Various Listings

```
# alpha.txt
zebra
yen
xmas
birthday
cat
apple
pune
mumbai
nashville
```

```
# config.ru
use Rack::Static,
:urls => ["/stylesheets", "/images"],
:root => "public"
run lambda { |env|
  [
   200,
   {
    'Content-Type' => 'text/html',
    'Cache-Control' => 'public, max-age=86400'
   },
   File.open('Dosa_Diner/Dosa_Diner.htm', File::RDONLY)
  ]
```

```
# gc.yml
--- !ruby/object:GameCharacter
power: 120
type: Magician
weapons:
- spells
- invisibility
```

```
# hpricot_scan_doc.rb

require 'open-uri'
require 'hpricot'

url = 'http://ruby-
metaprogramming.rubylearning.com/html/ruby_metaprogramming_1.html' # =&gt;
```

```
String
pattern = /\bthe\b/i                              # =&gt; Regexp
doc = Hpricot(open(url))                          # =&gt; Hpricot::Doc
html_string = doc.to_s                            # =&gt; String
count = html_string.scan(pattern).size            # =&gt;  Fixnum
puts "The number of times the word 'the' appears in the document is #{count}."
```

```
# infinity.rb
require 'rack'

env = []
version = ["infinity 0.1"]
last = ["infinity beta 0.0"]

infinity = Proc.new {|env| [200, {"Content-Type" => "text/html"}, env]}
builder = Rack::Builder.new do
  use Rack::CommonLogger

  map '/' do
    run infinity
  end

  map '/version' do
    map '/' do
      run Proc.new {|env| [200, {"Content-Type" => "text/html"}, version]}
    end

    map '/last' do
      run Proc.new {|env| [200, {"Content-Type" => "text/html"}, last] }
    end
  end
end

Rack::Handler::Thin.run builder, :Port => 9292
```

```
# infinity.ru
infinity = Proc.new {|env| [200, {"Content-Type" => "text/html"},
[env.inspect]]}

use Rack::CommonLogger

map '/' do
  run infinity
end

map '/version' do
  map '/' do
```

```
    run Proc.new {|env| [200, {"Content-Type" => "text/html"}, ["infinity
0.1"]] }
  end

  map '/last' do
    run Proc.new {|env| [200, {"Content-Type" => "text/html"}, ["infinity beta
0.0"]] }
  end
end
```

```
# lobster.ru
require 'rack/lobster'
run Rack::Lobster.new
```

```
# lobster1.ru
require 'rack/lobster'
require './my_middleware'
use MyMiddleware::Hello
run Rack::Lobster.new
```

```
# make_yaml.rb
require 'yaml'
require './p051gamecharacters'
gc = GameCharacter.new(120, 'Magician', ['spells', 'invisibility'])
open("gc", "w") { |f| YAML.dump(gc, f) }
data = open("gc") { |f| YAML.load(f) }
puts data.power.to_s + ' ' + data.type + ' '
data.weapons.each do |w|
puts w + ' '
end
```

```
# method_call_Rack.rb
require 'rack'

def my_method env
  [200, {}, ["method called"]]
end

Rack::Handler::WEBrick.run method(:my_method)
```

```
# most_simple_Rack.rb
```

```ruby
require 'rack'

class HelloWorld

  def self.call(env)
    [200, {'Content-Type' => 'text/html'}, ["Hello World!"]]
  end

end

new_object = HelloWorld
new_object.call({})
Rack::Handler::WEBrick.run new_object
```

```ruby
# my_middleware.rb

module MyMiddleware
  class Hello
    def initialize(app)
      @app = app
    end

    def call(env)
      if env['PATH_INFO'] == '/hello'
        [200,  {"Content-Type" => "text/plain"}, ["Hello from the
middleware!"]]
      else
        # forward the request
        @app.call(env)
      end
    end
  end
end
```

```ruby
# my_middleware.ru
require './my_middleware'
use MyMiddleware::Hello # this comes in between
run Proc.new{|env|  [200, {"Content-Type" => "text/plain"}, ['Try accessing
visiting /hello']] }
```

```ruby
# my_middleware1.rb
# <img alt="big grin" width="99" height="99"
src="http://rubylearning.org/class/pix/u/user100.png"/>
#~ $body = ['<html><head><title>OK!</title><body><h1>Okay!</h1></body></html>'
module MyMiddleware
```

```ruby
  class Hello
    def initialize(app)
      @app = app
    end

    def call(env)
      if env['PATH_INFO'] != '/'
        # forward the request
        @app.call(env)
      else
        [200,  {"Content-Type" => "text/html"}, ['<h1>Okay!</h1>']]
      end
    end
  end
end
```

```ruby
# my_middleware1.ru
require './my_middleware1'
use MyMiddleware::Hello # this comes in between
run lambda{|env| [404,
                {"Content-Type" => "text/html"},
                ['<h1>NOT OK!</h1>']]
        }
```

```ruby
# ruby my_rack_method2.rb Hello_Rack_app

require 'rack'

$body = ["my_method called\n", "The time is #{Time.now}\n",
         "The command line argument you typed was: #{ARGV[0]}"]

def my_method(env)
  [200, {}, $body]
end

Rack::Server.new( { :Port => 8500,
                    :server => 'webrick',
                    :app => method(:my_method) } ).start

=begin
    http://localhost:8500

    method called
    The time is 2012-03-07 17:01:35 -0600
    The command line argument you typed was: Hello_Rack_app
=end
```

```ruby
# my_rack_proc.rb
require 'rack'

# Rack::Handler.constants

# Rack::Handler::WEBrick

$body = ["Hello. The time is #{Time.now}"]

my_rack_proc = lambda { |env| [200, {"Content-Type" => "text/plain"}, $body] }

server_hash = { :app => my_rack_proc, :server => 'webrick', :Port => 9876}
# Rack::Handler::WEBrick.run my_rack_proc

Rack::Server.new( server_hash ).start

=begin
Options may include:

:app
  a rack application to run (overrides :config)
:config
  a rackup configuration file path to load (.ru)
:environment
  this selects the middleware that will be wrapped around
  your application. Default options available are:
    - development: CommonLogger, ShowExceptions, and Lint
    - deployment: CommonLogger
    - none: no extra middleware
  note: when the server is a cgi server, CommonLogger is not included.
:server
  choose a specific Rack::Handler, e.g. cgi, fcgi, webrick
:daemonize
  if true, the server will daemonize itself (fork, detach, etc)
:pid
  path to write a pid file after daemonize
:Host
  the host address to bind to (used by supporting Rack::Handler)
:Port
  the port to bind to (used by supporting Rack::Handler)
:AccessLog
  webrick acess log options (or supporting Rack::Handler)
:debug
  turn on debug output ($DEBUG = true)
:warn
  turn on warnings ($-w = true)
:include
  add given paths to $LOAD_PATH
:require
  require the given libraries
=end
```

```
# my_rack_proc2.rb
require 'rack'

my_rack_proc = lambda { |env|
                       [200,
                        {"Content-Type" => "text/plain"},
                        ["proc called\n","Hello. The time is #{Time.now}\n",
                         "The command line argument you typed was: #
{ARGV[0]}"]
                       ]
                      }

run_hash = {:Port => 8500 ,
            :server => 'webrick',
            :app => my_rack_proc}

Rack::Server.new( run_hash ).start

# ruby my_rack_proc2.rb "Hello Simple Rack app"
=begin
                                http://localhost:8500

                    proc called
                                Hello. The time is 2013-01-11 17:01:35 -0600
                                The command line argument you typed was:
Hello_Rack_app
=end
```

```
# my_request.rb

class MyRequest
  def call(env)
    req = Rack::Request.new(env)

          req.get? name = req.params["name"]
          req.get? text = req.params["text"]
          # if req.get?
          #   name = "my"
          #   text = "data"
          # end

    Rack::Response.new.finish do |res|
      res['Content-Type'] = 'text/plain'
      res.status = 200
      # if req.post?
      #   name = req.params["name"]
      #   text = req.params["text"]
```

```
      # end
      res.write "Parameters sent: name - #{name} | text - #{text}"
                res.write "Parameters reversed: name - #{name.reverse} | text
- #{text.reverse}"
        end
  end
end

# curl -X POST -d 'name=GD,nellA&text=!emosewa si gninraelybuR' localhost:9292
```

```
# my_request.ru
require './my_request'
run MyRequest.new
```

```
#net_html_uri_scan_doc.rb

require 'net/http'

url = 'http://ruby-
metaprogramming.rubylearning.com/html/ruby_metaprogramming_1.html'
                                                  # =&gt; String
pattern = /\bthe\b/i                              # =&gt; Regexp
doc = URI.parse(url)                              # =&gt; URI::HTTP
html_string = Net::HTTP.get(doc)                  # =&gt; String
count = html_string.scan(pattern).size            # =&gt;  Fixnum
puts "The number of times the word 'the' appears in the document is #{count}."
```

```
curl --data "str=gnirts detsop siht"  -X POST "http://rl-string-
reverse.herokuapp.com/reverse"
```

```
# nokogiri_scan_doc.rb

require 'nokogiri'
require 'open-uri'

url = 'http://ruby-
metaprogramming.rubylearning.com/html/ruby_metaprogramming_1.html'
                                                # =&gt; String
pattern = /\bthe\b/i                            # =&gt; Regexp
doc = Nokogiri::HTML(open(url))                 #
=&gt;Nokogiri::HTML::Document
html_string = doc.to_s
count = html_string.scan(pattern).size          # =&gt;  Fixnum
```

```
puts "The number of times the word 'the' appears in the document is #{count}."
```

```
# open_uri_scan_doc.rb

require 'open-uri'

url = 'http://ruby-
metaprogramming.rubylearning.com/html/ruby_metaprogramming_1.html'
                                          # =&gt; String
pattern = /\bthe\b/i                      # =&gt; Regexp
doc = open(url)                           # =&gt; StringIO
html_string = doc.readlines.join          # =&gt; String
count = html_string.scan(pattern).size    # =&gt; Fixnum
puts "The number of times the word 'the' appears in the document is #{count}."
```

```
# p051gamecharacters.rb
class GameCharacter
  def initialize(power, type, weapons)
    @power = power
    @type = type
    @weapons = weapons
  end
  attr_reader :power, :type, :weapons
end
```

```
# rack_builder.rb
# enter your latitude as first arg the longitude
# ruby rack_builder.rb 51.1789 -1.8264   (Stonehenge coordinates)

require 'eot'
require 'rack'
# require 'logger'
require 'thin'

app = Rack::Builder.new do
  @solar = Eot.new
  @t = Time.now.utc
  @date = Date.parse(@t.year.to_s + "-" + @t.month.to_s + "-" + @t.day.to_s)
  @solar.date = @date
  @solar.jd = @date.jd
  @solar.ajd = @date.ajd
  @solar.latitude= ARGV[0].to_f
  @solar.longitude= ARGV[1].to_f
  use Rack::CommonLogger
  use Rack::ContentType, "text/html"
```

```
  run lambda { |env| [200, {}, ["<p><h2>Sunrise: #
{@solar.display_time(@solar.sunrise_time(@solar.jd))[0..7]} UTC
 <p><h2>Sunset: #{@solar.display_time(@solar.sunset_time(@solar.jd))[0..7]}
UTC" ]] }
end

Rack::Handler::Thin.run app, :Port => 8500
```

```
# rack_builder.ru
rack_time = Proc.new { |env| [200, {"Content-Type" => "text/plain"}, ["Hello.
The time is #{Time.now}"]] }
Rack::Handler::Thin.run rack_time, :Port => 9292
```

```
# rack_builder1.ru
rack_time = lambda { |env| [200, {"Content-Type" => "text/plain"}, ["Hello.
The time is #{Time.now}"]] }
builder = Rack::Builder.new do
  run rack_time
end
Rack::Handler::Thin.run builder, :Port => 9292
```

```
# rack_builder2.ru
require 'logger'
rack_time = Proc.new { |env| [200, {"Content-Type" => "text/plain"}, ["Hello.
The time is #{Time.now}"]] }
builder = Rack::Builder.new do
  use Rack::CommonLogger
  Logger.new('rack.log')
  run rack_time
end
Rack::Handler::Thin.run builder, :Port => 9292
```

```
# rack_builder3.ru
require 'logger'
rack_app = Rack::Builder.new do
  use Rack::CommonLogger
  Logger.new('rack.log')

  map "/" do
    run Proc.new {|env| [200, {"Content-Type" => "text/html"}, ["This is
public page"]] }
  end
```

```
  map "/secret" do
    use Rack::Auth::Basic, "Restricted Area" do |user, password|
      user == 'super' && password == 'secretsauce'
    end

    map "/" do
      run Proc.new {|env| [200, {"Content-Type" => "text/html"}, ["This is a
secret page"]] }
    end

    map "/files" do
      run Proc.new {|env| [200, {"Content-Type" => "text/html"}, ["Here are
the secret files"]] }
    end
  end
end
Rack::Handler::WEBrick.run rack_app, :Port => 9292
```

```
# rack_embedded_html_page.rb
require 'rack'
require 'rack/server'

# "Content-Type" => "text/html"
# "Content-Type" => "application/xhtml+xml"
class HelloWorld

  def initialize(str)
    @content = str
  end

  def response
    [200, {'Content-Type' => 'text/html'}, [@content]]
  end

end

$doc = <<"page"
<!DOCTYPE html>
<!-- saved from url=(0038)http://calm-plains-9022.herokuapp.com/ -->
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"></meta>
    <title>Dosa Diner</title>
    <meta charset="utf-8"></meta>
    <link rel="stylesheet" src="dosasite.css"></link>
    <script>
        alert("Hello Course Participants");
     </script>
  </head>
```

```
  <body>
    <h1><img src="/dosa.jpg" alt=""></img>Diner</h1>
    <h2>The Restaurant</h2>
    <p>The Dosa Diner offers casual lunch and dinner fare in a hip atmosphere.
The menu changes regularly to highlight the freshest ingredients.</p>

    <h2>Catering Services</h2>
    <p>You have fun... we'll do the cooking. Dosa Diner Catering can handle
events from snacks for bridge club to elegant corporate fundraisers.</p>

    <h2>Location and Hours</h2>
    <p>Deccan Corner in Pune, India;
    Monday through Thursday 11am to 9pm, Friday and Saturday, 11am to
midnight</p>
  </body>
</html>
page

class HelloWorldApp
  def self.call(env)
    HelloWorld.new($doc).response
  end
end

# Rack::Static.new(options={:urls => ["/css", "/images"], :root => "public"})
Rack::Server.start(:app => HelloWorldApp, :urls => ["/css", "/images"], :root
=> "public" )



class EnvInspector
  def self.call(env)
    [200, {}, env.inspect]
  end
end

# Rack::Server.start :app => EnvInspector
```

```
# rack_example.ru
##########################################
# Very basic rack application showing how to use a router based on the uri
# and how to process requests based on the HTTP method used.
#
# Usage:
# $ rackup rack_example.ru
#
# $ curl -X POST -d 'title=retire&body=I should retire in a nice island'
localhost:9292/ideas
# $ curl -X POST -d 'title=ask Satish for a gift&body=Find a way to get Satish
to send me a gift' localhost:9292/ideas
```

```
# $ curl localhost:9292/ideas
#
# More info: https://github.com/rack/rack/wiki/Rack-app-with-uri-and-HTTP-
specific-responses
#########################################

class Idea
  attr_accessor :title, :body, :created_at

  # Memory store, gets cleared as the process is restarted
  def self.store
    @ideas ||= []
  end

  class InvalidParams < StandardError; end

  # create an instance based on some passed params
  def initialize(params)
    raise InvalidParams, "You need to provide at least a title" unless
params['title']
    self.title = params['title']
    self.body = params['body']
    self.created_at = Time.now
  end

  # Converts an instance into a string
  def to_s
    "#{title} at #{created_at.to_s}\n#{body}"
  end
end

class IdeaAPI
  def call(env)
    request = Rack::Request.new(env)
    case request.request_method
    when 'POST'
      begin
        idea = Idea.new(request.params)
      rescue Idea::InvalidParams => error
        [400, {"Content-Type" => "text/plain"}, [error.message] ]
      else
        Idea.store << idea
        [200, {"Content-Type" => "text/plain"}, ["Idea added, currently #
{Idea.store.size} ideas are in memory!"]]
      end
    when 'GET'
      [200, {"Content-Type" => "text/plain"}, [Idea.store.map{|idea, idx|
idea.to_s }.join("\n\n") + "\n"]]
    else
      [404, {}, ["Did you get lost?"]]
    end
  end
```

```
end

map '/ideas' do
  run IdeaAPI.new
end
```

```
# racktest_test.rb
require "rack/test"
require 'test/unit'

class HomepageTest < Test::Unit::TestCase
  include Rack::Test::Methods
  class MyApp
  end
  def app
    MyApp.new
  end

  # def test_redirect_logged_in_users_to_dashboard
  # authorize "bryan", "secret"
  # get "/"
  #   follow_redirect!
  #   assert_equal "http://example.org/redirected", last_request.url
  #   assert last_response.ok?
  # end
end
```

```
# README
Preparing the Core Ruby participants for the web

RubyLearning.org is planning a free, online course on topics that hopefully
will help those that have some knowledge of Ruby programming to get started
with web programming - call it "Intermediate Ruby Programming". This does not
cover Ruby on Rails.

The course material is under preparation and I am looking for help from you to
help create - add / subtract / modify the topics and material for this course.
We also require many problems/solutions for the course participants.

The course is scheduled for 7th Jan. 2012 and we would have 3-4 mentors
helping out the participants during the course.

Can you help?


Update (21st Nov 2011):
Credits -
```

```
a. Oto Brglez ( https://github.com/otobrglez ) for "Using Nokogiri"
b. Michael Kohl ( https://github.com/citizen428 ) for proof reading and making
relevant corrections to chapters 1 and 2.

http://chneukirchen.org/blog/archive/2007/02/introducing-rack.html
http://rack.github.io/
http://en.wikipedia.org/wiki/Rack_%28web_server_interface%29
https://github.com/rack/rack
https://github.com/rack/rack/wiki
```

```
# rfc2616
https://tools.ietf.org/html/rfc2616
```

```
# simple.ru
# config.ru
# Usage: rackup <arg> config.ru
run lambda { |env|
             [200,
             {"Content-Type" => "text/plain"},
             ["Hello. The command line argument you entered is #{ARGV[0]}"]]
           }
```

```
# simple_rack.rb
# ruby simple_rack.rb Hello_World

ARGV[0] = "Hello. The time is #{Time.now}" if ARGV[0].nil?
env = {}
env[:arg] = ARGV[0]
simple_rack = lambda {|env|
                      [200,
                       {"Content-Type" => "text/html"},
                       ["ruby simple_rack.rb #{env[:arg]}"]
                      ]
                     }

puts simple_rack.call(env)

#~ puts "ARGV[0] is a #{ARGV[0].class} class"
```

```
# simple_rack1.rb
require 'rack'
```

```
my_rack_proc = lambda {
                        |env| #block
                        [
                         # status
                         200,
                         # header
                         {"Content-Type" => "text/html"},
                         # body
                          [
<<EOF
<h1>Hello!</h1><br/>
<h2>I 'm a simple Rack application.</h2><br/>
<h2>I 'm running in Ruby.</h2><br/>
<h3>This is a lambda which creates a new Proc object.</h3><br/>
<h3>My header says "Content-Type" => "text/html"</h3>
<h4>So I may display on your browser that way.</h4>
<h4>test h4</h4><br/>
<a href="https://gist.github.com/4082756">Put me back when you're done.</a>
EOF
                           ]
                         ]
                        }


#~ p my_rack_proc.call({})
Rack::Handler::WEBrick.run my_rack_proc
```

```
# simple_rack2.rb
# ruby simple_rack2.rb Hello_World
require 'rack'

ARGV[0] = "Hello World"
env = {}
env[:arg] = ARGV[0]
simple_rack2 =  lambda {
                         |env| [200,
                         {"Content-Type" => "text/html"},
                         ["Command line argument you typed was: #{env[:arg]}"]]
                        }
```

```
# test_headers.rb
require 'sinatra/base'

class App < Sinatra::Base

  get '/' do
    "#{env.inspect}"
```

```
  end

end
```

```
# test_headers.ru
require 'sinatra'
require './test_headers'

run App
```

```
# toc.html
<!doctype html>
<html>
<head>
<title>Intermediate Ruby</title>
<meta charset="utf-8">
</head>
<body>
<div>
<h2>Course Contents</h2>
<ul>
<li>DAY 1</li>
<li>Using Git
  <ul>
    <li>What's Version Control</li>
    <li>What's Git?
      <ul>
        <li>Downloading and Installing Git</li>
        <li>Create a local folder</li>
        <li>Let us start using Git</li>
        <li>Introduce yourself to Git</li>
        <li>Create your SSH Key</li>
      </ul>
    </li>
  </ul>
</li>
<li>Using GitHub
  <ul>
    <li>What's GitHub?
      <ul>
        <li>Set up your GitHub account</li>
        <li>Creating a new repository</li>
        <li>Add your SSH key to GitHub</li>
      </ul>
    </li>
  </ul>
</li>
```

```
<li>DAY 2</li>
<li>Creating a simple webpage using HTML5, CSS and JavaScript
  <ul>
    <li>A Webpage, Step by Step</li>
    <li>Before we begin, Launch a Text Editor
      <ul>
        <li>Step 1: Start with content</li>
        <li>Step 2: Give the document structure</li>
        <li>Step 3: Identify text elements</li>
        <li>Step 4: Add an image</li>
        <li>Step 5: Change the look with a style sheet</li>
        <li>Add some JavaScript</li>
      </ul>
    </li>
  </ul>
</li>
<li>Store your webpage files on GitHub</li>
<li>DAY 3</li>
<li>Understanding HTTP concepts
  <ul>
    <li>What's HTTP?
      <ul>
        <li>Loading a web page</li>
      </ul>
    </li>
    <li>HTTP request methods (verbs)
      <ul>
        <li>GET</li>
        <li>POST</li>
        <li>PUT</li>
        <li>DELETE</li>
      </ul>
    </li>
    <li>HTTP response codes</li>
    <li>net/http library
      <ul>
        <li>Using URI</li>
      </ul>
    </li>
    <li>Using open-uri</li>
    <li>Using Hpricot</li>
    <li>Using Nokogiri
      <ul>
        <li>Fetching documents from web</li>
        <li>Searching inside HTML documents</li>
      </ul>
    </li>
  </ul>
</li>
<li>DAY 4</li>
<li>Creating one's own Ruby Gem
  <ul>
```

```
    <li>What's a Ruby Gem?</li>
    <li>Let us create a simple Ruby library</li>
    <li>Steps for publishing our gem</li>
  </ul>
</li>
<li>DAY 5</li>
<li>Learning Rack
  <ul>
    <li>What's Rack?</li>
    <li>Rack Documentation</li>
    <li>Installing Rack</li>
    <li>A quick visit to Ruby's proc object</li>
    <li>A simple Rack app - my_rack_proc</li>
    <li>Another Rack app - my_method</li>
    <li>Using rackup</li>
  </ul>
</li>
<li>DAY 6</li>
<li>Deploying Pure Rack Apps to Heroku
  <ul>
    <li>What's Heroku?
      <ul>
        <li>Create an account on Heroku</li>
      </ul>
    </li>
  </ul>
</li>
<li>DAY 7</li>
<li>Deploying a static webpage to Heroku</li>
<li>DAY 8</li>
<li>What's JSON?</li>
<li>Using MongoDB with Ruby Mongo driver
  <ul>
    <li>What's NoSQL?</li>
    <li>What's MongoDB?</li>
    <li>Setup MongoDB</li>
    <li>MongoDB Core Concepts</li>
    <li>The Basics
      <ul>
        <li>Switch databases</li>
                <li>Insert a document</li>
        <li>Use find()</li>
        <li>Removing all documents</li>
        <li>Query Selectors</li>
        <li>Updating a document</li>
      </ul>
    </li>
    <li>MongoDB Ruby Driver - mongo
      <ul>
        <li>Installation</li>
        <li>Using the mongo gem</li>
        <li>Making a Connection</li>
```

```
        <li>Getting a List Of Collections</li>
        <li>Getting a Collection</li>
        <li>Inserting a Document</li>
        <li>Updating a Document</li>
      </ul>
    </li>
    <li>MongoHQ the hosted database
      <ul>
        <li>Sign Up</li>
        <li>Create a database</li>
        <li>Accessing the database</li>
      </ul>
    </li>
  </ul>
</li>
<li>DAY 9</li>
<li>Sinatra with MongoDB
  <ul>
    <li>What's Sinatra?</li>
    <li>Create a folder on your hard disk</li>
    <li>Install Sinatra</li>
    <li>Work-In-Progress</li>
  </ul>
</li>
<li>References</li>
</ul>

</div>
</body>
</html>
```

```
# upload.ru
app = proc do |env|
  response  = Rack::Response.new
  request   = Rack::Request.new(env)
  info      = request.params['info']

  if info and info[:tempfile]
    response['Content-Type'] = info[:type]
    response.body            = info[:tempfile].readlines.sort
  else
    response['Content-Type'] = 'text/plain'
    response.status          = 400
    response.write "info parameter must be a file upload"
  end

  response.finish
end

run app
```

```
# curl -F "info=@alpha.txt" localhost:9292
```

```
# use_Rack_Builder.rb
require 'rack'

infinity = Proc.new {|env| [200, {"Content-Type" => "text/html"},
env['rack.input']]}
builder = Rack::Builder.new do
  use Rack::CommonLogger
  run infinity
end
Rack::Handler::WEBrick.run builder, :Port => 9292
```

Most of these are different enough to just put them all in one directory.