocial.IdSpec.Field.NETWORK_DISTANCE field**, 246
- opensocial.IdSpec.PersonId object**, 158
- opensocial.Medialtem object**, 158
- opensocial.Medialtem.Field object**, 159

- opensocial.Medialtem.Type object**, 159
- opensocial.Message object**, 159
- opensocial.Message.Field object**, 159–160
- opensocial.Message.Type object**, 160
- opensocial.Name.Field object**, 160
- opensocial.NavigationParameters object**, 160–161
- opensocial.NavigationParameters.DestinationType object**, 161
- opensocial.Organization object**, 161
- opensocial.Organization.Field object**, 162
- opensocial.Permission object**, 162
- opensocial.Person object**, 162–163
- opensocial.Person.Field object**, 163–166
- opensocial.Phone object**, 166
- opensocial.Phone.Field object**, 166
- opensocial.ResponseItem object**, 166–167
- opensocial.ResponseItem.Error object**, 167
- opensocial.template.renderAll Templates**, 328
- opensocial.Url object**, 167
- opensocial.Url.Field object**, 167
- optional parameters (OpenSocial methods)**, 116
- ordered lists, defined**, 65
- orkut**
 - appearance of, 23
 - basics, 7
 - deployment, 75–78
- os.data.resendRequest ('key') method call**, 349–350
- os.getTemplate function**, 328–329
- <os:Locale> tag**, 344
- OSML. *See* OpenSocial Markup Language (OSML)**
- <os:OwnerRequest> tag**, 345
- <os:Render> tag**, 344
- <os:Repeat> tag**, 345
- owned/proprietary system solution**, 291
- owners**
 - defined, 116
 - email message from viewers, 132–133

P

- parallel downloads**, 309
- parameters**
 - calling templates with, 331–332
 - OAuth, 179, 199
 - OAuthSignatureMethod, 188
 - opensocial.Message.Field.*, 131
 - opensocial.newActivity, 134
 - optional (OpenSocial methods), 116
 - request parameters for user validation, 231
- partitioning, defined**, 284
- partnering**, 43, 320
- Pautasso, C**, 172
- people**, 116–130
 - basics, 116–117
 - data, OpenSocial REST support of, 204–207
 - GetFriends example, 119–122
 - getting information about, 117
 - IdSpec. *See* IdSpec
 - Info example, 122
 - OpenSocial REST requests and, 200–201

- types of, 116
 - ViewerData example, 117–119
 - percent encoding (OAuth), 179**
 - performance**
 - defined (server-side programming), 173
 - improving with preload, 232
 - metrics, 278
 - monitoring client code, 309–311
 - problem areas, 280
 - performance tuning**
 - AppData as container cache, 307
 - batching multiple requests, 306–307
 - caching, 309
 - client code performance, monitoring, 309–311
 - CSS style sheets in headers, 308
 - DNS lookups, reducing number of, 307
 - files, reducing number of, 307
 - flushing server responses, 309
 - GZIP and, 308
 - HTTP requests and, 306
 - image sprites, 312
 - JavaScript, minifying, 308
 - JavaScript code, locating at bottom, 309
 - loading times, 312
 - OpenSocial get from cache, 312
 - overview, 306
 - persistence feature in Web servers, 307
 - preloading content, 311
 - requests for external files, 309
 - permissions**
 - asking for, 263–264
 - checking, 263–264
 - platforms for, 356, 357
 - programming and, 232–233
 - persistence, 139–144**
 - defined, 139
 - information, retrieving, 140–144
 - information, storing, 140
 - in Web servers, 307
 - personas, creating, 304**
 - PersonData example, 130**
 - person/people applications, 235–246**
 - friend selector application, 241–243
 - friends list, paginating, 238–239
 - friends of friends, finding, 246
 - multiple requests for friends, 236–237
 - pronouns, creating gender-specific, 240–241
 - requesting friends, 236, 237–238
 - testing users as friends, 243–244
 - top friends, finding, 244–246
 - PHP. See also OAuth in PHP**
 - accepting/verifying signed requests, 247–248
 - REST program, 172
 - platforms, 2–14**
 - Freebar, 9–11
 - Friendster, 8–9
 - hi5, 5–7
 - imeem, 9
 - MySpace, 2–4
 - Netlog, 11
 - orkut, 7
 - other networks, 14
 - for setting update permissions, 356, 357
 - Yahoo!, 12–14
 - Y!OS architecture and, 354–355
 - YQL and, 364
 - POST requests, 229**
 - posting of activities, 134–137**
 - pre-fetching, defined, 285**
 - preloading**
 - content, 311
 - improving performance with, 232
 - Presentation layers, 281**
 - pricing, Amazon, 297–298**
 - “Principled Design of the Modern Web Architecture”, 172**
 - problem of rebalancing, 302**
 - productivity, 279**
 - Profile view**
 - application with contents for, 107
 - basics, 19, 22
 - programming. See OpenSocial programming; social network programming**
 - pronouns, creating gender-specific, 240–241**
 - properties**
 - OAuth, 176
 - REST architecture, 170
 - proposed tags (OSML), 341–342**
 - proxied content, 352–353**
 - proxy servers, 286**
 - publishing applications**
 - defined, 81
 - Freebar, 83
 - Zemby, 373
- ## Q
- quality of service, defined, 278**
 - query parameters (OAuth), 199**
- ## R
- Rabois, Keith, 36**
 - random selection algorithm, 284**
 - reach applications, 35–36**
 - real goods, 321**
 - redundancy, 302**
 - refresh intervals, setting, 229**
 - registerOnLoadHandler method (gadgets.io), 99, 100**
 - relationship building, 33**
 - repeat attribute for looping, 332**
 - replication, 301**
 - Representational State Transfer (REST). See REST**
 - requestPermission method, 264**
 - requests**
 - dynamic, for data, 349–350
 - for HTML resources (PHP REST), 172
 - OAuth, 179–180
 - OAuthRequest, 182–188
 - OAuthToken, 181–182

requests (continued)

requests (continued)

- rendering templates, 328
- REST request construction, 198–199
- require attribute**, 347
- Require feature**, 342
- resources**. *See also* Web sites for further information
 - REST, 170, 172
 - templates, 351–352
 - XRDS Simple, 216
- response time, defined**, 278
- responses**
 - OAuth, 181
 - OpenSocial REST requests and, 211–212
- REST**. *See also* OpenSocial REST API; OpenSocial RESTful API
 - basics, 169–172
 - non-RESTful, defined, 169
- “REST API Specific URL Pattern”, 200
- RESTful API**. *See* OpenSocial RESTful API
- RESTful Web Services (O’Reilly Media, 2007)**, 171
- “RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision”, 172
- RESTful-RPC hybrid architecture**, 171
- retrieving data**, 140–144, 202
- reverse proxy servers**, 286–287
- Richardson, Leonard**, 171
- RockYou**
 - case study, 299, 322
 - self expression and, 43
 - slideshow application, 174
- role playing**, 304
- round robin rotation**, 284
- RPC protocol**
 - vs. OpenSocial REST API, 216–219
 - RESTful/RESTful-RPC hybrid, 171
- Ruby, Sam**, 171

S

S3

- Amazon Simple Storage Server, 296
- S3Firefox Organizer, 296

sandboxes

- open sandbox access, 53
- running applications in, defined, 9

scalability

- defining, 278–279

- metrics, 279

- scalable user interface design, 316–317

- scaling up/out, 280

- <script> tag, 347, 348

- security (OpenSocial REST API), 216

- selector, defined (OpenSocial REST API), 200

self expression

- applications and, 43
- in social networking, 32

Send Good Karma

- jPoints, 44
- partnering and, 43, 44, 320
- virtual goods and, 320

- serial order for request processing, 64

server-based API

- OpenSocial, 58, 59–62
- OpenSocial RESTful API, 173–175

servers

- flushing server responses, 309
- multiple cache, 287
- persistence feature in Web servers, 307
- proxy, 286
- reverse proxy, 286–287
- S3, 296

- server-side code for signed requests, 231

- server-side programming (OpenSocial RESTful API), 173–175

services

- for blogs, 52
- defined, 370
- for email marketing, 52
- finding and sharing, 373
- Zemby, 370–374

- sharding, database, 301–302

- Share Good Karma, 52

- shared disk cloning, 302

- shared hosting, 290–291

- Shen, Jia, 299

- Shindig, 57

- ShopIt case study, 322

- Shukla, Anu, 320

- signatures, checking, 188–190

- signed requests, 230–231

- signed requests, creating, 247–249

- signing requests (OAuth), 180

- “Silly is Serious Business”, 36

- Simple Storage Server, Amazon (S3), 296

- Slide case study, 300, 322

- Small view (YAP), 358

- Smith, Andy, 181

- snippets (Zemby), 371

social data

- application requesting, 126–129
- fields requesting, 123–125
- use of in applications, 45–46

- social exposure, application growth and, 33

- social gadgets, 64

- social hooks, 46–48

social network programming

- applications. *See* applications
- identity of social networks, 48–50
- platforms. *See* platforms
- retaining users, 53

social networking

- basics, 1–2
- foreign-based networks, 9–10

- social network-provided analytic tools, 316

software

- analytic, 313
- bug-tracking, 317
- issue-tracking, 317
- for monitoring, 302–303
- version control, 305

uploading code from external servers

software design

- language choices, 305
- overview, 303–304
- versioning and, 305

Sometrics, 316

specifications

- Google Gadget XML, 71–72
- OpenSocial REST API, 200–203

Squid server, 287

standard tags (OSML), 335–338

storing data, 140–143, 202

styles for social networks, 225

subclasses of OAuthSignatureMethod Class, 189

sub-elements, 271–272

- Content, 93–94
- ModulePrefs, 88–91
- OpenSocial REST response, 211
- UserPrefs, 91–93

subfields

- for organizations, 207
- for person fields, 206

subsystems

- application servers, 283–284
- caching, 284–289
- CDNs, 290
- load balancing, 284
- Web servers, 283

supportsField method, 259, 261

SWFObject JavaScript library, 272–273

swfUrl variable, 272

system support, 317–318

T

tables, social data displayed as, 126–130

tabs

- navigation, 222–224
- styled, 226

tag="namespace:templateNameX" parameter, 343

tags

- accessing data with (OSML), 343
- adding (YML), 366
- custom (OSML), 343
- to define Templates XML file, 333
- future (OSML), 339–340
- multiple content, 108–109
- proposed (OSML), 341–342
- standard (OSML), 335–338

targeted applications, 36

targeted social networks, defined, 9

Taylor, Richard, 172

template-based application development, 36–38

template.RenderInto function, 328–329

templates

- hi5 template library, 225, 266
- message and activity, 251–254

Templates Standard, 325–352

- basics, 325–326
- calling with parameters, 331–332
- conditional tests, 332

construction and use, 327–328

data pipelining. *See* data pipelining

definition files, 333–334

examples, 350–352

expressions and, 329–330

localization with, 332

naming templates, 328–329

OSML. *See* OpenSocial Markup Language (OSML)

repeat attribute for looping, 332

requiring features, 326

variables and passing data, 330–331

testing

applications, 221–222

GetFriends example, 121

tags in templates, 332

Theurer, Tenni, 307

three-layer architecture, 281

throughput, defined, 278

tokens (OAuth)

management of, 195

OAuthToken, 181–182

the response, 181

tools

analytic, 316

Dojo Toolkit, 274

Gadgets i18n, 270

JavaScript, 274–275

OpenSocial, 54

OpenSocial Tool/Harness, 222

for testing code, 222

UML, 304

trends in applications, 33–46

application data and, 46

basics, 33–35

brand-based applications, 39

destination applications, 39

internationalization, 41–42

longer engagement applications, 39

mobile applications, 45

partnering, 43

reach applications, 35–36

self expression, 43

social data and, 45–46

template-based development, 36–38

use of media, 39–41

vertical applications, 36

virtual currencies, goods and points, 43–45

U

Unified Modeling Language (UML) tools, 304

unmanaged leased dedicated hosting, 291

updates

Gadget API updates, 95

OpenSocial, 134–137

update permissions, 356, 357

Yahoo! users, 355

Y!OS users, 355–356

uploading code from external servers,

71–74

URL patterns (OpenSocial REST API)

URL patterns (OpenSocial REST API)

- for activities, 202
- for AppData, 203
- for groups, 201, 203
- for messaging, 202, 203
- for people, 201

URLs

- avoiding caching and, 229
- OpenSocial REST, 198, 200
- RESTful-RPC hybrid architecture and, 171

user agent cache, 285

UserPrefs

- attributes and sub-elements for, 91–93
- defined, 87

users

- experience/satisfaction, defined, 278
- request parameters for validating, 231
- retaining, 53
- user control of applications, 28
- user ID (OpenSocial REST API), 200
- user languages, 100–102
- user profiles (Y!os), 355
- user support, 317–318
- user updates (Y!OS), 355–356

V

value delivered, defined, 279

variables

- Core Gadget API, 96–100
- Feature-Specific API, 102–106
- OpenSocial (listed), 144–167
- templates and, 330–331

Varnish Web accelerator, 287

version control software, 305

vertical applications, 36

vertical partitioning (distributed databases), 301

vertical scaling, 280

ViewerData people example, 117–119

viewers

- defined, 116
- email messages from, 132–133

views

- changing dynamically, 109
- defined, 107
- directing content to different, 108–109
- multiple, 107–109
- of OpenSocial applications, 19–20
- YAP, 358

viral applications, 29, 47–48

viral calculator, 51

viral channels, 46–48

virtual currency

- Maffia new and, 50–51
- virtual currencies, goods and points, 43–45, 320–321

virtual goods, 45, 320–321

virtual money, 50–51

W

Walton, Ken, 42, 299

Watercooler

- applications, 36–38
- case study, 300, 322
- server-side programming and, 173

Web accelerator, 287

Web proxy caching servers, 286

Web servers, 283

Web sites, for downloading

- Adobe Express Install application, 273
- Coderunner JavaScript testing application, 274
- Dojo Toolkit, 274
- lighttpd, 289
- Nginx, 289
- Varnish Web accelerator, 287

Web sites for further information, 34

- Amazon Web Services, 295
- analytic software, 313
- blogs, 84
- bug-tracking software, 317
- CAJA, 222, 358
- client libraries, 353
- CloudFront, 297
- curl library, 172
- database sharding, 302
- developer page (MySpace), 68
- extensions for OpenSocial containers, 326
- forum on application policies, 27
- Freebar deployment, 83
- Friendster, 80, 225
- Gadget API updates, 95
- gadget documentation, 87
- Google Analytics, 313
- Google App Engine, 299
- Hadoop, 296
- hi5 Analytics, 316
- hi5 developer application, 74–75
- hi5 shortcuts to social data, 225
- hi5 translation service, 270
- HTCP, 287
- ICP, 287
- imeem deployment, 78
- internationalization, 41
- Internet trends, 33, 34
- issue-tracking software, 317
- JSON, 66
- key cache, 196
- localization resources, 272
- meetups, 304
- Memcached system, 288
- MySpace, 27, 51
- Netlog developer contract, 78
- OAuth libraries, 196
- OpenID, 195
- OpenSocial, 14, 83–85
- OpenSocial Templates library, 328

orkut, 75, 76, 77
 proxied content, 353
 RockYou slideshow, 174
 rules for developers, 27
 S3Firefox Organizer, 296
 services, creating, 371
 "Silly is Serious Business", 36
 software monitoring, 302–303
 Sometrics, 316
 sprites, 312
 SWFObject JavaScript library, 272
 templates, 350, 351–352
 TOGAF Method, 282
 user profiles, 355, 356
 XRDS Simple, 216
 Yahoo! Web Analytics, 315
 YAP, 360, 361
 YML, 365

Web system architecture, 280–283

widgets (Zemby), 371

Woodside, Murray, 278

X

X.509 certificates, 295

XML

elements in XRDS-Simple documents, 214
 Friend Finder navigation tabs and, 223–224
 gadget file for Friend Finder, 112
 OpenSocial and, 66

XRDS. See Extensible Resource Descriptor Sequence (XRDS)

Y

Y!OS. See Yahoo! Open Strategy (Y!OS)

Yahoo!

basics, 12–14
 user profiles, 355
 user updates, 355
 Yahoo! Social Platform, 355

Yahoo! Applications (YAP)

application development, 361–364
 basics, 356–357

CAJA, using, 358–359

development steps, 360–361

views, 358

Yahoo! Markup Language (YML),
 365–366

Yahoo! Open Strategy (Y!OS), 353–367

architecture, 354–355

examples, 366–367

methods, 354

OpenSocial and, 353

user profiles, 355

user updates, 355–356

YAP basics, 356–360

YAP development steps, 360–361

YAP OpenSocial application development,
 361–364

YML, 365–366

YQL, 364–365

YUI, 365

Yahoo! Query Language (YQL), 364–365

Yahoo! User Interface (YUI), 365

Yahoo! Web Analytics, 315

Yahoo!'s America Decides YAP application,
 366, 368

YAP. See Yahoo! Applications (YAP)

yelp, 59

YML (Yahoo! Markup Language),
 365–366

Y!OS. See Yahoo! Open Strategy (Y!OS)

YQL (Yahoo! Query Language), 364–365

YSlow extension, 309–311

YUI (Yahoo! User Interface), 365

Z

Zemby

application structure, 369–370

applications, publishing, 373

applications in, creating, 371–373

basics, 368–369

key chains, 371

services, 370–371, 373–374

snippets, 371

widgets, 371

Zimmermann, O., 172