

# Index

## A

### Action Pack, 25–26

ActionView, 25

**actions, 50, 54. See also controllers, in MVC**

### ActionView, 25

### ActiveRecord, xxi, 25, 28–30

associations, 29

blocks, 190–192

scope of, 191–192

model, MVC, 61–62

patterns, 61–62

### ActiveResource, 25, 70

### Agile software, 55–56

philosophy manifesto, 55

### Agile web Development with Rails (Hansson/Thomas), 22

### AJAX. See Asynchronous Javascript and XML

### Ajax on Rails (Raymond), 102

### aliasing methods, 221–222

### Allen, David, 37

### American Gladiator, 36

### Andreessen, Marc, 9

### AOP. See aspect-oriented programming

### Apache Hypertext Transfer Protocol (HTTP), 118

### API applications, web, xix, 90–113

ActiveRecord, xxi, 25, 28–30

associations, 29

model, MVC, 61–62

AJAX, 155–156

custom MIME type additions, 103–105

creation of, 105

registration, 104

response formats, 103

documentation, 60–61

metering, 105–109

algorithm, 107

filters, 108–109

user authentication, 105–107

method calls, 96–97

components, 96

method\_missing method, 241, 247–248

overlying, 97–103

non-HTML results, 99–100

RDF, 103

response\_to method, 97–99

RSS, 101–103

XML, 100–101

REST, 126–127

routing, 93–96

definition, 94–95

Libs, 94

naming methods, 94

options hash, 95

templates, 95

service, 109–112

definitions, 110

implementation, 111–112

structs, 112

SOAP, 109–110

URLs, 91–93, 95–96

concept v. function for, 92

design, 95–96

development of, 91–92

hierarchy, 93

### Array.each method, 170, 237–240

### artofrails.com, 105, 117, 145

### aspect-oriented programming (AOP), 38

blocks, 187–189

benchmarking, 188–189

### associations, 29

model objects, MVC, 62

polymorphic, 68–70

## Asynchronous Javascript and XML (AJAX)

---

**Asynchronous Javascript and XML (AJAX), xxi, 140–166. See also compiled to web style, AJAX; in-place application style, AJAX; partial style, AJAX; proxy style, AJAX; puppet style, AJAX**

as API, 155–156

design issues, 141–143

development of, 140–141

elegant degradation, 162–163

frameworks, 142–144

Dojo, 144

Ext, 144

jQuery, 144

MooTools, 144

Prototype Javascript library, 142–143

script.aculo.us libraries, 31, 143

Yahoo! UI, 143

Ruby on Rails, 157–162

inner layouts, 159–160

styles, 144–155

compiled-to-web, 151–153

in-place application style, 153–155

partial, 148–149, 157–159, 163–164

proxy, 146–148

puppet, 149–151, 160–162

user interfaces, 164–165

DataTables, 164–165

XMLHttpRequest, 141

**benchmarking, 188–189**

**Berners-Lee, Tim, 5, 8**

**Bina, Eric, 9**

**blocks, 169–195**

ActiveRecord transactions, 190–192

scope of, 191–192

AOP, 187–189

benchmarking, 188–189

Array.each method, 170

callbacks, 194–195

code wrapping, 172

definition, 169–173

dual-use functions, 194

environmental effects, 184–186

filters, 189–190

HTML writing, 192–193

iterations, 186–187

procs and, 180–181

source environment influences, 182

XML, 193–194

**Blogger, 4**

**Brooks, Frederick, 43**

**browsers, 6, 8–9**

Internet Explorer, 9

Mosaic, 8–9

Netscape, 8

**buddy lists, 146, 148**

compiled to web style AJAX, 152

in-place application style AJAX, 154

partial style AJAX, 148

proxy style, 146

puppet style AJAX, 151

## B

**backward compatibility workaround, 120–122**

partial style AJAX, 163–164

**badge summary, 76**

**BDD. See behavior-driven development**

**behavior-driven development (BDD), xxi, 273, 276–296. See also RSpec**

principles, 276–277

RSpec, 279–295

additional features, 295

before/after statements, 287–288

development cycle, 279–280, 288–295

implementing examples, 280–281

matchers, 28–287

writing process, 280–281

## C

**CakePHO for PHP, 22**

**callbacks, 194–195**

**Calliau, Robert, 6**

**CGI. See Common Gateway Interface**

**class Class; end method, 215–216**

**class<<Class method, 218–219**

**class\_eval method, 213–215**

instance\_eval v., 219

**class-level endpoint, 121**

**code view, Ruby on Rails, 25–26**

ActionPack, 25–26

## duck-typed languages

- ActiveRecord, 25
  - ActiveResource, 25
  - code wrapping, 172**
  - code-first development, 13–15**
    - advantages/disadvantages, 14
    - control logic, 15
    - HTML, 14
    - Perl, 13
  - CodeIgniter for PHP, 22**
  - commands, 11**
  - Common Gateway Interface (CGI), 12–18**
    - code-first development, 13–15
      - advantages/disadvantages, 14
      - control logic, 15
      - HTML, 14
      - Perl, 13
    - document-first development, 13, 15–18
      - code juxtaposition, 18
      - comments, 16–17
      - components, 15–16
      - document storage, 16
      - HTML, 17
    - HTML documents, 12–13
      - duplication, 15
    - Perl language, 13
  - compiled to web style, AJAX, 151–153**
    - buddy list, 152
    - GWT, 152
    - meebo, 153
  - constants, Ruby on Rails, 249–250**
  - controllers, in MVC, 47, 50, 53–54, 77–88. See also outsourcing**
    - CRUD, 78–80
      - scaffold generator, 79–80
    - dependency graph, 47
    - operation patterns, 78
    - outsourcing, 83–87
      - data-related operations, 85–87
      - filters, 77, 84–85
      - validations, 87
    - refactoring, 65, 87–88
    - reset\_password method, 65
    - resource, 122
    - two-step actions, 80–83
      - flash variables, 83
      - GET/POST paradigm, 81
      - object displays, 82–83
      - object mutations, 82–83
      - types, 53–54
  - “Create, Read, Update, Delete” (CRUD), 78–80**
    - endpoints, 123
    - HTTP, 119
    - partial style AJAX, 157–159
    - scaffold generator, 79–80
  - CRUD. See “Create, Read, Update, Delete”**
  - CSS, 2, 4**
  - custom getters, 272–273**
  - custom MIME types. See MIME types, custom**
- ## D
- database(s), Rails code, 24**
  - Dean, Jeffrey, 37**
  - deep nesting, 135**
  - define\_method, 228–234**
    - class definitions, 228
    - scope of, 230–232
    - variables, 231
  - DELETE method, 8**
  - Digg, 4**
  - directories, Rails code, 23–24**
  - Disqus, 127–128**
  - Django for Python, 22**
  - document-first development, 13, 15–18**
    - code juxtaposition, 18
    - comments, 16–17
    - components, 15–16
    - document storage, 16
    - HTML, 17
  - documents, on web, 5**
    - document-first development, 13, 15–18
    - Google docs, 2, 12
    - UDIs, 6
  - Dojo, 144**
  - domain-specific language (DSL), 27, 227–228**
  - “Don’t Repeat Yourself” (DRY) philosophy, 128**
  - DRY. See “Don’t Repeat Yourself” philosophy**
  - DSL. See domain-specific language**
  - duck punching. See monkey patching**
  - duck-typed languages, 205**

## E

**elegant degradation, 162–163**

**endpoints, 120–121**

- class-level, 121
- instance-level, 121
- non-REST CRUD, 123

***Enquire Within Upon Everything* (Berners-Lee), 5**

**“Essence and Accidents of Software**

**Engineering” (Brooks), 43**

**eval method, monkey patching, 211–219**

- class Class; end, 215–216
- class<<Class, 218–219
- class\_eval, 213–215
- instance\_eval, 216–218

**Ewing, Patrick, 211**

**Ext (AJAX framework), 144**

**Extensible Markup Language (XML), 100–101.**

**See also Asynchronous Javascript and XML**

- AJAX, 140–166
  - as API, 155–156
  - design issues, 141–143
  - development of, 140–141
  - elegant degradation, 162–163
  - frameworks, 142–144
  - Ruby on Rails, 157–162
  - styles, 144–155
  - user interfaces, 164–165
  - XLMHttpRequest, 141
- blocks, 192–193, 193–194
- RXML, 100–101

## F

**facades, 242–243**

**Facebook, 4, 79, 127–128**

**Fielding, Roy, 118**

**filters, 77**

- blocks, 189–190
- metering, 108–109
- method\_missing method, 243–246
- with outsourcing, 84–85

**flash variables, 66**

- two-step actions, 83

**Flickr, 2, 4, 79, 91, 127–128**

**forms**

- HTTP, 10–12
- partials, 75
  - extended search, 75
  - search, 75

**Fowler, Martin, 61**

**Fulciniti, Alessandro, 39**

## G

**GET method, 7**

- GET/POST paradigm, 81

***Getting Things Done: The Art of Stress* (Allen), 37**

**Ghemawat, Sanjay, 37**

**GMail, 79**

**Google**

- docs, 2, 12
- GWT, 72
- MapReduce, 37–38

**Google Web Toolkit (GWT), 72**

- compiled to web style AJAX, 152

**gorilla patching. See monkey patching**

**graphical user interface (GUI), 49**

**Grosenbach, Geoffrey, 295**

**guerilla patching. See monkey patching**

**GUI. See graphical user interface**

**GWT. See Google Web Toolkit**

## H

**Hagen, Steve, 45–46**

**Hanselman, Scott, 36**

**Hansson, David Heinemeier, 22, 36**

**hashes, 271–272**

**Hernandez, Obie, 22**

**Hewlett Packard, 41**

**Hibernate, 48**

**HTML. See Hypertext Markup Language**

**HTTP. See Hypertext Transfer Protocol**

**Hypertext Markup Language (HTML), 2, 4–6**

- blocks, 192–193
- CGI, 12–13
- code-first development, 14
- document-first development, 17
- MVC applications, static prototype for, 53

overlying in web API, 99–100  
 page development, 9–10  
   stylesheets, 10  
   TABLE tag, 10  
 RHTML, 101  
 Ruby on Rails libraries, 30–31  
 XHTML, 4

### **Hypertext Transfer Protocol (HTTP), 5–8, 10–12**

Apache, 118  
 commands, 11  
 CRUD, 119–120  
 development of, 7–8  
 forms, 10–12  
 primary methods, 7–8  
 REST, 119–128  
   API web applications, 126–127  
   backward compatibility workaround, 120–122  
   CRUD, 119–120, 123  
   endpoints, 120–121  
   mapping, 122  
   networking, 127–128  
   refactoring, 123–126  
   resource controllers, 122  
   specification excerpts, 120

## I

**IANA. See Internet Assigned Numbers Authority IBM, 41**

**IDE. See Integrated Development Environment**

**inline reference partial, 76–77**

**in-place application style, AJAX, 153–155**

  buddy list, 154  
   meebo, 154–155

**instance\_eval method, 216–218**

  class\_eval v., 219

**instance-level endpoint, 121**

**Integrated Development Environment (IDE), 71**

**Interactive Ruby (IRB), 172–173**

**interface(s)**

  AJAX, 164–165  
   DataTables, 164–165  
 CGI, 12–18  
   code-first development, 13–15  
   document-first development, 13, 15–18

  HTML documents, 12–13

  Perl language, 13

  graphical user interface, 49

**Internet. See World Wide Web**

**Internet Assigned Numbers Authority (IANA), 104**

**Internet Explorer, 9, 140**

**iPhone, 153**

**IRB. See Interactive Ruby**

## J

**Java Swing, 48**

**JavaScript, 2, 31–32. See also Asynchronous**

**Javascript and XML**

  AJAX, xxi, 31, 140–166  
   as API, 155–156  
   design issues, 141–143  
   development of, 140–141  
   elegant degradation, 162–163  
   frameworks, 142–144  
   Ruby on Rails, 157–162  
   styles, 144–155  
   user interfaces, 164–165  
   XMLHttpRequest, 141  
 Prototype library, 142–143  
 Ruby on Rails, 31–32, 72–73  
   RJS, 31–32  
 view, in MVC, 72–73  
   GWT, 72

**Jetty, 23**

**jQuery, 144**

## K

**Keys, Adam, 211**

## L

**lambda-created procs, 178**

**LAMP model, of schema development, 254–256**

**language(s)**

  DSL, 27  
   duck-typed, 205  
   HTML, 2, 4–6

## language(s) (continued)

---

### **language(s) (continued)**

JavaScript, 2, 31–32, 72–73  
Perl, 13  
RJS, 31–32  
Ruby on Rails, 27–32  
  ActiveRecord, 28–30  
  associations, 29  
  HTML helper libraries, 30–31  
  plug-ins, 32  
  REST-based routes, 30  
  RJS, 31–32  
  symbols, 27  
UML, 51  
XML, 100–101  
  RXML, 100–101  
YAML, 23

### **layouts, 159–160**

### **Lerdorf, Rasmus, 16**

## **M**

### **mapping, 67–68**

geocoding, 67  
HTTP, 122  
routing resources, 129–133  
  anonymous, 129  
  automatically provided, 131  
  named, 129–133  
  shorthand for, 132–133

### **MapReduce, 37–38**

### **matchers, 282–287**

custom, 285–287  
methods, 283–285

### **meebo, 147–149**

compiled to web style AJAX, 153  
in-place application style AJAX, 154–155  
partial style AJAX, 149  
proxy style AJAX, 147–148  
puppet style AJAX, 151

### **Merb for Ruby, 22**

### **message paradigms, 176–177**

### **metaprogramming, 169**

### **metering, 105–109**

algorithm, 107  
filters, 108–109  
user authentication, 105–107

### **method(s)**

aliasing, 221–222  
Array.each, 170  
calls, API applications, 96–97  
define\_method, 228–234  
  class definitions, 228  
  scope of, 230–232  
  variables, 231  
DELETE, 8  
eval, 211–219  
  class Class; end, 215–216  
  class<<Class, 218–219  
  class\_eval, 213–215  
  instance\_eval, 216–218  
GET, 7  
matchers, 283–285  
method\_missing, xxi, 241–246  
  creative API applications, 241, 247–248  
  facades, 242–243  
  filters, 243–246  
  implementation, 242–246  
mixins, 201  
POST, 7  
PUT, 8  
refactored controller, 65  
reset\_password, 65  
respond\_to, 97–99  
Ruby on Rails, 173–177, 234–248,  
  250–251  
  Array.each, 237–240  
  message paradigms, 176–177  
  method\_missing, 241–246  
  scope, 175–176

### **method\_missing method, xxi, 241–246**

creative API applications, 241, 247–248  
data-driven objects, 241, 246–247  
easy reading, 241  
facades, 242–243  
filters, 243–246  
implementation, 242–246

### **methods, method\_missing**

data-driven objects, 241, 246–247  
easy reading, 241

### **Microsoft, 41**

early web page rendering, 9  
Windows, 3–4

**migrations, 256–260**

writing, 257–259

**MIME types, custom, 103–105**

creation of, 105

registration, 104

response formats, 103

**mixing, 199, 202–203**

extend with, 204–205

modules into classes, 202–203

**mixins, 198–210**

module codes, 199–200

methods, 201

source code, 206–210

examples, 207–210

Yahoo! UI, 208–210

**model, in MVC, 47, 61–77**

ActiveRDF, 70

ActiveRecord, 61–62

patterns, 61–62

ActiveResource, 70

exceptions, 64–67

definitions, 66–67

flash variables, 66

mapping, 67–68

geocoding, 67

objects, 62–64

associations, 62

hierarchy, 62

portable code, 64

user table stores, 63

polymorphic associations, 68–70

**Model-Viewer-Controller (MVC) applications,****32–33, 47–57, 61–77. See also controllers, in MVC; model, in MVC; view, in MVC**

Agile software, 55–56

philosophy manifesto, 55

architecture, 49

pages v. views, 49

components, 47–48, 50, 53–54

actions, 50, 54

controllers, 47, 50, 53–54, 77–88

CRUD, 78–80

dependency graph, 47

filters, 77

operation patterns, 78

outsourcing, 83–87

refactored, 65

refactoring, 65, 87–88

reset\_password method, 65

two-step actions, 80–83

types, 53–54

design process, 50–55

components, 50–51

UML diagrams, 51

model, 47, 61–77

ActiveRDF, 70

ActiveRecord, 61–62

ActiveResource, 70

exceptions, 64–67

mapping, 67–68

objects, 62–64

polymorphic associations, 68–70

social networking prototype, 51–55

static HTML prototype, 53

testing, 41

view, 47, 70–77

GWT, 72

IDE, 71

partials, 73–77

variables, 70–71

web applications, 48–50

GUI, 49

**modules, 251–252**

classes, 202–203

codes, 199–200

**monkey patching, 198, 210–222**

aliasing methods, 221–222

definition, 210

eval method, 211–219

class Class; end, 215–216

class&lt;&lt;Class, 218–219

class\_eval, 213–215

instance\_eval, 216–218

hazards, 220–221

techniques, 219–222

as temporary, 221

**MooTools, 144****Mosaic, 8–10**

NSCA, 8, 10

**Mozilla, 140****MVC applications. See Model-Viewer-Controller applications**

## National Center for Supercomputer Applications

---

### N

**National Center for Supercomputer Applications, 8**

#### nesting

- deep, 135
- resource controllers, 135–136
- Ruby on Rails, 133–136
  - route creation, 134–135

**Netscape, 8**

**New York Times, 116**

**NeXT system, 6**

**“Nifty Corners” (Fulciniti), 39**

**NSCA Mosaic, 8, 10**

### O

**objects, 38–39, 62–64**

- method\_missing method, 241, 246–247
- model hierarchies, 266–270
  - fields, 268–269
  - helper methods, 269–270
  - subclasses, 268–270
- model, MVC, 62–64
  - associations, 62
  - hierarchy, 62
  - portable code, 64
  - user table stores, 63
- repetition, 38–39

**outsourcing, 83–87**

- data-related operations, 85–87
  - data loading, 85–86
  - security, 86–87
- filters, 84–85
  - types, 84–85
- validations, 87

**overlying, 97–103**

- non-HTML results, 99–100
- RDF, 103
- response\_to method, 97–99
- RSS, 101–103
- XML, 100–101
  - RXML, 100–101

### P

**partial style, AJAX, 148–149, 157–159, 163–164**

backward-compatible links, 163–164

buddy lists, 148

CRUD, 157–159

page fragments, 148

**partials, 73–77**

- badge summary, 76
- forms, 75
  - extended search, 75
  - search, 75
- inline reference, 76
- profile, 75
- selection, 74–75
- sparkline, 76–77
- summaries, 76
  - badge, 76
  - row, 76

**peepcode.com, 295**

**Perl, 13**

- code-first development, 13

**plug-ins, 32**

**plural endpoint. See class-level endpoint**

**poignantguide.net, xxii**

**polymorphic associations, 68–70**

**POST method, 7**

- GET/POST paradigm, 81

**process view, Ruby on Rails, 26–27**

- action invocation, 26
- controller instantiation, 26
- response, 27
- routing, 26
- view rendering, 27

**procs, 177–179**

- blocks and, 180–181
- callbacks, 194–195
- lambda-created, 178

**Professional Rich Internet Applications: AJAX and Beyond (Wrox), 36**

**Prototype JavaScript library, 142–143**

**Prototype libraries, 31**

**proxy style, AJAX, 146–148**

- buddy lists, 146
- meebo, 147–148

**puppet style, AJAX, 149–151, 160–162**

- buddy lists, 151

control limits, 150  
 meebo, 151  
 RJS, 160–161

**PUT method, 8**

## R

**Rails. See Ruby on Rails**

**Rails JavaScript (RJS), 31, 72–73, 160–162**

application of, 162  
 Prototype libraries, 31  
 puppet style AJAX, 160–162  
 Script.aculo.us libraries, 31

**The Rails Way (Hernandez), 22**

**Raymond, Scott, 102**

**RDF, 103**

**refactored controller method, 65, 87–88**

HTTP, 123–126

**Representation State Transfer (REST), xx, 41–42, 116–138**

development, 4, 118–119  
 HTTP, 119–128  
 API web applications, 126–127  
 backward compatibility workaround, 120–122  
 CRUD, 119–120, 123  
 endpoints, 120–121  
 mapping, 122  
 networking, 127–128  
 refactoring, 123–126  
 resource controllers, 122  
 specification excerpts, 120  
 resources, 137–138  
 RESTful services, 41  
 Ruby on Rails, 30, 128–138  
 DRY philosophy, 128  
 nested resources, 133–136  
 primary keys, 136–137  
 router mapping, 129–130  
 scaffolding, 133  
 URIs, 134  
 URLs, 133  
 Semantic Web, 116  
 web services, 41–42  
 resources, 42, 116–118

**reset\_password method, 65**

**resource controllers, 122**

nesting, 135–136

**respond\_to method, 97–99**

**REST. See Representation State Transfer**

**RESTful services, 41**

**RHTML, 101**

**RJS. See Rails JavaScript**

**routing, 24–26**

API applications, 93–96  
 definition, 94–95  
 components, 94–95  
 Libs, 94  
 mapping resources, 129–133  
 anonymous, 129  
 automatically provided, 131  
 named, 129–133  
 shorthand, 132–133  
 naming methods, 94  
 options hash, 95  
 process view for Rails, 26  
 templates, 95

**row summary, 76**

**RSpec, xxi, 279–295**

additional features, 295  
 before/after statements, 287–288  
 development cycle, 279–280, 288–295  
 testing in, 293–295  
 implementing examples, 280–281  
 matchers, 28–287  
 custom, 285–287  
 methods, 283–285  
 writing process, 280–281  
 descriptions, 280

**rspec.info/.com, 295**

**RSS, 101–103**

**Ruby on Rails, xix, 2, 21–44. See also API**

**applications, web; behavior-driven development; code view, Ruby on Rails; method(s); method\_missing method; mixins; Model-Viewer-Controller applications; process view, Ruby on Rails; schema development**  
 aesthetics, 36–37  
 AJAX, 157–162

## Ruby on Rails (continued)

---

### **Ruby on Rails (continued)**

- API web applications, xix, 90–113
  - ActiveRecord, xxi, 25, 28–30
  - custom MIME type additions, 103–105
  - documentation, 60–61
  - metering, 105–109
  - method calls, 96–97
  - overlying, 97–103
  - routing, 93–96
  - service, 109–112
  - SOAP, 109–110
  - URLs, 91–93, 95–96
- assumptions, 34–36
- blocks, 169–195
  - ActiveRecord transactions, 190–192
  - AOP, 187–189
  - Array.each method, 170
  - callbacks, 194–195
  - code wrapping, 172
  - definition, 169–173
  - dual-use functions, 194
  - environmental effects, 184–186
  - filters, 189–190
  - HTML writing, 192–193
  - iterations, 186–187
  - source environment influences, 182
  - XML, 193–194
- code repetition, 38–40
  - AOP, 38
  - behaviors, 40
  - objects, 38–39
  - processes, 40
- code view, 25–26
  - ActionPack, 25–26
  - ActiveRecord, 25
  - ActiveResource, 25
- code-writing macros, 228–234
  - define\_method, 228–234
- constants, 249–250
- convention-based philosophies, 34–35
  - relaxed, 35
- CRUD, 78–80
  - scaffold generators, 79–80
- development, 37–38, 43
  - philosophy, 43
  - restraint, 37–38
- DSL, 227–228
- as ecosystem, 33–34
- framework, 22–27
  - code view, 25–26
  - configuration, 23–25
  - database, 24
  - directories, 23–24
  - process view, 26–27
  - routes, 24–25
- IRB, 172–173
- language, 27–32, 72–73
  - ActiveRecord, 28–30
  - associations, 29
  - HTML helper libraries, 30–31
  - Javascript, 31–32, 72–73
  - plug-ins, 32
  - REST-based routes, 30
  - RJS, 31–32
  - symbols, 27
  - YAML, 23
- loading data, 85–86
- methods, 173–177, 234–248, 250–251
  - Array.each, 237–240
  - message paradigms, 176–177
  - method\_missing, 241–246
  - scope, 175–176
- mixing, 199, 202–203
  - extend with, 204–205
  - modules into classes, 202–203
- mixins, 198–210
  - module codes, 199–200
  - source code, 206–210
- modules, 251–252
- monkey patching, 198, 210–222
  - aliasing methods, 221–222
  - definition, 210
  - eval method, 211–219
  - hazards, 220–221
  - techniques, 219–222
  - as temporary, 221
- MVC applications, 32–33, 47–57
  - Agile software, 55–56
  - architecture, 49
  - components, 47–48, 50, 53–54
  - design process, 50–55

- social networking prototype, 51–55
- static HTML prototype, 53
- testing, 41
- web applications, 48–50
- pluralization, 36
- procs, 177–179
  - callbacks, 194–195
  - lambda-created, 178
- REST, 30, 128–138
- schema development, 254–273
  - custom getters, 272–273
  - hashes, 271–272
  - LAMP model, 254–256
  - list storage, 271–272
  - migrations, 256–260
  - model object hierarchies, 266–270
  - seeded data, 262–266
  - setters, 272–273
  - teams for, 260–262
- security, 86–87
- singleton class, 217
- testing, 40–41
- variables, 249–250
- web services, 41–42
  - REST, 41–42
  - SOAP, 41
  - WS-\*, 41–42
  - WSDL, 41

**RXML, 100–101****S****Sapir-Worf Hypothesis, 282****scaffold generator, 79–80**

Ruby on Rails, 133

**schema development, 254–273**

custom getters, 272–273

hashes, 271–272

LAMP model, 254–256

list storage, 271–272

migrations, 256–260

writing, 257–259

model object hierarchies, 266–270

fields, 268–269

helper methods, 269–270

subclasses, 268–270

- seeded data, 262–266
  - large datasets, 264–266
  - medium datasets, 263–264
  - small datasets, 262–263
- setters, 272–273
- teams for, 260–262

**script.aculo.us libraries, 31, 143****search forms, extended, 75****seeded data, 262–266**

large datasets, 264–266

medium datasets, 263–264

small datasets, 262–263

**Semantic Web, 4**

REST, 116

**service API applications, web, 109–112**

definitions, 110

implementation, 111–112

structs, 112

**setters, 272–273****singleton class, 217****singular endpoint. See instance-level endpoint****SOAP, 41**

web API applications, 109–110

**sparkline partial, 76–77****SQL, 2****structs, 112****stylesheets, 10****T****TABLE tag, 10****TDD. See test-driven development****test-driven development (TDD), 276****testing**

MVC applications, 41

Ruby on Rails, 40–41

**Thomas, Dave, 22****Tomcat, 23, 33****Tufte, Edward, 76****Tumblr, 127–128****TurboGears for Python, 22****two-step actions, 80–83**

flash variables, 83

GET/POST paradigm, 81

object displays, 82–83

object mutations, 82–83

## UDIs

---

### U

**UDIs.** *See* **Universal Document Identifiers**

**UML.** *See* **United Modeling Language**

**Uniform Resource Locators (URLs), 91–93, 95–96**

concept v. function for, 92

design, 95–96

development of, 91–92

hierarchy, 93

Ruby on Rails, 133

**United Modeling Language (UML), 51**

**Universal Document Identifiers (UDIs), 6**

**Universal Resource Identifier (URI), 117–118**

Ruby on Rails, 134

**Unix, 33**

**URI.** *See* **Universal Resource Identifier**

**URLs.** *See* **Uniform Resource Locators**

**user table stores, 63**

### V

**variables, 70–71**

flash, 66

Ruby on Rails, 249–250

**view, in MVC, 47, 70–77.** *See also* **partials**

GWT, 72

IDE, 71

JavaScript, 72–73

partials, 73–77

badge summary, 76

forms, 75

inline reference, 76

profile, 75

row summary, 76

selection, 74–75

sparkline, 76–77

variables, 70–71

### W

**Weaving the Web (Berners-Lee), 5**

**web services, 41–42.** *See also* **World Wide Web**

REST, 41–42, 116–118

resources, 42, 116–118

SOAP, 41

WS-\*, 41–42

server capabilities, 42

WSDL, 41

**websites**

artofrails.com, 105

p2p.wrox.com, xxiii

peepcode.com, 295

poignantguide.net, xxii

rspec.info/.com, 295

wrox.com, xxii

**Why's (poignant) Guide to Ruby, xxii**

**Wikipedia, 4**

**World Wide Web, 1–19.** *See also* **API**

**applications, web; Berners-Lee, Tim;**

**Hypertext Markup Language; Hypertext**

**Transfer Protocol; Model-Viewer-Controller**

**applications**

1.0, 3

2.0, 3–4

3.0, 3–4

API applications, xix, 90–113

ActiveRecord, xxi, 25, 28–30

custom MIME type additions,  
103–105

documentation, 60–61

metering, 105–109

method calls, 96–97

overlying, 97–103

routing, 93–96

service, 109–112

SOAP, 109–110

URLs, 91–93, 95–96

application development, 1–2, 12–18

CGI, 12–13

browsers, 6

CSS, 2

development history, 5–18

HTML, 2, 4–6

page development, 9–10

HTTP, 5–8

development of, 7–8

JavaScript, 2

MVC applications, 48–50

Ruby on rails, 2

Semantic, 4, 9–12

SQL, 2  
UDIs, 6

**Writely, 72**

**wrox.com, xxii**

**WSDL, 41**

## X

**XHTML, 4**

**XMLHttpRequest, 141**

**XML. See Extensible Markup Language**

## Y

**Yahoo! UI (YUI), 143**

mixins, 208–210

**YAML, 23**

**YouTube, 2, 4**

**YUI. See Yahoo! UI**

## Z

**Zope Content Management System, 211**