Tutorium Skriptsprachen

2009 - Max Maurer

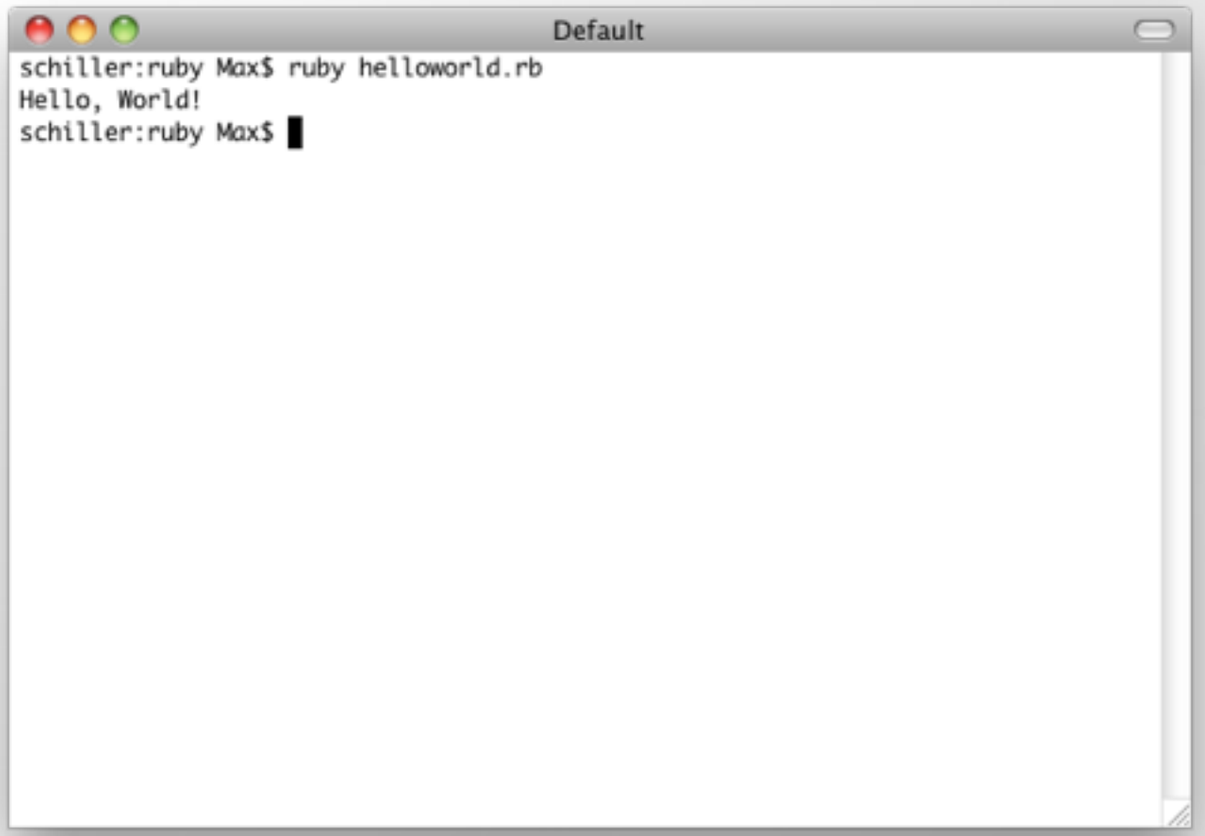
Von der Bibel zu Ruby

> 44 Mit dem Himmelreich ist es wie mit einem Schatz, der in einem Acker vergraben war. Ein Mann entdeckte ihn, grub ihn aber wieder ein. Und in seiner Freude verkaufte er alles, was er besaß, und kaufte den Acker. 45 Auch ist es mit dem Himmelreich wie mit einem Kaufmann, der schöne Perlen suchte. 46 Als er eine besonders wertvolle Perle fand, verkaufte er alles, was er besaß, und kaufte sie.
>
> Matthäus 13, Vers 44-46

Hello World

Hello World!

```
#!/path/to/ruby
puts "Hello, World!"
```



Allgemeines

Fakten

Entstanden:	1995
Erfinder:	Yukihiro Matsumoto
Firma:	keine
Lizenz:	GPL
Stärken:	„Duck Typing“
Anwendungsgebiete:	Internetskriptsprache C-Erweiterung

Yukihiro Matsumoto

Yukihiro Matsumoto
(c) Wikimedia Commons

> “Often people, especially computer engineers, focus on the machines. They think, "By doing this, the machine will run faster. By doing this, the machine will run more effectively. By doing this, the machine will something something something." They are focusing on machines. But in fact we need to focus on humans, on how humans care about doing programming or operating the application of the machines. We are the masters. They are the slaves.”

Wikipedia.de

Einsatzgebiete

- Ruby wurde bereits 1993 entwickelt und 1995 veröffentlicht
- Anfangs nur japanische Dokumentation
  - Erfolg kam als englische Dokumentation zur Verfügung stand
- Sehr flexibel (objektorientiert aber auch funktional)
- Von jedem Programmierer mit Erfahrungen einer anderen Sprache leicht zu erlernen

Characteristika

- Interpreterbasierte Sprache
- vereint verschiedene Programmierparadigmen (imperativ, objektorientiert, funktional, aspektorientiert)
- dynamische Typbindung („duck typing“)
- garbage collection
- Semantik durch ‚do‘ und ‚end‘ **oder** geschweifte Klammern
- Keine Strichpunkte
- Kein Standard für die GUI-Erzeugung (aber z.B.: GUI:TK, GTK+, FOX, SWin/VRuby)
- Open classes

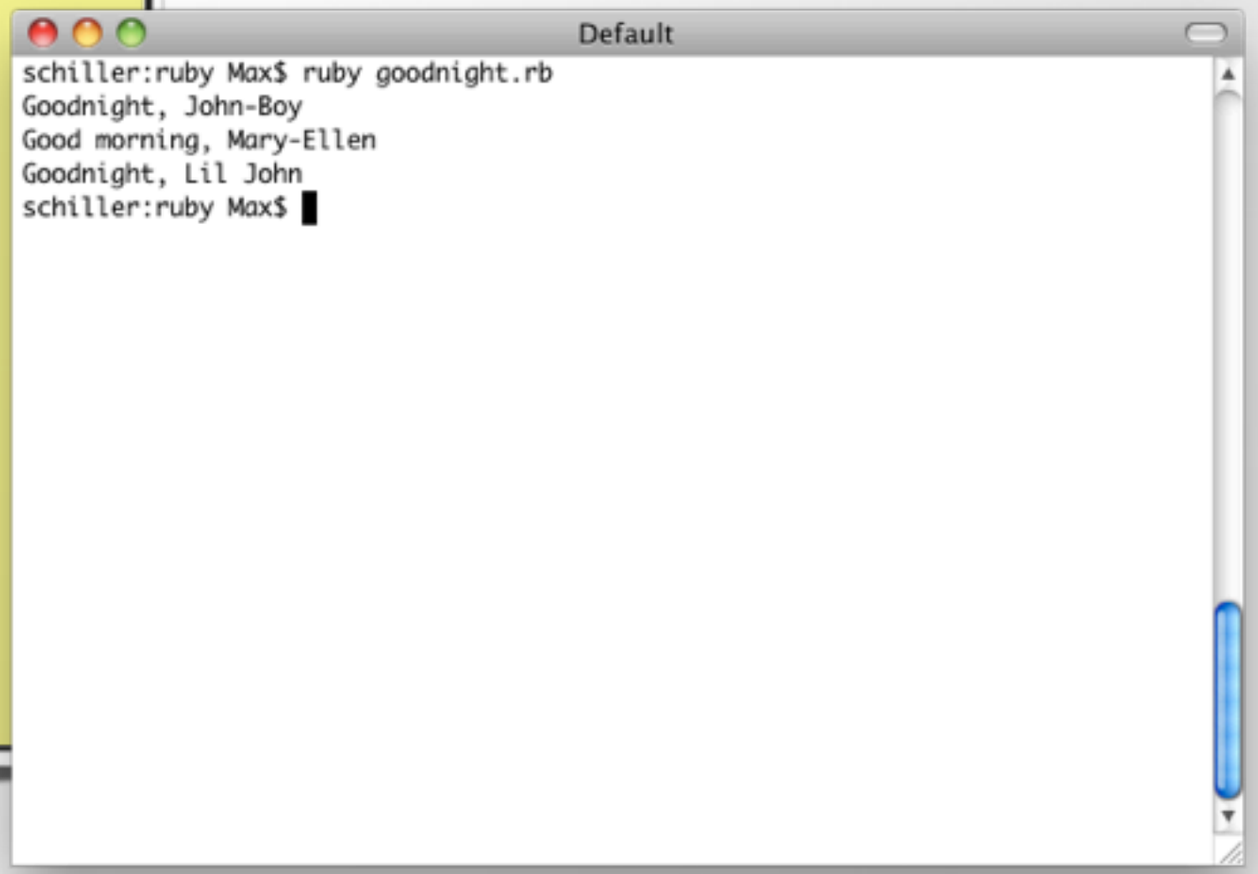
Code-Beispiele

Funktionen und Parameter

- Funktionsdefinitionen mit Schlüsselwort def
- Parameter erhalten Namen und optional einen Standardwert

```
#!/usr/bin/ruby
def sayGoodnight(morning=0, name="Lil John")
    if morning==0
        result = "Goodnight, "
    else
        result = "Good morning, "
    end
  result = result + name
  return result
end


puts sayGoodnight(0, "John-Boy")
puts sayGoodnight(1, "Mary-Ellen")
puts sayGoodnight()
```



Globale und lokale Variablen

- Variablenarten: Globale und lokale sowie Instanz- und Klassenvariablen (s. Klassen)
- **Ganz neu** in Ruby seit Version 1.9 ‚Blockvariablen‘

```
#!/usr/bin/env ruby

$x = "Global"

def lokaleFunktion
    x = "Lokal"
    puts $x
    puts x
    puts defined? $x
    puts defined? x
end

lokaleFunktion()
puts defined? $x
puts defined? x
```

```
Default
schiller:ruby Max$ ruby variabletest2.rb
Global
Lokal
global-variable
local-variable
global-variable
nil
schiller:ruby Max$
```

Alles ist ein Objekt

- Alles ist ein Objekt (ohne Ausnahme!)
- Zuweisung sind immer Referenzen

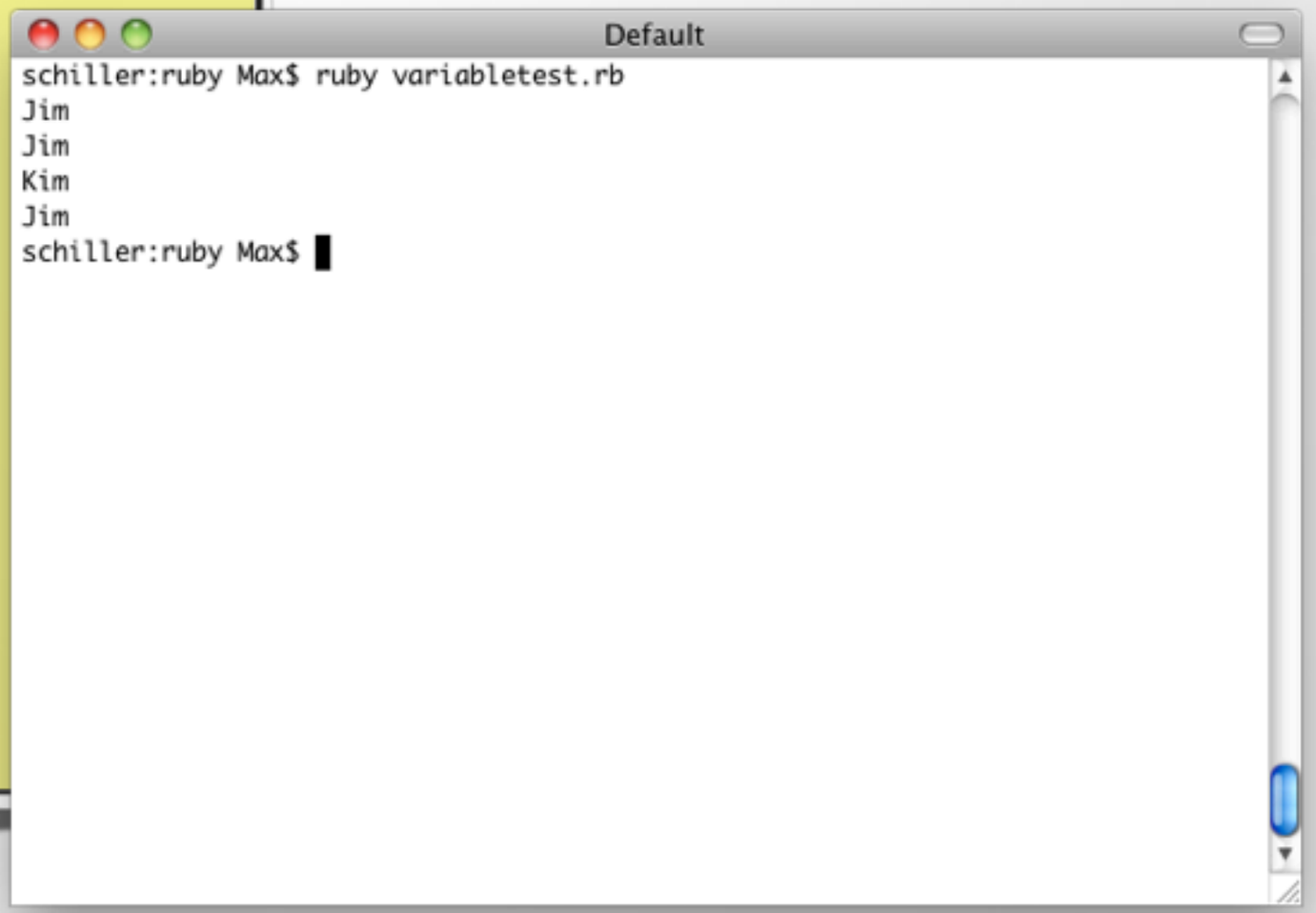
```
#!/usr/bin/env ruby

person1 = "Kim"
person2 = person1
person2[0] = "J"

puts person1
puts person2

person1 = "Kim"
person2 = person1.dup
person2[0] = "J"

puts person1
puts person2
```



„Duck typing“

- Die Parameter und Methoden bestimmen ob Objekte kompatibel sind. Nicht Interfaces oder Oberklassen

```ruby
#!/usr/bin/env ruby
class Ente
  def beschreibung
    "Eine graue Ente"
  end
  def sprechen
    "Quak!"
  end
end

class Kuh
  def beschreibung
    "Eine dicke Kuh"
  end
  def sprechen
    "Muuuh!"
  end
end

def lass_sprechen(tier)
  puts tier.beschreibung+" macht "+tier.sprechen
end

lass_sprechen(Ente.new)
lass_sprechen(Kuh.new)
```

```
Default
Max-Maurers-MacBook:ruby Max$ ruby duck.rb
Eine graue Ente macht Quak!
Eine dicke Kuh macht Muuuh!
Max-Maurers-MacBook:ruby Max$
```

Klassen

- sehr einfach Klassendefinition
- Instanzvariablen werden mit @ deklariert Klassenvariablen mit @@

```ruby
#!/usr/bin/env ruby
class Human
    @@population = 0
    def initialize(_name)
        @@population = @@population+  1
        @popNum = @@population
        @name = _name
        @age=0
    end

    def birthday()
        @age = @age + 1
        puts ""+ @name + " is %d year(s) old." % @age
        puts "He or she is the %d human!" % @popNum
    end
end
```

```
Default
schiller:ruby Max$ ruby klassen.rb
Sarah is 1 year(s) old.
He or she is the 1 human!
Mike is 1 year(s) old.
He or she is the 2 human!
Sarah is 2 year(s) old.
He or she is the 1 human!
schiller:ruby Max$
```

Offene Klassen

- Klassen können jederzeit neue Methoden lernen

```
#!/usr/bin/ruby
# re-open Ruby's Time class
class Time
  def yesterday
    self - 86400
  end
end

today = Time.now
yesterday = today.yesterday+1200 2008
puts today;
puts yesterday;
```

```
Default
schiller:ruby Max$ ruby openclasses.rb
Sun Nov 08 09:10:27 +0100 2009
Sat Nov 07 09:10:27 +0100 2009
schiller:ruby Max$
```

Variablenrückblick

- Variablen erhalten ihre Bedeutung durch Position und Namen
- Abfrage über ‚defined?‘

Name beginnt mit	Variablenart
$	Globale Variable
@	Instanzvariable
@@	Klassenvariable
[a-z_]	Lokale Variable
[A-Z]	Konstante

Error Handling

- Fehlerbehandlung über per rescue
- Fehler werden mit raise geworfen und müssen von Exception erben

```
#!/usr/bin/env ruby
def double(number)
      if !number.is_a?(Fixnum)
            raise ArgumentError.new("No number!")
      end
      return number*2
end

begin
      puts double(2)
      puts double("Hallo")
      puts double(5)
rescue ArgumentError
      puts "Sorry for that wrong argument"
else
      puts "No error happend"
ensure
      puts "This happens always"
end

begin
      puts double(2)
      puts double(5)
rescue ArgumentError
      puts "Sorry for that wrong argument"
else
      puts "No error happend"
ensure
      puts "This happens always"
end
```

```
Default
schiller:ruby Max$ ruby errors.rb
4
Sorry for that wrong argument
This happens always
4
10
No error happend
This happens always
schiller:ruby Max$
```

Bibliotheken

- Achtung: In Ruby Unterschied zwischen Modul und Bibliothek
- Module ähnlich zu Interfaces aber mit Funktionalität
- Bibliotheken verfügbar über RubyGems (muss nachinstalliert werden)

```
Default
Max-Maurers-MacBook:htdocs Max$ sudo gem install rake
Successfully installed rake-0.8.7
1 gem installed
Installing ri documentation for rake-0.8.7...
Installing RDoc documentation for rake-0.8.7...
Max-Maurers-MacBook:htdocs Max$
```

Eigene Bibliotheken

- Module können sehr komplex werden
- Prinzipiell aber auch eine normale Ruby Datei sein

```
puts "test_library included"

class TestClass
  def initialize
    puts "TestClass object created"
  end
end
```

```
#!/usr/bin/env ruby
require 'test_library.rb'

t = TestClass.new
```

```
Default
Max-Maurers-MacBook:ruby Max$ ruby libtest.rb
test_library included
TestClass object created
Max-Maurers-MacBook:ruby Max$
```

Module

- Module vererben Ihre Funktionen

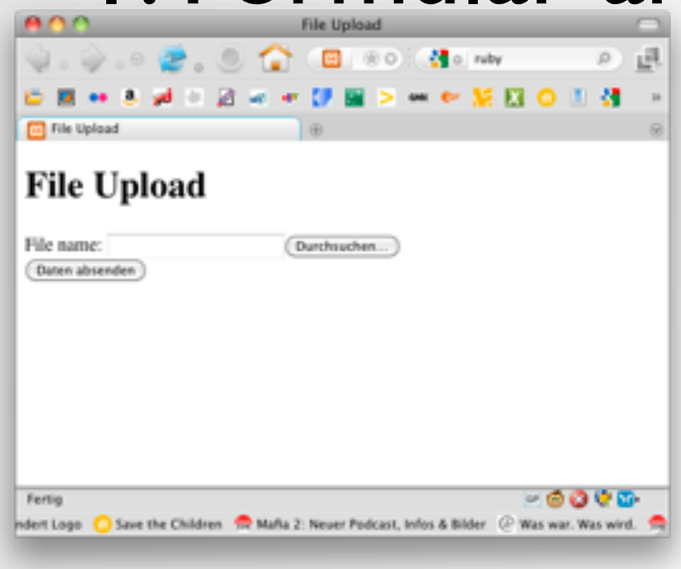
```
#!/usr/bin/env ruby
module Debug
  def whoAmI?
    "#{self.class.name} (\##{self.object_id}): #{self.to_s}"
  end
end
class Phonograph
  include Debug
     def initialize(s)
          @name = s
     end
     def to_s()
          return @name
     end
end
class EightTrack
  include Debug
     def initialize(s)
          @name = s
     end
     def to_s()
          return @name
     end
end
ph = Phonograph.new("West End Blues")
et = EightTrack.new("Surrealistic Pillow")
puts ph.whoAmI?
puts et.whoAmI?
```

```
Default
Max-Maurers-MacBook:ruby Max$ ruby moduleexample.rb
Phonograph (#2148247760): West End Blues
EightTrack (#2148247720): Surrealistic Pillow
Max-Maurers-MacBook:ruby Max$
```

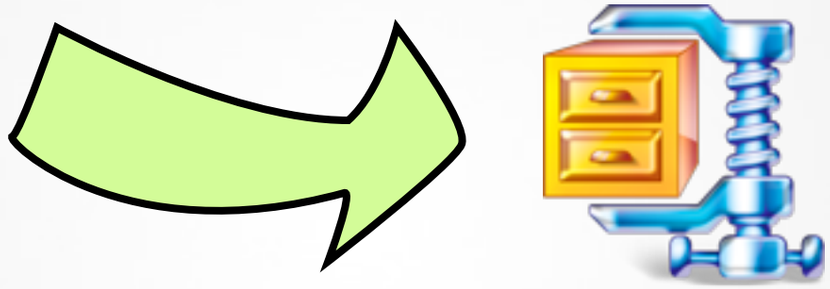
Die Aufgabe

Galerie Baukasten

1. Formular anzeigen

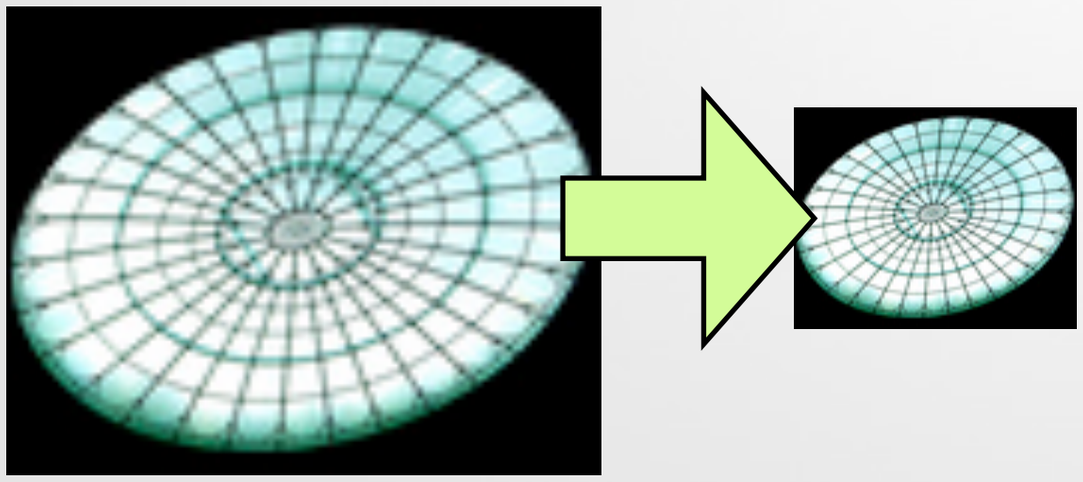


2. Zip-Datei hochladen



3. Entpacken

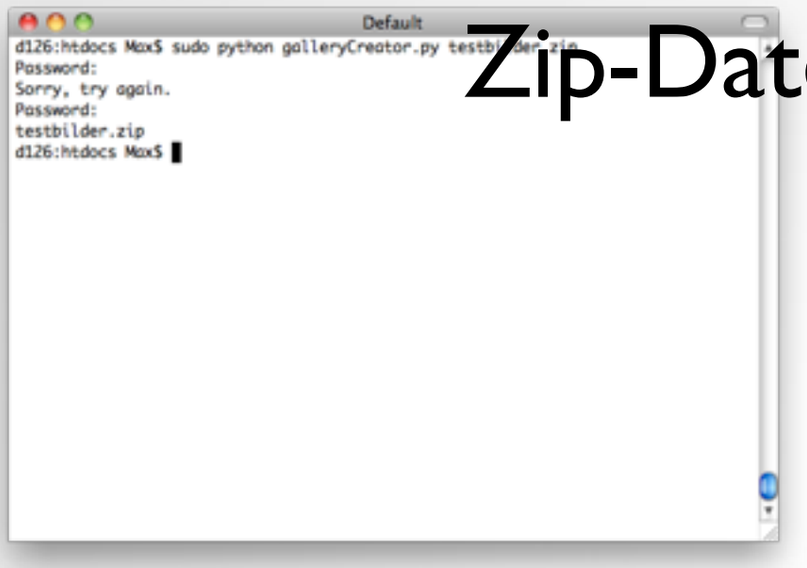
4. Thumbnails rendern



5. Gallerie Seite anzeigen

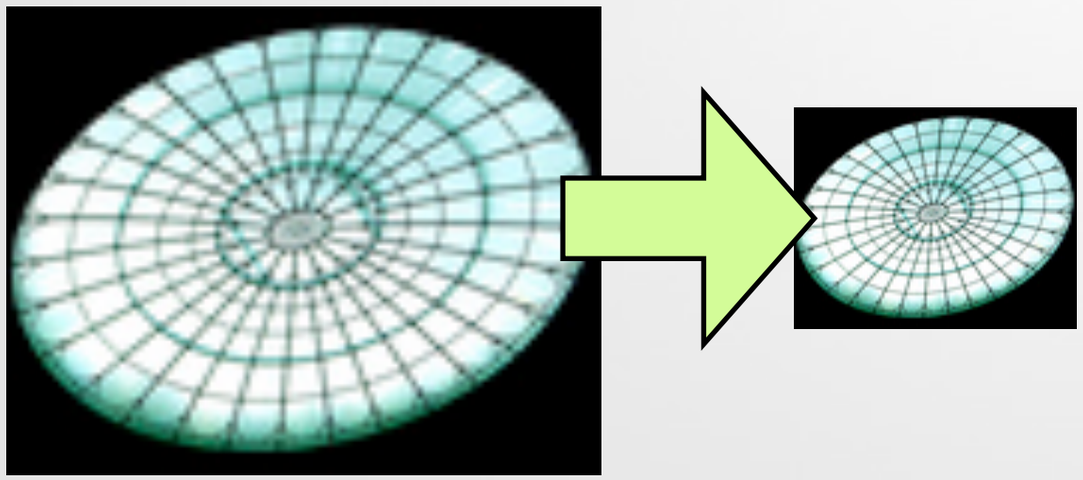
Kommandozeilenversion

1. Kommandozeilenaufruf mit Zip-Datei



2. Entpacken

3. Thumbnails rendern



4. Gallerie Seite erzeugen

Grundskript und Kommandozeile

Grundskript

```
#!/usr/bin/env ruby

HEADER = "content-type: text/html\n\n"

def error(text)
    puts HEADER
    puts "<html><body><h1>"+text+"</h1></body></html>"
    exit()
end

if ARGV.length>=1
    # wir haben ein command line argument!
    puts "Command line"
    exit()
else
    puts HEADER
    puts HTML_TEMPLATE
end
```

HTML-Formulare

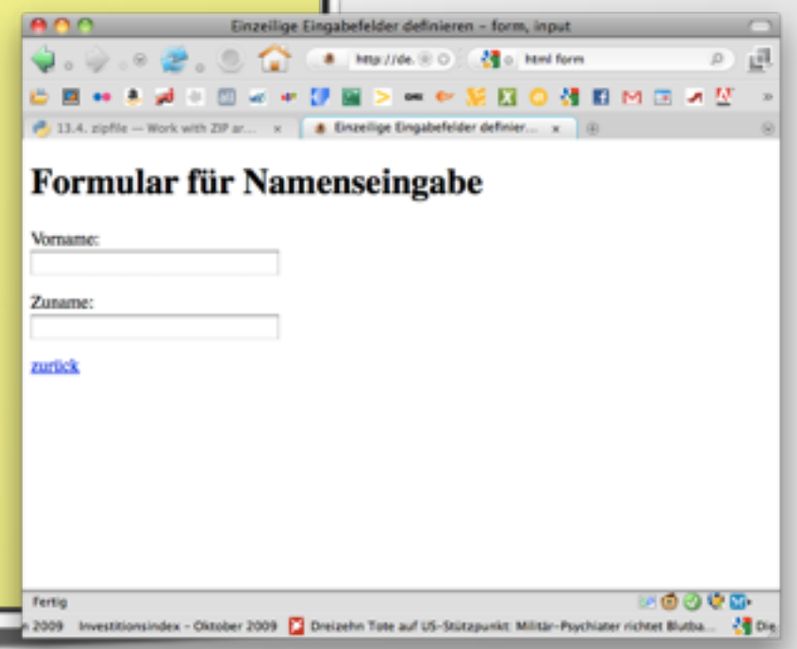
HTML-Formulare

- Darstellung verschiedener Eingabemöglichkeiten
  - Eingabefelder, Dropdown, Checkbox, Radio Button, Datei Upload, TextArea, Button

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
       "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Einzeilige Eingabefelder definieren</title>
</head>
<body>

<h1>Formular f&uuml;r Namenseingabe</h1>

<form action="input_text.htm" method="post">
  <p>Vorname:<br><input name="vorname" type="text" size="30"
maxlength="30"></p>
  <p>Zuname:<br><input name="zuname" type="text" size="30"
maxlength="40"></p>
</form>
```



Formulardaten allgemein

- Zwei Methoden: GET oder POST
- GET
  - Übergabe über die URL: „test.py?action=hallo&var1=wert1“
  - Direkt sichtbar und manipulierbar. Variablen bleiben bei Copy&Paste in E-Mails z.B. erhalten (z.B. Google Maps)
- POST
  - Nicht im Browser sichtbar auch nicht im Browser Cache gespeichert, werden im Anfrage-Header von HTTP übergeben

Formulardaten in Ruby

- Modul ‚cgi‘:
- Zugriff unabhängig von GET und POST
- verwendet man ein POST Formular ist gleichzeitiger Zugriff auf GET Parameter nicht mehr möglich

```
#!/usr/bin/env ruby
require 'cgi'
$cgi = CGI.new()  # New CGI object
$cgi.params.keys.each do |key|
   puts key+":"+$cgi.params[key].first.to_s
end
```

```
Default
schiller:ruby Max$ ruby rbCgi.rb
(offline mode: enter name=value pairs on standar
arg1=value1
arg2=value2
argX=valueX
argX:valueX
arg1:value1
arg2:value2
schiller:ruby Max$
```

Die ‚cgi'-Bibliothek

- Bietet HTML-spezifische Funktionen (Formulare, Cookies, Header)
- Wichtige Parameter und Funktionen
  - cgi.params enthält alle einen Hash aller übergeben Formularwerte
  - cgi.cookies() gibt eine Liste aller für diesen Server gesetzter Cookies zurück
  - cgi.out() { inhalt } erzeugt je nach Inhalt einen gültigen HTTP-Header und schickt den Inhalt an den Browser

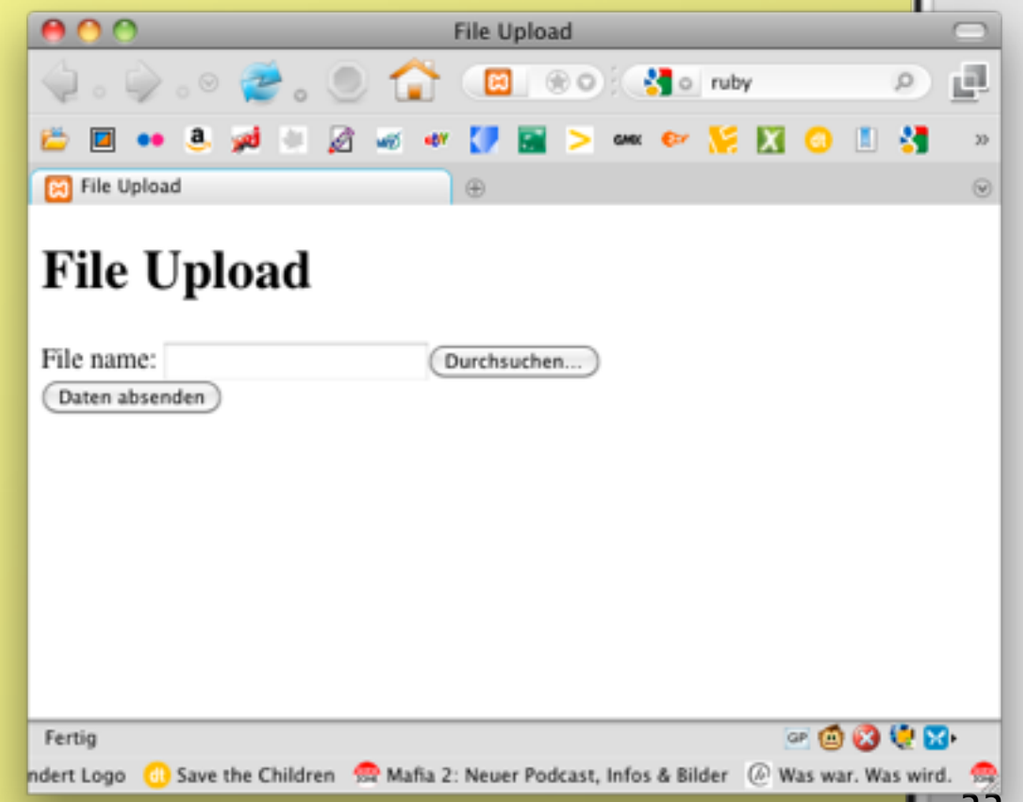
Datei-Upload

Formular ausgeben

```
HTML_TEMPLATE = <<eot
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<html><head><title>File Upload (Ruby)</title>
</head><body><h1>File Upload (Ruby)</h1>
<form action="" method="post" enctype="multipart/form-data">
<input type="hidden" name="action" value="upload"/>
File name: <input name="file" type="file"/><br/>
<input name="submit" type="submit"/>
</form>
</body>
</html>
eot

HEADER = "content-type: text/html\n\n"

puts HEADER
puts HTML_TEMPLATE
```



Upload auslesen

- Gelesene Datei wird je nach Größe als String oder als temporäre Datei gespeichert

```
require 'cgi'
$cgi = CGI.new()  # New CGI object
UPLOAD_DIR = "./tmp"

def processFile(form_field)
     fileitem = $cgi.params[form_field].first
     filepath = UPLOAD_DIR+"/"+fileitem.original_filename()
     File.open(filepath.untaint, 'w') { |file| file << fileitem.read}
     return filepath
end

if $cgi.has_key?('file')
   filepath = processFile('file')
end
```

ZIP-Datei entpacken

Zugriff auf ZIP-Dateien

```
#!/usr/bin/env python
require 'zip/zip'
require 'zip/zipfilesystem'

zipFile = Zip::ZipFile.open(file)
zipFile.each do |single_file|
  single_file.extract()
end
```

Das ‚zip'-Modul

- Umgang mit ZIP-Dateien
- Arbeitsweise wie mit einem Dateisystem
- Wichtige Parameter und Funktionen
  - Zip::ZipFile.open(file) öffnet einen Dateipfad der dann eine Liste von Zip-Dateien zurück
  - zipfile.get_output_stream(filename) gibt einen Outputstream zu einer neuen Datei in der ZIP-Datei zurück in den man schreiben kann
  - zipfile.mkdir(dirname) erzeugt in Verzeichnis innerhalb der ZIP-Datei
  - Auf einzelnen Dateien der ZIP-Datei kann zum Beispiel ‚extract' aufgerufen werden

Zugriff auf ZIP-Dateien

- Zip-Datei ‚flat' entpacken

```
#!/usr/bin/ruby
require 'zip/zip'
require 'zip/zipfilesystem'

def unzip(file, path)
    zipFile = Zip::ZipFile.open(file)
    zipFile.each do |single_file|
        next unless (single_file.name =~ /\/([^\/]*)$/)
        targetName = $1
        next if targetName =~ /^\./
    zipFile.extract(single_file, path+"/"+targetName) {true}
    end
end

unzip(filepath, UPLOAD_DIR)
```

Regular Expressions

- Zeichenketten prüfen und ändern
  - „/test/“ testet ob die Zeichenkette „test“ im Text vorkommt
  - „/[a-z]/“ testet ob ein kleiner Buchstabe im Text vorkommt
  - „/^[a-z]/“ oder „/[a-z]$“ kleiner Buchstabe am Anfang und am Ende
  - „/^[a-z]*$/“ Quantifikatoren (*,+,?): Test besteht aus keinem Zeichen oder nur aus Kleinbuchstaben
  - „/Anfang(.*)Ende/“: Group matching sucht alle Zeichen zwischen Anfang und Ende und speichert diese
  - „/^...$/“: Was ist das?
  - „/([^/]*)$“: und das?

Zugriff auf ZIP-Dateien

- Zip-Datei ‚flat' entpacken

```
#!/usr/bin/ruby
require 'zip/zip'
require 'zip/zipfilesystem'

def unzip(file, path)
    zipFile = Zip::ZipFile.open(file)
    zipFile.each do |single_file|
        next unless (single_file.name =~ /\/([^\/]*)$/)
        targetName = $1
        next if targetName =~ /^\./
    zipFile.extract(single_file, path+"/"+targetName) {true}
    end
end

unzip(filepath, UPLOAD_DIR)
```

Thumbnails schreiben

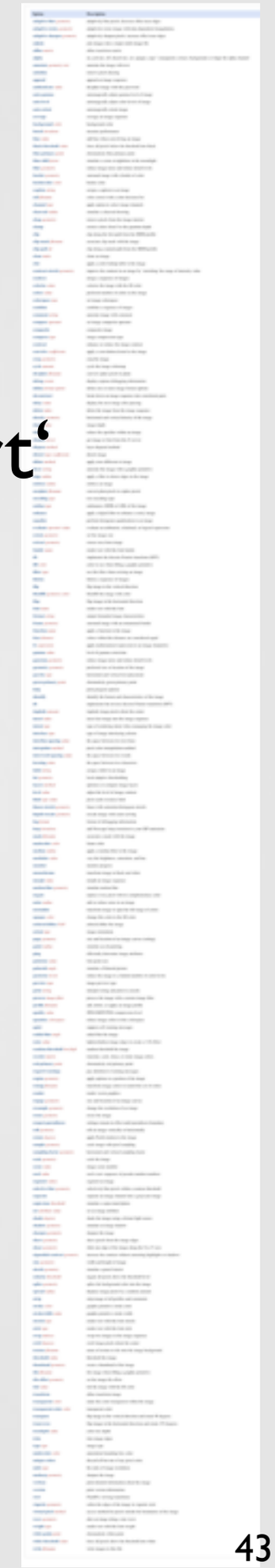
Thumbnails schreiben

- Intern per Modul
- Extern per Kommandozeilenaufruf (‚convert‘)

```
def createThumbnails(path)
    allFiles = Dir.entries(path)
    imageFiles = allFiles.find_all{|item| item =~ /\.jpg$/ && !(item =~ /^\./) && !(item =~ /_T\.jpg$/)}
    imageFiles.each do |f|
        inputfile = path+"/"+f
        f =~ /(.*)\.jpg$/
        outputfile = path+"/"+$1+"_T.jpg"
        cmd = "/usr/local/bin/convert -resize 100x100 "+inputfile+" "+outputfile
        File.new(inputfile).delete() if (!system(cmd))
    end
end
```

ImageMagick

- Kommandozeilen Bildverarbeitung mit umfangreichem Bildumwandlungstool ‚convert‘
- Dateiformate (> 100!)
  - Beispiele: AVI, BMP, JPEG, MPEG, PCX, PNG, PSD, SVG, TTF, WMF
- Optionen (s. rechts)
  - nur Einige: fill, rotate, resize, white-point
- Mehr Infos: www.imagemagick.org

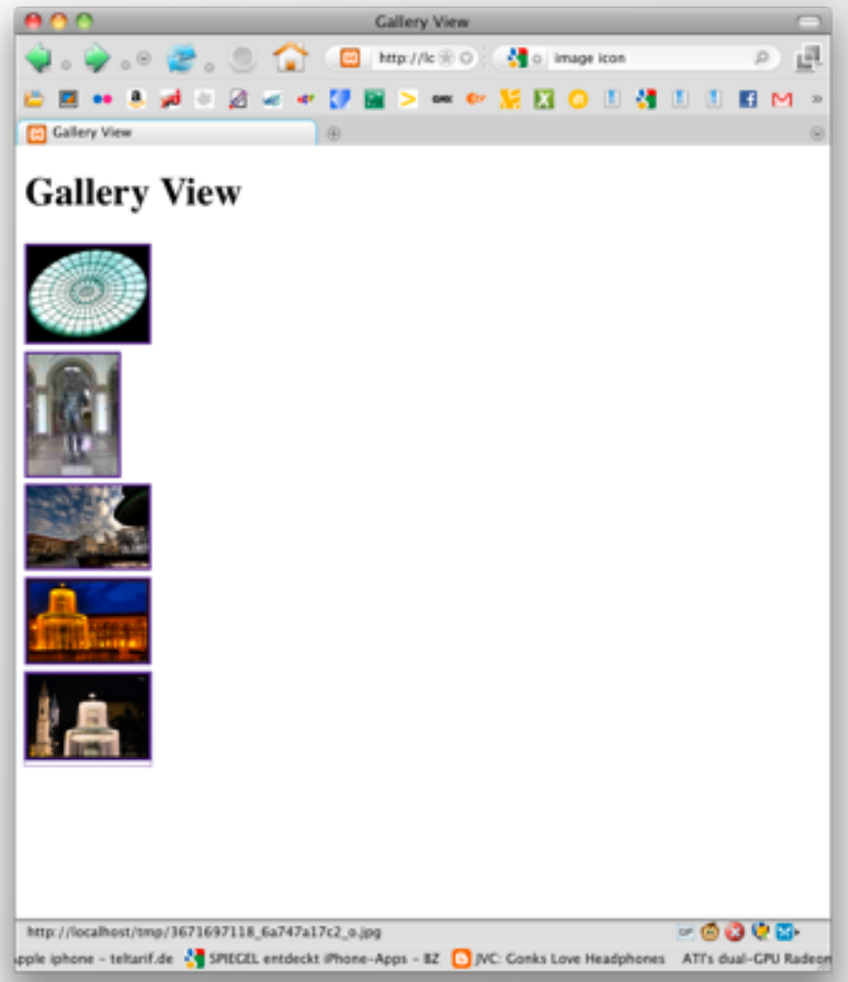


Gallery-Seite erzeugen

Thumbnails schreiben

- Seite mit Template bauen und alle gefunden Bilder einfügen

```
UPLOAD_TEMPLATE = <<eot
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<html><head><title>Gallery View (Ruby)</title>
</head><body><h1>Gallery View (Ruby)</h1>
eot
UPLOAD_DIR = "./tmp"
def generateWebsite(path)
     html = UPLOAD_TEMPLATE
     allFiles = Dir.entries(path)
     imageFiles = allFiles.find_all{|item| item =~ /\.jpg$/ && !(item
=~ /^\./) && !(item =~ /_T\.jpg$/)}
     imageFiles.each do |f|
          inputfile = path+"/"+f
          f =~ /(.*)\.jpg$/
          outputfile = path+"/"+$1+"_T.jpg"
          html += '<div class="image"><a href="'
          html += inputfile
          html += '"><img src="'
          html += outputfile
          html += '"/></a></div>'
     end
     html += """</body></html>"""
     return html
end
puts HEADER
puts generateWebsite(UPLOAD_DIR)
```



Zeit messen

Zeitmessung

- Nicht zwangsläufig notwendig für die Aufgabenstellung aber interessant zum Vergleichen
- Time.now() gibt eine genaue Zeit zurück

```
$start = Time.now()
# Aufwaendige Berechnung
$zeitpunkt1 = Time.now()-$start
# Aufwaendige Berechnung
$zeitpunkt2 = Time.now()-$start
puts „Gebrauchte Zeit (1): %f Sekunden“ % $zeitpunkt1
puts „Gebrauchte Zeit (2): %f Sekunden“ % $zeitpunkt2
```

Kompletter Code

Gallery Uploader Code

```
#!/usr/bin/ruby
$start = Time.now()
require 'rubygems'
require 'pathname'
Gem.path.push "/opt/local/lib/ruby/gems/1.8"
Pathname.new(__FILE__).realpath.to_s =~ /(.*)\/[^\/]*$/
APP_PATH = $1
# needed for html
require 'cgi'
require 'stringio'
# needed for zip
require 'fileutils'
require 'zip/zip'
require 'zip/zipfilesystem'
UPLOAD_DIR = "./tmp"

HTML_TEMPLATE = <<eot
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<html><head><title>File Upload (Ruby)</title>
</head><body><h1>File Upload (Ruby)</h1>
<form action="" method="post" enctype="multipart/form-data">
<input type="hidden" name="action" value="upload"/>
File name: <input name="file" type="file"/><br/>
<input name="submit" type="submit"/>
</form>
</body>
</html>
eot
```

Gallery Uploader Code

```
UPLOAD_TEMPLATE = <<eot
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<html><head><title>Gallery View (Ruby)</title>
</head><body><h1>Gallery View (Ruby)</h1>
eot

HEADER = "content-type: text/html\n\n"

def error(text)
     puts HEADER
     puts "<html><body><h1>"+text+"</h1></body></html>"
     exit()
end

def processFile(form_field)
     fileitem = $cgi.params[form_field].first
     filepath = UPLOAD_DIR+"/"+fileitem.original_filename()
     File.open(filepath.untaint, 'w') { |file| file << fileitem.read}
     return filepath
end

def unzip(file, path)
     zipFile = Zip::ZipFile.open(file)
     zipFile.each do |single_file|
          next unless (single_file.name =~ /\/([^\/]*)$/)
          targetName = $1
          next if targetName =~ /^\./
    zipFile.extract(single_file, path+"/"+targetName) {true}
     end
end
```

Gallery Uploader Code

```
def createThumbnails(path)
     allFiles = Dir.entries(path)
     imageFiles = allFiles.find_all{|item| item =~ /\.jpg$/ && !(item =~ /^\./) && !(item =~ /_T\.jpg
$/)}
     imageFiles.each do |f|
          inputfile = path+"/"+f
          f =~ /(.*)\.jpg$/
          outputfile = path+"/"+$1+"_T.jpg"
          cmd = "/usr/local/bin/convert -resize 100x100 "+inputfile+" "+outputfile
          File.new(inputfile).delete() if (!system(cmd))
     end
end
#
def generateWebsite(path)
     html = UPLOAD_TEMPLATE
     allFiles = Dir.entries(path)
     imageFiles = allFiles.find_all{|item| item =~ /\.jpg$/ && !(item =~ /^\./) && !(item =~ /_T\.jpg
$/)}
     imageFiles.each do |f|
          inputfile = path+"/"+f
          f =~ /(.*)\.jpg$/
          outputfile = path+"/"+$1+"_T.jpg"
          html += '<div class="image"><a href="'
          html += inputfile
          html += '"><img src="'
          html += outputfile
          html += '"/></a></div>'
     end
     html += "Store file: %f seconds<br/>" % $timeUpload
     html += "Unzip file: %f seconds<br/>" % ($timeUnzip-$timeUpload)
     html += "Create thumbnails: %f seconds<br/>" % ($timeThumbnails-$timeUnzip)
     html += "Overall: %f seconds<br/>" % $timeThumbnails
     html += """</body></html>"""
     return html
end
```

Gallery Uploader Code

```
if ARGV.length>=1
     # wir haben ein command line argument!
     puts "Command line"
     filepath = ARGV[0]
     $timeUpload = Time.now()-$start
     unzip(filepath, UPLOAD_DIR)
     $timeUnzip = Time.now()-$start
     createThumbnails(UPLOAD_DIR)
     $timeThumbnails = Time.now()-$start
     html = generateWebsite(UPLOAD_DIR)
     File.open("gallery.html", 'w') {|f| f.write(html) }
     exit()
else
     $cgi = CGI.new()  # New CGI object
     if $cgi.has_key?('file')
          filepath = processFile('file')
          $timeUpload = Time.now()-$start
          unzip(filepath, UPLOAD_DIR)
          $timeUnzip = Time.now()-$start
          createThumbnails(UPLOAD_DIR)
          $timeThumbnails = Time.now()-$start
          puts HEADER
          puts generateWebsite(UPLOAD_DIR)
          exit()
     end

     puts HEADER
     puts HTML_TEMPLATE
end
```

Nächste Woche: Python

Literatur

- Dave Thomas, with Chad Fowler and Andy Hunt: Programming Ruby (www.rubycentral.com/book)
- Lucas Carlson & Leonard Richardson: Python Cookbook
- GUI Toolkits für Ruby (http://home.arcor.de/scite/index.html)
- Techtopia: Ruby Variable Scope (http://www.techotopia.com/index.php/Ruby_Variable_Scope)
- Wikipedia: Ruby (Programmiersprache)

Bildnachweis

- http://www.creativeuncut.com/gallery-07/art/mlpit-mario-baby-hammer.jpg
- http://www.dotolearn.com/picturecards/images/imageschedule/proud_l.gif
- http://www.hakstpoelten.ac.at/buchoase/Buch.gif
- http://school.discoveryeducation.com/clipart/images/arteasel4c.gif
- http://www.gg-schnitt.at/wp-content/uploads/2009/01/lego_brick.png
- http://www.helliot.com/cms/images/stories/logo/logo_school.gif
- http://www.clker.com/clipart-window-icon.html
- http://globaleuropeans.com/uploads/images/GE_global%20projects%202.jpg
- http://www.sbac.edu/~tpl/clipart/Animals%20and%20Insects/bug%20cartoon%2002.jpg
- http://jasoncirillo.files.wordpress.com/2009/03/pong.jpg
- http://www.leuchtfeuer-preetz.de/bilder/bibel.gif
- http://upload.wikimedia.org/wikipedia/commons/c/c3/PerlmuttAusst.jpg
- http://www.schmuckmarkt.ch/media/schmuckwissen/schmuckwissen_rubin/rubin.jpg