Themenabend Ruby

Sven & Astro

March 28, 2006

- **Ruby** has simple syntax, partially inspired by Eiffel and Ada.
- **Ruby** has exception handling features, like Java or Python, to make it easy to handle errors.
- **Ruby's** operators are syntax sugar for the methods. You can redefine them easily.
- **Ruby** is a complete, full, pure object oriented language: OOL. This means all data in **Ruby** is an object, in the sense of Smalltalk: no exceptions. Example: In **Ruby**, the number 1 is an instance of class Fixnum.
- **Ruby's** OO is carefully designed to be both complete and open for improvements. Example: **Ruby** has the ability to add methods to a class, or even to an instance during runtime. So, if needed, an instance of one class *can* behave differently from other instances of the same class.
- **Ruby** features single inheritance only, *on purpose*. But **Ruby** knows the concept of modules (called Categories in Objective-C). Modules are collections of methods. Every class can import a module and so gets all its methods for free. Some of us think that this is a much clearer way than multiple inheritance, which is complex, and not used very often compared with single inheritance (don't count C++ here, as it has often no other choice due to strong type checking!).
- **Ruby** features true closures. Not just unnamed function, but with present variable bindings.
- **Ruby** features blocks in its syntax (code surrounded by '{' ... '}' or 'do' ... 'end'). These blocks can be passed to methods, or converted into closures.
- **Ruby** features a true mark-and-sweep garbage collector. It works with all **Ruby** objects. You don't have to care about maintaining reference counts in extension libraries. This is better for your health. ;-)
- Writing C extensions in **Ruby** is easier than in Perl or Python, due partly to the garbage collector, and partly to the fine extension API. SWIG interface is also available.
- Integers in **Ruby** can (and should) be used without counting their internal representation. There *are* small integers (instances of class Fixnum) and large integers (Bignum), but you need not worry over which one is used currently. If a value is small enough, an integer is a Fixnum, otherwise it is a Bignum. Conversion occurs automatically.
- **Ruby** needs no variable declarations. It uses simple naming conventions to denote the scope of variables. Examples: simple 'var' = local variable, '@var' = instance variable, '$var' = global variable. So it is also not necessary to use a tiresome 'self.' prepended to every instance member.
- **Ruby** can load extension libraries dynamically if an OS allows.
- **Ruby** features OS independent threading. Thus, for all platforms on which **Ruby** runs, you also have multithreading, regardless of if the OS supports it or not, even on MS-DOS! ;-)
- **Ruby** is highly portable: it is developed mostly on Linux, but works on many types of UNIX, DOS, Windows 95/98/Me/NT/2000/XP, MacOS, BeOS, OS/2, etc.

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Herkunft

- „Perl“: Perle
- „Ruby“: Rubin
- *„Ruby ist eine objektorientierte, interpretierte Programmiersprache. Sie hat ihre Wurzeln in Perl, Smalltalk, Python, LISP, Bash und CLU.“* Wikipedia[1]

[1] http://de.wikipedia.org/wiki/Ruby_%28Programmiersprache%29

Entwicklung

- Februar 1993: Yukihiro „Matz“ Matsumoto beginnt Entwicklung
- Dezember 1995: Erste Veröffentlichung

Entwicklung

- Februar 1993: Yukihiro „Matz“ Matsumoto beginnt Entwicklung
- Dezember 1995: Erste Veröffentlichung
- `$ ruby -v`
  `ruby 1.8.4 (2005-12-24) [i386-freebsd6]`

Entwicklung

- Februar 1993: Yukihiro „Matz“ Matsumoto beginnt Entwicklung
- Dezember 1995: Erste Veröffentlichung
- `$ ruby -v`
  `ruby 1.8.4 (2005-12-24) [i386-freebsd6]`
- Ruby 1.9: Rite, Ruby2

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Hilfe?

- www.ruby-lang.org
- „Programming Ruby“
  http://www.rubycentral.com/book

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

irb – Interactive Ruby

- Um schnell etwas auszuprobieren

irb – Interactive Ruby

- Um schnell etwas auszuprobieren
- Toller `bc(1)`-Ersatz

ri

- Integrierte Dokumentation

ri

- Integrierte Dokumentation
- Lässt sich mit Rdoc erstellen

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Variablen

- Lokale Variablen: `v = 23`

Variablen

- Lokale Variablen: `v = 23`
- Instanzvariablen: `@v = 23`

Variablen

- Lokale Variablen: `v = 23`
- Instanzvariablen: `@v = 23`
- Klassenvariablen: `@@v = 23`

Variablen

- Lokale Variablen: `v = 23`
- Instanzvariablen: `@v = 23`
- Klassenvariablen: `@@v = 23`
- Globale Variablen: `$v = 23`
  Zum Beispiel: `$stdin`, `$stdout`, `$stderr`

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Alles ist ein Objekt

- `mein_string = "Hallo"`
  `=> "Hallo"`

Alles ist ein Objekt

- `mein_string = "Hallo"`
  `=> "Hallo"`
- `mein_string.reverse`
  `=> "ollaH"`
- `"Hallo".reverse`
  `=> "ollaH"`

Alles ist ein Objekt

- `mein_string = "Hallo"`
  `=> "Hallo"`
- `mein_string.reverse`
  `=> "ollaH"`
- `"Hallo".reverse`
  `=> "ollaH"`
- `mein_string => "Hallo"`

Alles ist ein Objekt

- `mein_string = "Hallo"`
  `=> "Hallo"`
- `mein_string.reverse`
  `=> "ollaH"`
- `"Hallo".reverse`
  `=> "ollaH"`
- `mein_string => "Hallo"`
- `mein_string.reverse!  => "ollaH"`

Alles ist ein Objekt

- `mein_string = "Hallo"`
  `=> "Hallo"`
- `mein_string.reverse`
  `=> "ollaH"`
- `"Hallo".reverse`
  `=> "ollaH"`
- `mein_string => "Hallo"`
- `mein_string.reverse!  => "ollaH"`
- `mein_string => "ollaH"`

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Konventionen

- Module, Klassen, Konstanten, Klassenmethoden, Modulmethoden:
  `Net::HTTP`, `Thread::stop` $\Leftrightarrow$ `Thread.stop`

Konventionen

- Module, Klassen, Konstanten, Klassenmethoden, Modulmethoden:
  `Net::HTTP`, `Thread::stop` $\Leftrightarrow$ `Thread.stop`
- Instanzmethoden:
  `String#upcase`, `Array#collect`

Konventionen

- Module, Klassen, Konstanten, Klassenmethoden, Modulmethoden:
  `Net::HTTP`, `Thread::stop` $\Leftrightarrow$ `Thread.stop`
- Instanzmethoden:
  `String#upcase`, `Array#collect`
- Modul- und Klassennamen in `CamelCase`

Konventionen (Methodennamen)

- Methodennamen kleingeschrieben: `Array#each_with_index`

Konventionen (Methodennamen)

- Methodennamen kleingeschrieben: `Array#each_with_index`
- Funktion mit booleschem Ergebnis:
  `heuhaufen.include? nadel`
  `IO#closed?`

Konventionen (Methodennamen)

- Methodennamen kleingeschrieben: `Array#each_with_index`
- Funktion mit booleschem Ergebnis:
  `heuhaufen.include? nadel`
  `IO#closed?`
- *Gefährliche* Funktionen: `String#reverse!`, `Array#collect!`

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

- `zeilenverzeichnis = "Zeile 1\nZeile 2"`

p

- `zeilenverzeichnis = "Zeile 1\nZeile 2"`
- `p zeilenverzeichnis`
  `"Zeile 1\nZeile 2"`

p

- `zeilenverzeichnis = "Zeile 1\nZeile 2"`
- `p zeilenverzeichnis`
  `"Zeile 1\nZeile 2"`
- `puts zeilenverzeichnis.inspect`
  `"Zeile 1\nZeile 2"`

p

- `zeilenverzeichnis = "Zeile 1\nZeile 2"`
- `p zeilenverzeichnis`
  `"Zeile 1\nZeile 2"`
- `puts zeilenverzeichnis.inspect`
  `"Zeile 1\nZeile 2"`
- `p zeilenverzeichnis.inspect.class`
  `String`

p

- `zeilenverzeichnis = "Zeile 1\nZeile 2"`
- `p zeilenverzeichnis`
  `"Zeile 1\nZeile 2"`
- `puts zeilenverzeichnis.inspect`
  `"Zeile 1\nZeile 2"`
- `p zeilenverzeichnis.inspect.class`
  `String`
- `p zeilenverzeichnis.inspect`
  `"\"Zeile 1\\nZeile 2\""`

p

- `a = [11, 'X']`

p

- `a = [11, 'X']`
- `p a`
  `[11, "X"]`

p

- `a = [11, 'X']`
- `p a`
  `[11, "X"]`
- `p meine_instanz`
  `#<MeineKlasse:0x82690b4 @variable2="discordia", @variable1="eris">`

inspect selbst definieren

```
class << meine_instanz
  def inspect
    "<#self.class(#@variable1,#@variable2)>"
  end
end
```

inspect selbst definieren

- ```
  class << meine_instanz
    def inspect
      "<#self.class(#@variable1,#@variable2)>"
    end
  end
  ```
- ```
  p meine_instanz
  <MeineKlasse(eris,discordia)>
  ```

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Primitive Datentypen

- `nil.class`
  `=> NilClass`

Primitive Datentypen

- `nil.class`
  `=> NilClass`
- `false.class`
  `=> FalseClass`

Primitive Datentypen

- `nil.class`
  `=> NilClass`
- `false.class`
  `=> FalseClass`
- `true.class`
  `=> TrueClass`

Strings

- `'Hier müssen nur \\ und \' escaped werden.'`

Strings

- `'Hier müssen nur \\ und \' escaped werden.'`
- `"Hier können wir auch anderes escapen\n"`

Strings

- 'Hier müssen nur \\ und \' escaped werden.'
- "Hier können wir auch anderes escapen\n"
- "und Code einbetten: #{ENV['PATH']}"

Nummern

- `23.class`
  `=> Fixnum`

Nummern

- `23.class`
  `=> Fixnum`
- `23.5.class`
  `=> Float`

Nummern

- `23.class`
  `=> Fixnum`
- `23.5.class`
  `=> Float`
- `(5**100).to_s.size`
  `=> 32`
  `(5**100).class`
  `=> Bignum`

Arrays

- `mein_array = [5, 23, 'ROOby']`
- `mein_array`
  `=> [5, 23, 'ROOby']`

Arrays

- `mein_array = [5, 23, 'ROOby']`
- `mein_array`
  `=> [5, 23, 'ROOby']`
- `mein_array[1]`
  `=> 23`

Arrays

- `mein_array = [5, 23, 'R00by']`
- `mein_array`
  `=> [5, 23, 'R00by']`
- `mein_array[1]`
  `=> 23`
- `mein_array.reverse`
  `=> ['R00by', 23, 5]`

Arrays

- ```
  mein_array.each do |element|
    puts element
  end
  ```
  ```

Arrays

- ```
  mein_array.each do |element|
    puts element
  end
  ```
- ```
  mein_array.each { |element|
    puts element
  }
  ```

Arrays

- ```
  mein_array.each do |element|
    puts element
  end
  ```
- ```
  mein_array.each { |element|
    puts element
  }
  ```
- ```
  5
  23
  ROOby
  ```
  ```

Arrays

- ```
  mein_array.each do |element|
    puts element
  end
  ```
- ```
  mein_array.each { |element|
    puts element
  }
  ```
- ```
  5
  23
  R00by
  ```
- ```
  mein_array.each_with_index { |element,index|
    puts "#{index}:  #{element}"
  }
  ```

Arrays

- ```
  mein_array.each do |element|
    puts element
  end
  ```
- ```
  mein_array.each { |element|
    puts element
  }
  ```
- ```
  5
  23
  R00by
  ```
- ```
  mein_array.each_with_index { |element,index|
    puts "#{index}:  #{element}"
  }
  ```
- ```
  0:  5
  1:  23
  2:  R00by
  => [5, 23, "R00by"]
  ```
  ```

Arrays und Variablenzuweisung

- `a = 'Hund', 'Katze'`
  `=> ["Hund", "Katze"]`

Arrays und Variablenzuweisung

- `a = 'Hund', 'Katze'`
  `=> ["Hund", "Katze"]`
- `hund, katze = a`
  `=> ["Hund", "Katze"]`
- `hund`
  `=> "Hund"`
- `katze`
  `=> "Katze"`

Arrays und Variablenzuweisung

- `a = 'Hund', 'Katze'`
  `=> ["Hund", "Katze"]`
- `hund, katze = a`
  `=> ["Hund", "Katze"]`
- `hund`
  `=> "Hund"`
- `katze`
  `=> "Katze"`
- `hund, katze = 'Hund', 'Katze'`
  `=> ["Hund", "Katze"]`
- `hund`
  `=> "Hund"`
- `katze`
  `=> "Katze"`

Arrays und Variablenzuweisung

- `a = 'Hund', 'Katze'`
  `=> ["Hund", "Katze"]`
- `hund, katze = a`
  `=> ["Hund", "Katze"]`
- `hund`
  `=> "Hund"`
- `katze`
  `=> "Katze"`
- `hund, katze = 'Hund', 'Katze'`
  `=> ["Hund", "Katze"]`
- `hund`
  `=> "Hund"`
- `katze`
  `=> "Katze"`
- Inhalt tauschen: `a, b = b, a`

Hashes

- `mein_hash = {'Mate' => 1.0, 'Datenschleuder' => 2.5}`
- `mein_hash`
  `=> {'Datenschleuder' => 2.5, 'Mate' => 1.0}`

Hashes

- `mein_hash = {'Mate' => 1.0, 'Datenschleuder' => 2.5}`
- `mein_hash`
  `=> {'Datenschleuder' => 2.5, 'Mate' => 1.0}`
- `mein_hash['Mate']`
  `=> 1.0`

Hashes

- `mein_hash = {'Mate' => 1.0, 'Datenschleuder' => 2.5}`
- `mein_hash`
  `=> {'Datenschleuder' => 2.5, 'Mate' => 1.0}`
- `mein_hash['Mate']`
  `=> 1.0`
- `mein_hash.keys`
  `=> ['Datenschleuder', 'Mate']`

Hashes (Iteratoren)

- ```
  mein_hash.each do |artikel,preis|
    puts "#{artikel} kostet #{preis} FRZ"
  end
  ```
  ```

Hashes (Iteratoren)

- ```
  mein_hash.each do |artikel,preis|
    puts "#{artikel} kostet #{preis} FRZ"
  end
  ```
- ```
  Datenschleuder kostet 2.5 FRZ
  Mate kostet 1.0 FRZ
  => {"Datenschleuder"=>2.5, "Mate"=>1.0}
  ```

Hashes (Iteratoren)

- ```
  mein_hash.each do |artikel,preis|
    puts "#{artikel} kostet #{preis} FRZ"
  end
  ```
- ```
  Datenschleuder kostet 2.5 FRZ
  Mate kostet 1.0 FRZ
  => {"Datenschleuder"=>2.5, "Mate"=>1.0}
  ```
- ```
  puts mein_hash.collect { |artikel,preis| "#{artikel}:
  #{preis}" }.join("\n")
  ```
  ```

Hashes (Iteratoren)

- ```
  mein_hash.each do |artikel,preis|
    puts "#{artikel} kostet #{preis} FRZ"
  end
  ```
- ```
  Datenschleuder kostet 2.5 FRZ
  Mate kostet 1.0 FRZ
  => {"Datenschleuder"=>2.5, "Mate"=>1.0}
  ```
- ```
  puts mein_hash.collect { |artikel,preis| "#{artikel}:
  #{preis}" }.join("\n")
  ```
- ```
  Datenschleuder:  2.5
  Mate:  1.0
  => nil
  ```

Ranges

- `mein_range = 5..23`
  `=> 5..23`

Ranges

- `mein_range = 5..23`
  `=> 5..23`
- `mein_range.include? 8`
  `=> true`
- `mein_range.include? 1`
  `=> false`

Ranges

- `mein_range = 5..23`
  `=> 5..23`
- `mein_range.include? 8`
  `=> true`
- `mein_range.include? 1`
  `=> false`
- ```
  case coolness
    when 0..20 then puts "Perl"
    when 21..60 then puts "Python"
    when 61..100 then puts "Ruby"
  end
  ```
  ```

Regexp

- `puts "Mentions Ruby" if text =~ /Ruby/`

Regexp

- `puts "Mentions Ruby" if text =~ /Ruby/`
- ```
  if 'Ruby is cool' =~ /(.+) is (.+)/
    puts "#{$1} ist #{$2}"
  end
  ```
- `Ruby ist cool`
  ```

Regexp

- `puts "Mentions Ruby" if text =~ /Ruby/`
- ```
  if 'Ruby is cool' =~ /(.+) is (.+)/
    puts "#{$1} ist #{$2}"
  end
  ```
- `Ruby ist cool`
- ```
  if m = r.match('Ruby is cool')
    p m.captures
  end
  ```
- `["Ruby", "cool"]`

Regexp

- `puts "Mentions Ruby" if text =~ /Ruby/`
- ```
  if 'Ruby is cool' =~ /(.+) is (.+)/
    puts "#{$1} ist #{$2}"
  end
  ```
- `Ruby ist cool`
- ```
  if m = r.match('Ruby is cool')
    p m.captures
  end
  ```
- `["Ruby", "cool"]`
- ```
  'Ruby is cool.  Sky is blue.'.scan(/ *(.+?)  is
  (.+?)\./)
  => [["Ruby", "cool"], ["Sky", "blue"]]
  ```
  ```

Symbols

- Symbole repräsentieren Namen oder Strings

Symbols

- Symbole repräsentieren Namen oder Strings
- ```
  a = "Ruby"
  b = "Ruby"
  a.object_id
    => 403028500
  b.object_id
    => 403010020
  ```
  ```

Symbols

- Symbole repräsentieren Namen oder Strings
- ```
  a = "Ruby"
  b = "Ruby"
  a.object_id
    => 403028500
  b.object_id
    => 403010020
  ```
- ```
  c = :Ruby
  d = :"Ruby"
  c.object_id
    => 4121870
  d.object_id
    => 4121870
  ```

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - **Kontrollstrukturen**
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

if

- ```
  if Ruby::is_cool?
    applause!
  end
  ```
  ```

if

- ```
  if Ruby::is_cool?
    applause!
  end
  ```
- ```
  if hacker.is_hacking?
    hacker.continue_hacking
  else
    hacker.start_hacking
  end
  ```

if

- ```
  if Ruby::is_cool?
    applause!
  end
  ```
- ```
  if hacker.is_hacking?
    hacker.continue_hacking
  else
    hacker.start_hacking
  end
  ```
- ```
  unless hacker.is_hacking?
    hacker.start_hacking
  else
    hacker.continue_hacking
  end
  ```
  ```

if

- ```
  if Ruby::is_cool?
    applause!
  end
  ```
- ```
  if hacker.is_hacking?
    hacker.continue_hacking
  else
    hacker.start_hacking
  end
  ```
- ```
  unless hacker.is_hacking?
    hacker.start_hacking
  else
    hacker.continue_hacking
  end
  ```
- Alternativ: `(condition ? then_clause : else_clause)`
  ```

while

- ```
  i = 0
  => 0
  ```
  ```

while

- ```
  i = 0
  => 0
  ```
- ```
  while i < 5
    puts (i += 1)
  end
  ```

while

- ```
  i = 0
  => 0
  ```
- ```
  while i < 5
    puts (i += 1)
  end
  ```
- ```
  1
  2
  3
  4
  5
  => nil
  ```
  ```

while

- ```
  i = 0
  => 0
  ```
- ```
  while i < 5
    puts (i += 1)
  end
  ```
- ```
  1
  2
  3
  4
  5
  => nil
  ```
- Abbruch: `break`
  ```

Hintendran

- `exit if user.wants_to_exit?`

Hintendran

- `exit if user.wants_to_exit?`
- `exit unless user.wants_to_go_on?`

Hintendran

- `exit if user.wants_to_exit?`
- `exit unless user.wants_to_go_on?`
- ```
  i = 0
  => 0
  ```
- ```
  begin
    puts(i+=1)
  end while i < 5
  ```

Hintendran

- `exit if user.wants_to_exit?`
- `exit unless user.wants_to_go_on?`
- ```
  i = 0
  => 0
  ```
- ```
  begin
    puts(i+=1)
  end while i < 5
  ```
- ```
  1
  2
  3
  4
  5
  => nil
  ```
  ```

case

```
case inputLine
  when "debug"
    dumpDebugInfo
    dumpSymbols
  when /p\s+(\w+)/
    dumpVariable($1)
  when "quit", "exit"
    exit
  else
    print "Illegal command:  #{inputLine}"
end
```

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - **Alles hat ein Ergebnis**
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Der ternäre Operator

- Bekannt, der ternäre Operator:
  `bedingung ? then_wert : else_wert`

Der ternäre Operator

- Bekannt, der ternäre Operator:

```
bedingung ?  then_wert :  else_wert
```

-
```
4.times { |zahl|
  puts "#{zahl} " +
  (zahl == 2 ?  'ist' :  'ist nicht') +
  " Zwei."
}
```

Der ternäre Operator

- Bekannt, der ternäre Operator:

```
bedingung ?  then_wert :  else_wert
```

-
```
4.times { |zahl|
  puts "#{zahl} " +
  (zahl == 2 ?  'ist' :  'ist nicht') +
  " Zwei."
}
```

-
```
0 ist nicht Zwei.
1 ist nicht Zwei.
2 ist Zwei.
3 ist nicht Zwei.
```

Alles hat ein Ergebnis

- ```
  variable = (if bedingung
      then_wert
    else
      else_wert
    end)
  ```
  ```

Alles hat ein Ergebnis

```
puts "Zufall " + case rand
     when 0..0.3 then "klein"
     when 0.3..0.7 then "mittel"
     when 0.7..1 then "gross"
     else "kaputt"
   end
```

Alles hat ein Ergebnis

- ```
  puts "Zufall " + case rand
       when 0..0.3 then "klein"
       when 0.3..0.7 then "mittel"
       when 0.7..1 then "gross"
       else "kaputt"
    end
  ```
- `Zufall gross`
  ```

Alles hat ein Ergebnis

- ```
  puts "Zufall " + case rand
       when 0..0.3 then "klein"
       when 0.3..0.7 then "mittel"
       when 0.7..1 then "gross"
       else "kaputt"
    end
  ```
- Zufall gross
- Zufall klein
  ```

Alles hat ein Ergebnis

- ```
  puts "Zufall " + case rand
       when 0..0.3 then "klein"
       when 0.3..0.7 then "mittel"
       when 0.7..1 then "gross"
       else "kaputt"
    end
  ```
- Zufall gross
- Zufall klein
- Zufall mittel
  ```

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - **Bibliotheken laden**
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Bibliotheken laden

- `require 'thread'`

Bibliotheken laden

- `require 'thread'`
- `$:` wird durchsucht

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - **Methoden**
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Methoden

- Haben immer einen Rückgabewert!
- ```
  def hallo_welt
    puts "Hallo Welt!"
  end
  ```
  ```

Methoden

- Haben immer einen Rückgabewert!
- ```
  def hallo_welt
    puts "Hallo Welt!"
  end
  ```
- ```
  hallo_welt()
  Hallo Welt!
  => nil
  ```

Methoden

- Haben immer einen Rückgabewert!
- ```
  def hallo_welt
    puts "Hallo Welt!"
  end
  ```
- ```
  hallo_welt()
  Hallo Welt!
  => nil
  ```
- ```
  hallo_welt
  Hallo Welt!
  => nil
  ```
  ```

Parameter

- ```
  def hallo(whom, bangs)
    puts "Hallo #{whom}#{'!'*bangs}"
  end
  ```
  ```

Parameter

- ```
  def hallo(whom, bangs)
    puts "Hallo #{whom}#{'!'*bangs}"
  end
  ```
- ```
  hallo 'Peter', 5
  Hallo Peter!!!!!
  => nil
  ```

Default-Parameter

- Normaler Parameter darf keinem Default-Parameter folgen!
- ```
  def hallo(whom='Eris')
    puts "Hallo #{whom}!"
  end
  ```
  ```

Default-Parameter

- Normaler Parameter darf keinem Default-Parameter folgen!
- ```
  def hallo(whom='Eris')
    puts "Hallo #{whom}!"
  end
  ```
- ```
  hallo 'Peter'
  Hallo Peter!
  => nil
  ```

Default-Parameter

- Normaler Parameter darf keinem Default-Parameter folgen!
- ```
  def hallo(whom='Eris')
    puts "Hallo #{whom}!"
  end
  ```
- ```
  hallo 'Peter'
  Hallo Peter!
  => nil
  ```
- ```
  hallo
  Hallo Eris!
  => nil
  ```
  ```

Variable Parameterliste

- ```
  def debug_parameters(*a)
    p a
  end
  ```
  ```

Variable Parameterliste

- ```
  def debug_parameters(*a)
    p a
  end
  ```
- ```
  debug_parameters(23, 5, 42)
  [23, 5, 42]
  ```
- Aha, ein Array!

Parameter als Hash

- ```
  def hash_me(hsh)
    p hsh
  end
  ```
  ```

Parameter als Hash

- ```
  def hash_me(hsh)
    p hsh
  end
  ```
- ```
  hash_me 'Datenschleuder'=>2.50, 'Aufkleber'=>1.00
  {"Aufkleber"=>1.0, "Datenschleuder"=>2.5}
  ```
- ```
  hash_me('Datenschleuder'=>2.50, 'Aufkleber'=>1.00)
  {"Aufkleber"=>1.0, "Datenschleuder"=>2.5}
  ```
  ```

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Blöcke

- Benutzung: `mein_array.each { |e| puts e }`

Blöcke

- Benutzung: `mein_array.each { |e| puts e }`
- ```
  def tu
    yield
  end
  ```
  ```

Blöcke

- Benutzung: `mein_array.each { |e| puts e }`
- ```
  def tu
    yield
  end
  ```
- `tu`
- ```
  LocalJumpError:  no block given
  from (irb):2:in ‘tu’
  from (irb):4
  from :0
  ```

Blöcke

- Benutzung: `mein_array.each { |e| puts e }`
- ```
  def tu
    yield
  end
  ```
- `tu`
- ```
  LocalJumpError:  no block given
  from (irb):2:in `tu'
  from (irb):4
  from :0
  ```
- `tu { puts 'Hallo' }`
- ```
  Hallo
  => nil
  ```
  ```

Blöcke implizit

- ```
  def tu
    yield if block_given?
  end
  ```
  ```

Blöcke implizit

- ```
  def tu
    yield if block_given?
  end
  ```
- ```
  tu
  => nil
  ```

Blöcke implizit

- ```
  def tu
    yield if block_given?
  end
  ```
- ```
  tu
  => nil
  ```
- ```
  tu { puts 'Hallo' }
  Hallo
  => nil
  ```
  ```

Blöcke explizit

- ```
  def tu(&block)
    block.call if block
  end
  ```
  ```

Blöcke explizit

- ```
  def tu(&block)
    block.call if block
  end
  ```
- ```
  tu
  => nil
  ```

Blöcke explizit

- ```
  def tu(&block)
    block.call if block
  end
  ```
- ```
  tu
  => nil
  ```
- ```
  tu { puts 'Hallo' }
  Hallo
  => nil
  ```
  ```

Blöcke merken

- `most_important_function = lambda { puts "Hello World!" }`

Blöcke merken

- `most_important_function = lambda { puts "Hello World!" }`
- `most_important_function.call`
- `Hello World!`
  `=> nil`

Blöcke merken

- `most_important_function = lambda { puts "Hello World!" }`
- `most_important_function.call`
- `Hello World!`
  `=> nil`
- `tu { most_important_function.call }`
- `Hello World!`
  `=> nil`

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Klassen

```
class Drink
  def initialize(name)
    @name = name
  end
  def drink
    puts "Drinking #{@name}"
  end
end
```

Vererben

```
class Mate < Drink
  def initialize
    super 'Club Mate'
  end
end
```

Vererben

- ```
  class Mate < Drink
    def initialize
      super 'Club Mate'
    end
  end
  ```
- ```
  m = Mate.new
  => #<Mate:0x8152664 @name="Club Mate">
  ```

Vererben

- ```
  class Mate < Drink
    def initialize
      super 'Club Mate'
    end
  end
  ```
- ```
  m = Mate.new
  => #<Mate:0x8152664 @name="Club Mate">
  ```
- ```
  m.drink
  Drinking Club Mate
  => nil
  ```
  ```

Getter

- ```
  class Drink
    attr_reader :name
  end
  ```
  ```

Getter

- ```
  class Drink
    attr_reader :name
  end
  ```
- ```
  m.name
  => "Club Mate"
  ```

Getter

- ```
  class Drink
    attr_reader :name
  end
  ```
- ```
  m.name
  => "Club Mate"
  ```
- ```
  m.name = 'B33r'
  NoMethodError:  undefined method `name=' for
  #<Mate:0x8152664 @name="Club Mate">
  from (irb):38
  from :0
  ```
  ```

Getter & Setter

- ```
  class Drink
    attr_accessor :name
  end
  ```
  ```

Getter & Setter

- ```
  class Drink
    attr_accessor :name
  end
  ```
- ```
  m.name
  => "Club Mate"
  ```

Getter & Setter

- ```
  class Drink
    attr_accessor :name
  end
  ```
- ```
  m.name
  => "Club Mate"
  ```
- ```
  m.name = 'Jolt'
  => "Jolt"
  ```
  ```

Getter & Setter

- ```
  class Drink
    attr_accessor :name
  end
  ```
- ```
  m.name
  => "Club Mate"
  ```
- ```
  m.name = 'Jolt'
  => "Jolt"
  ```
- ```
  m.name
  => "Jolt"
  ```

Getter und Setter selbst bauen

```
class Drink
  def initialize(erster_name)
    self.name = erster_name
  end
  def name
    @name
  end
  def name=(neuer_name)
    if neuer_name == 'Beer'
      raise 'Please stay sober while coding'
    else
      @name = neuer_name
    end
  end
end
```

Getter und Setter selbst bauen

- `m = Drink.new('Mate')`
  `=> #<Drink:0x812cef0 @name="Mate">`
- `m.name = 'Milk'`
  `=> "Milk"`

Getter und Setter selbst bauen

- `m = Drink.new('Mate')`
  `=> #<Drink:0x812cef0 @name="Mate">`
- `m.name = 'Milk'`
  `=> "Milk"`
- `m.name = 'Beer'`
  `RuntimeError: Please stay sober while coding`
  `from (irb):7:in 'name='`
  `from (irb):14`
  `from :0`

Getter und Setter selbst bauen

- `m = Drink.new('Mate')`
  `=> #<Drink:0x812cef0 @name="Mate">`
- `m.name = 'Milk'`
  `=> "Milk"`
- `m.name = 'Beer'`
  `RuntimeError: Please stay sober while coding`
  `from (irb):7:in 'name='`
  `from (irb):14`
  `from :0`
- `b = Drink.new('Beer')`
  `RuntimeError: Please stay sober while coding`
  `from (irb):7:in 'name='`
  `from (irb):17:in 'initialize'`
  `from (irb):20`
  `from :0`

Klassen erweitern

- Dringend benötigte Funktionen
- Flicken von fehlerhaftem Fremdcode
- Verstreuen der Implementation auf mehrere Dateien

Klassen erweitern

- Dringend benötigte Funktionen
- Flicken von fehlerhaftem Fremdcode
- Verstreuen der Implementation auf mehrere Dateien

- Beispielproblem von turbo24prg: `String#to_b`

Klassen erweitern

- Dringend benötigte Funktionen
- Flicken von fehlerhaftem Fremdcode
- Verstreuen der Implementation auf mehrere Dateien

- Beispielproblem von turbo24prg: `String#to_b`
- ```
  class String
    def to_b
      self == 'true'
    end
  end
  ```
  ```

Klassen erweitern

- Dringend benötigte Funktionen
- Flicken von fehlerhaftem Fremdcode
- Verstreuen der Implementation auf mehrere Dateien

- Beispielproblem von turbo24prg: `String#to_b`
- ```
  class String
    def to_b
      self == 'true'
    end
  end
  ```
- ```
  'chunky bacon'.to_b
  => false
  ```
- ```
  'true'.to_b
  => true
  ```
  ```

Object#method_missing

```
class MeineKlasse
  def method_missing(*a)
    p a
  end
end
```

Object#method_missing

- ```
  class MeineKlasse
    def method_missing(*a)
      p a
    end
  end
  ```
- ```
  k = MeineKlasse.new
  => #<MeineKlasse:0x813db38>
  ```

Object#method_missing

- ```
  class MeineKlasse
    def method_missing(*a)
      p a
    end
  end
  ```
- ```
  k = MeineKlasse.new
  => #<MeineKlasse:0x813db38>
  ```
- ```
  k.hello
  [:hello]
  => nil
  ```
  ```

Object#method_missing

- ```
  class MeineKlasse
    def method_missing(*a)
      p a
    end
  end
  ```
- ```
  k = MeineKlasse.new
  => #<MeineKlasse:0x813db38>
  ```
- ```
  k.hello
  [:hello]
  => nil
  ```
- ```
  k.hello(23, 42)
  [:hello, 23, 42]
  => nil
  ```

"Builder"

```
class Builder
  def self.print
    puts yield(new)
  end
  def method_missing(fname, attrs={})
    "<#{fname}" + attrs.collect { |k,v| " #{k}=\"#{v}\""
}.to_s + (block_given?  ?  ">#{yield}</#{fname}>" :
'/>')
  end
end
```

"Builder" angewandt

```
Builder.print { |b|
  b.html {
    b.head {
      b.title {
        'Hello World!'
      }
    } +
    b.body {
      b.input 'name'=>'search'
    }
  }
}
```

"Builder" angewandt

- ```
  Builder.print { |b|
    b.html {
      b.head {
        b.title {
          'Hello World!'
        }
      } +
      b.body {
        b.input 'name'=>'search'
      }
    }
  }
  ```
- ```
  <html><head><title>Hello
  World!</title></head><body><input
  name="search"/></body></html>
  ```

Getter definieren mit Module#define_method

```
class MeineKlasse
  def initialize
    @variable = 'hallo'
  end
  define_method(:var) do
    @variable
  end
end
```

Getter definieren mit Module#define_method

- ```
  class MeineKlasse
    def initialize
      @variable = 'hallo'
    end
    define_method(:var) do
      @variable
    end
  end
  ```
- ```
  m = MeineKlasse.new
  => #<MeineKlasse:0x8060bf4 @variable="hallo">
  ```

Getter definieren mit Module#define_method

- ```
  class MeineKlasse
    def initialize
      @variable = 'hallo'
    end
    define_method(:var) do
      @variable
    end
  end
  ```
- ```
  m = MeineKlasse.new
  => #<MeineKlasse:0x8060bf4 @variable="hallo">
  ```
- ```
  m.var
  => "hallo"
  ```
  ```

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Module und Namensäume

```
module Jabber
  class Client
    def initialize
      # Code
    end
  end
end
```

Module und Namensäume

- ```
  module Jabber
    class Client
      def initialize
        # Code
      end
    end
  end
  ```
- ```
  client = Jabber::Client.new
  ```

Module und Mixins

```
module Eatable
  def eat
    puts "#{self.class} #{@name} has been eaten."
  end
end
```

Module und Mixins

- ```
  module Eatable
    def eat
      puts "#{self.class} #{@name} has been eaten."
    end
  end
  ```
- ```
  class Fryhstyck
    include Eatable
    def initialize
      @name = 'Egg and Ham'
    end
  end
  ```

Module und Mixins

- ```
  module Eatable
    def eat
      puts "#{self.class} #{@name} has been eaten."
    end
  end
  ```
- ```
  class Fryhstyck
    include Eatable
    def initialize
      @name = 'Egg and Ham'
    end
  end
  ```
- ```
  f = Fryhstyck.new
  => #<Fryhstyck:0x811e378 @name="Egg and Ham">
  ```
  ```

Module und Mixins

- ```
  module Eatable
    def eat
      puts "#{self.class} #{@name} has been eaten."
    end
  end
  ```
- ```
  class Fryhstyck
    include Eatable
    def initialize
      @name = 'Egg and Ham'
    end
  end
  ```
- ```
  f = Fryhstyck.new
  => #<Fryhstyck:0x811e378 @name="Egg and Ham">
  ```
- ```
  f.eat
  Fryhstyck Egg and Ham has been eaten.
  => nil
  ```

Modulmethoden

```
module MeinModul
  def MeinModul.homeless
    puts "I've got no instance"
  end
end
```

Modulmethoden

- ```
  module MeinModul
    def MeinModul.homeless
      puts "I've got no instance"
    end
  end
  ```
- ```
  MeinModul.homeless
  I've got no instance
  ```

Klassenmethoden

```
class MeineKlasse
  def self.neu
    new
  end
end
```

Klassenmethoden

- ```
  class MeineKlasse
    def self.neu
      new
    end
  end
  ```
- ```
  MeineKlasse.neu
  => #<MeineKlasse:0x810d3fc>
  ```

Klassenmethoden

- ```
  class MeineKlasse
    def self.neu
      new
    end
  end
  ```
- ```
  MeineKlasse.neu
  => #<MeineKlasse:0x810d3fc>
  ```
- ```
  class MeineKindklasse < MeineKlasse
  end
  ```
  ```

Klassenmethoden

- ```
  class MeineKlasse
    def self.neu
      new
    end
  end
  ```
- ```
  MeineKlasse.neu
  => #<MeineKlasse:0x810d3fc>
  ```
- ```
  class MeineKindklasse < MeineKlasse
  end
  ```
- ```
  MeineKindklasse.neu
  => #<MeineKindklasse:0x814e500>
  ```

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Exceptions

- ```
  begin
    f = File.new('/nonexistant/imaginary_file')
  rescue
    puts "Fehler:  #{$!}"
  end
  ```
  ```

Exceptions

- ```
  begin
    f = File.new('/nonexistant/imaginary_file')
  rescue
    puts "Fehler:  #{$!}"
  end
  ```
- ```
  Fehler:  No such file or directory -
  /nonexistant/imaginary_file
  ```

Exceptions: Begrenzen

- ```
  begin
    f = File.new('/nonexistant/imaginary_file')
  rescue Errno => e
    p e
  end
  ```
  ```

Exceptions: Begrenzen

- ```
  begin
    f = File.new('/nonexistant/imaginary_file')
  rescue Errno => e
    p e
  end
  ```
- ```
  Errno::ENOENT: No such file or directory -
  /nonexistant/imaginary_file
  from (irb):2:in `initialize'
  from (irb):2
  from :0
  ```

Exceptions: Syntaxfehler

- ```
  begin
    xxx
  rescue Exception => e
    puts "#{e.class}:  #{e}\n#{e.backtrace.join("\n")}"
  end
  ```
  ```

Exceptions: Syntaxfehler

- ```
  begin
    xxx
  rescue Exception => e
    puts "#{e.class}:  #{e}\n#{e.backtrace.join("\n")}"
  end
  ```
- ```
  NameError:  undefined local variable or method ‘xxx’
  for main:Object
  (irb):7:in ‘irb_binding’
  /usr/local/lib/ruby/1.8/irb/workspace.rb:52:in
  ‘irb_binding’
  :0
  ```

Exceptions: Mehr

- ```
  begin
    # Ausnahmenwerfender Code
  ensure
    # Was auf jeden Fall gemacht werden muss
  end
  ```
  ```

Exceptions: Mehr

- ```
  begin
    # Ausnahmenwerfender Code
  ensure
    # Was auf jeden Fall gemacht werden muss
  end
  ```
- `raise "Computer gone.  No processor to run on left."`
- `raise NotImplementedError.new('Programmer too lazy')`
  ```

Exceptions: Mehr

- ```
  begin
    # Ausnahmenwerfender Code
  ensure
    # Was auf jeden Fall gemacht werden muss
  end
  ```
- `raise "Computer gone.  No processor to run on left."`
- `raise NotImplementedError.new('Programmer too lazy')`
- Nochmal ausführen: `retry`
  ```

Upcoming...

2. Einführung
   - Meta
   - Hilfe?
   - irb & ri
   - Variablen
   - Alles ist ein Objekt
   - Konventionen
   - print-Debugging
   - Primitive Datentypen
   - Kontrollstrukturen
   - Alles hat ein Ergebnis
   - Bibliotheken laden
   - Methoden
   - Blöcke
   - Klassen
   - Module
   - Exceptions
   - Design Patterns

Factory method

- ```
  def factory_method
    ConcreteProduct.new
  end
  ```
  ```

Observer

- `require 'observer'`
- `module Observable` als Mixin

Singleton

- `require 'observer'`
- `module Singleton` als Mixin

Decorator

- Object#extend

Upcoming...

3. Ruby-MediaWiki
   - MediaWiki?
   - Ruby-MediaWiki!

MediaWiki?


Anmelden
Artikel
Diskussion
Quelltext betrachten
Versionen/Autoren
Hauptseite
WIKIPEDIA
Die freie Enzyklopädie
Navigation
Hauptseite
Über Wikipedia
Themenportale
Von A bis Z
Zufälliger Artikel
Mitmachen
Hilfe
Wikipedia-Portal
Letzte Änderungen
Spenden
Suche
Artikel
Suche
Werkzeuge
Links auf diese Seite
Änderungen an verlinkten Seiten
Hochladen
Spezialseiten
Druckversion
Permanentlink
Artikel zitieren
Willkommen in der Wikipedia!
Die Wikipedia ist ein Projekt zum Aufbau einer freien Enzyklopädie in mehr als 100 Sprachen. Jeder kann mit seinem Wissen beitragen und die Artikel direkt im Browser bearbeiten. Seit Mai 2001 entstanden so 367.135 Artikel in deutscher Sprache. Wir heißen gute Autoren willkommen – die ersten Schritte sind ganz einfach!
Weitere Informationen zur Wikipedia
Artikel nach Themen · Alphabetischer Index · Artikel nach Kategorien
Wikipedia aktuell
Am 1. März hat der vierte Schreibwettbewerb in der deutschsprachigen Wikipedia begonnen. Nominierungen sind über den gesamten Monat März möglich.
Am 11. März lädt der Zenodot-Verlag die Community zu Gesprächen über die geplante gedruckte Wikipedia ein. Infos und Anmeldung.
Artikel des Tages
m-mal
Als Alkane bezeichnet man in der organischen Chemie eine Stoffgruppe einfacher Kohlenwasserstoffe, bei der keine Mehrfachbindungen zwischen den Atomen auftreten. Sie bestehen, wie der
Aktuelles
Laut spanischem Roten Kreuz starben seit Jahresbeginn über 1000 Flüchtlinge auf dem Seeweg zu den Kanaren; offizielle Zahlen sprechen hingegen von 106 Opfern.
Das US-Repräsentantenhaus hat einer unbefristeten Verlängerung des Patriot Acts zugestimmt, dabei aber in einzelnen Punkten die Stärkung der Bürgerrechte durchgesetzt.
Im brasilianischen Bundesstaat Pará hat Greenpeace gegen eine massive Abholzung des Regenwaldes protestiert; seit drei Jahren gingen über sieben Mio. Hektar verloren.
Im indischen Varanasi (Benares; Bundesstaat Uttar Pradesh) sterben bei drei Bombenattentaten, darunter auch in einem hinduistischen Hanuman-Tempel, mindestens 20 Menschen.
Weitere Nachrichten bei Wikinews
Kürzlich Verstorbene
Teresa Ciepły (68), polnische Leichtathletin († 8. März)
Gordon Parks (93), US-amerikanischer Fotograf und Filmregisseur († 7. März)
Ali Farka Touré (67?), malischer Musiker († 7. März)
Dana Reeve (44), US-amerikanische Schauspielerin († 6. März)
Milan Babić (50), serbischer Kriegsverbrecher († 5. März)
Weitere Verstorbene

MediaWiki?


Log in / create account
article
discussion
view source
history
MediaWiki
navigation
Main Page
Recent changes
SourceForge project page
Extensions
Skins
development
#MediaWiki
Browse CVS
CVS statistics
API documentation
resources
Help
FAQ
Bug tracker
Mailing list
search
Go
Search
toolbox
What links here
Welcome to MediaWiki.org
MediaWiki is a free software package originally written for Wikipedia but is now run on other projects of the non-profit Wikimedia Foundation and many other wikis.
This site is the home of MediaWiki. If you want to explore its basic contents, use the navigation on the right side. You'll find the fundamental introduction translated into some other languages, but the reference language on the whole site is English. Please read more about this site.
If you have a question or a suggestion dealing with this wiki, use the related discussion page. For general questions dealing with the software see the recommended possibilities of communication.
+ + + News + + +
Problems after upgrading to PHP 4.4.1? – Don't forget to upgrade MediaWiki to 1.5.7 / 1.4.14 / 1.3.18!
2006-03-02: 1.5.7 fixes MySQL password problem in installer
Various bug fixes and improved compatibility with IE7 beta 2
2006-01-19: 1.5.6 and 1.4.14 fix infinite loop bug in malformed edit comments
Recommended upgrade; also various bug fixes in 1.5.6.
2006-01-05: Protection against Windows WMF vulnerability in 1.5.5 and 1.4.13
Please upgrade if you have uploads enabled to avoid being a vector for unpatched clients.
2005-12-21: Fix for remote code execution bug in 1.5 branch
Upgrade all 1.5 installations to 1.5.4 or higher immediately!
Current Versions
1.5.7 · 2006-03-02
1.4.14 · 2006-01-19
1.3.18 · 2005-11-02
Fundamental Introduction to MediaWiki
Contents
How does MediaWiki work?
Documentation
Customization
Versions & Download
Installation
Support & Contact
Development
All other topics
See navigation on the left
Reference: English – Other: Deutsch · Español · Français · Italiano · 日本語 · 한국어 · Português · 中文 ·

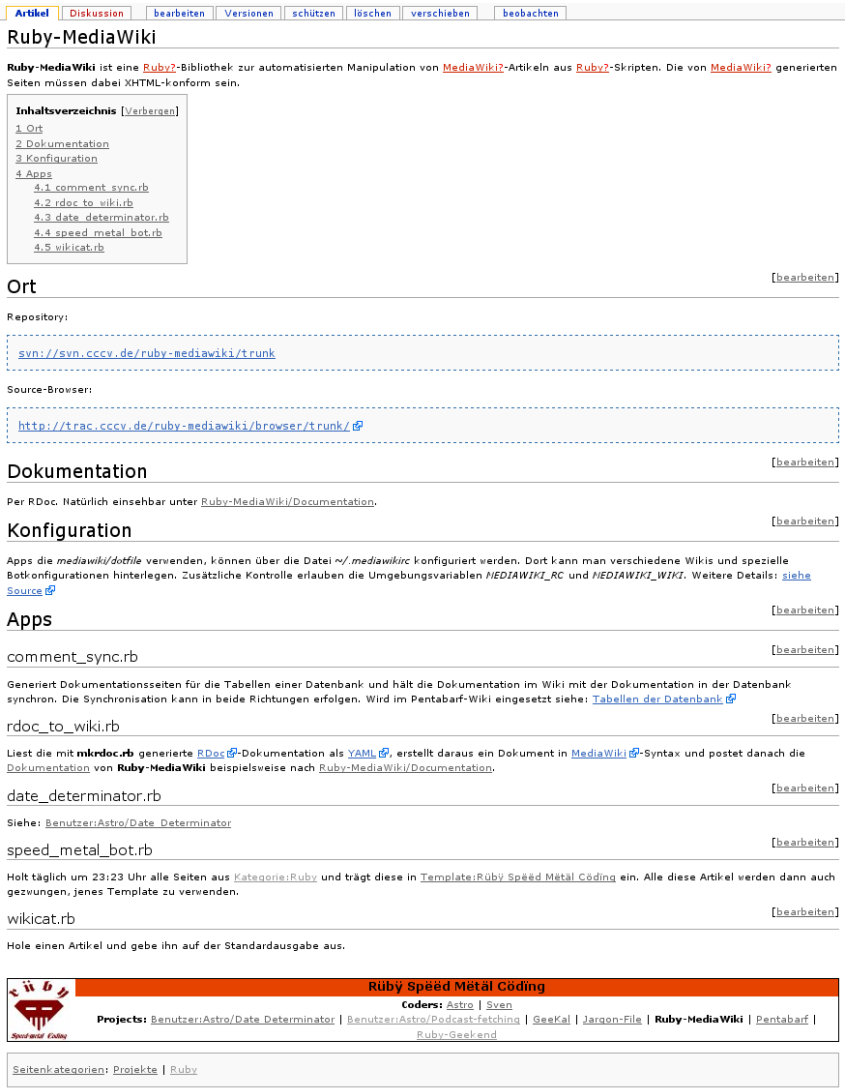
MediaWiki?


Artikel
Diskussion
bearbeiten
Versionen
schützen
löschen
verschieben
beobachten
Hauptseite
Navigation
Hauptseite
C3D2 Webseite
C3D2-Webseite ++cool
Letzte Änderungen
Zufälliger Artikel
Hilfe
Suche
Los
Suche
Werkzeuge
Was zeigt hierhin
Verlinkte Seiten
Hochladen
Spezialseiten
Druckversion
Permanentlink
Willkommen im Wiki des CCC Dresden (aka C3D2)
[bearbeiten]
Hinweise zum Selbstverständnis gibt es auf der Seite über C3D2. Im Moment umfasst dieses Wiki 87 Seiten.
Bitte lest zuerst die Einführung, damit ihr wisst, wie hier am Besten mitgewirkt werden kann. Zum Rumspielen kann die Spielwiese benutzt werden.
Einen Überblick darüber, was sich in diesem Wiki geändert hat, findet man stets in den Letzten Änderungen. Wichtige Änderungen im System finden sich im Logbuch. Neugierige können auch mal einen Blick auf die Seitenstatistik werfen.
Da auch das Chaos organisiert werden muss, gibt es mittlerweile verschiedene Kategorien. Für den allumfassenden Überblick gibt es eine Auflistung aller Artikel und aller Benutzerseiten.
Termine
[bearbeiten]
2006 - Alles, was 2006 kommt.
Themenabende:
10. März, 19oo Uhr: TA Ruby in Vorbereitung des Ruby-Geekends
es gibt weitere Ideen für Themenabende
Protokolle: Plenum 2006-01-24, Netzbiotop?, Realitätsabgleich 2?, Realitätsabgleich
Archiv: 2005
Projekte
[bearbeiten]
ChaosVPN
Pentabarf
C3D2-Web Public Beta
Ruby Speed Metal Coding
Jargon-File
Archiv: Vortragsreihe an Schulen, C3D2Bibliothek, cccddcd, ddca, Open Root Server Network, Hackschiff?, GeeKal, Googlebombing, Deathmatch, Link
Infrastruktur
[bearbeiten]
SILC
Office2/Netz
Teams:
Web-Team
Mailingliste
Redaktion
DNS-Team
Archiv: Netzbiotop, Satzung, Kassenordnung, Beitragsordnung, Räume, Office2, Office2/Netzanbindung
Wissen
[bearbeiten]
Allgemein: Mate, Presse Arbyte
C3D2: Presse, DS-Selbstdarstellung, Blitz Artikel, Erfa Dresden,

Upcoming...

3. Ruby-MediaWiki
   - MediaWiki?
   - Ruby-MediaWiki!

Ruby-MediaWiki!



Artikel | Diskussion | bearbeiten | Versionen | schützen | löschen | verschieben | beobachten

Ruby-MediaWiki

Ruby-MediaWiki ist eine Ruby?-Bibliothek zur automatisierten Manipulation von MediaWiki?-Artikeln aus Ruby?-Skripten. Die von MediaWiki? generierten Seiten müssen dabei XHTML-konform sein.

Inhaltsverzeichnis [Verbergen]

1 Ort
2 Dokumentation
3 Konfiguration
4 Apps
- 4.1 comment_sync.rb
- 4.2 rdoc_to_wiki.rb
- 4.3 date_determinator.rb
- 4.4 speed_metal_bot.rb
- 4.5 wikicat.rb

Ort [bearbeiten]

Repository:

svn://svn.cccv.de/ruby-mediawiki/trunk

Source-Browser:

http://trac.cccv.de/ruby-mediawiki/browser/trunk/

Dokumentation [bearbeiten]

Per RDoc. Natürlich einsehbar unter Ruby-MediaWiki/Documentation.

Konfiguration [bearbeiten]

Apps die *mediawiki/dotfile* verwenden, können über die Datei *~/.mediawikirc* konfiguriert werden. Dort kann man verschiedene Wikis und spezielle Botkonfigurationen hinterlegen. Zusätzliche Kontrolle erlauben die Umgebungsvariablen *MEDIAWIKI_RC* und *MEDIAWIKI_WIKI*. Weitere Details: siehe Source

Apps [bearbeiten]

comment_sync.rb [bearbeiten]

Generiert Dokumentationsseiten für die Tabellen einer Datenbank und hält die Dokumentation im Wiki mit der Dokumentation in der Datenbank synchron. Die Synchronisation kann in beide Richtungen erfolgen. Wird im Pentabarf-Wiki eingesetzt siehe: Tabellen der Datenbank

rdoc_to_wiki.rb [bearbeiten]

Liest die mit **mkrdoc.rb** generierte RDoc-Dokumentation als YAML, erstellt daraus ein Dokument in MediaWiki-Syntax und postet danach die Dokumentation von **Ruby-MediaWiki** beispielsweise nach Ruby-MediaWiki/Documentation.

date_determinator.rb [bearbeiten]

Siehe: Benutzer:Astro/Date_Determinator

speed_metal_bot.rb [bearbeiten]

Holt täglich um 23:23 Uhr alle Seiten aus Kategorie:Ruby und trägt diese in Template:Rübÿ Spëëd Mëtäl Cödîng ein. Alle diese Artikel werden dann auch gezwungen, jenes Template zu verwenden.

wikicat.rb [bearbeiten]

Hole einen Artikel und gebe ihn auf der Standardausgabe aus.

Rübÿ Spëëd Mëtäl Cödîng
Coders: Astro | Sven
Projects: Benutzer:Astro/Date Determinator | Benutzer:Astro/Podcast-fetching | GeeKal | Jargon-File | **Ruby-MediaWiki** | Pentabarf | Ruby-Geekend

Seitenkategorien: Projekte | Ruby

Ruby-MediaWiki!

Ruby-MediaWiki Documentation [bearbeiten]

MediaWiki [bearbeiten]

MediaWiki::dotfile(myrealm=nil) [bearbeiten]

dotfile function reads the user's MediaWiki config and creates a Wiki instance.

The filename is determined by the environment variable MEDIAWIKI_RC or defaults to ~/.mediawikirc .

A configured wiki can be chosen with the MEDIAWIKI_WIKI environment variable, or defaults to the wiki pointed by default.

A robot may set [myrealm] to retrieve a second result output: a section with this name in the current wiki's configuration file for configuration of specific robot tasks.

MediaWiki::Article [bearbeiten]

Inherited from **Object**

The Article class represents MediaWiki articles.

MediaWiki::Article#name (RW) [bearbeiten]

Article name, will be refreshed upon Article#reload

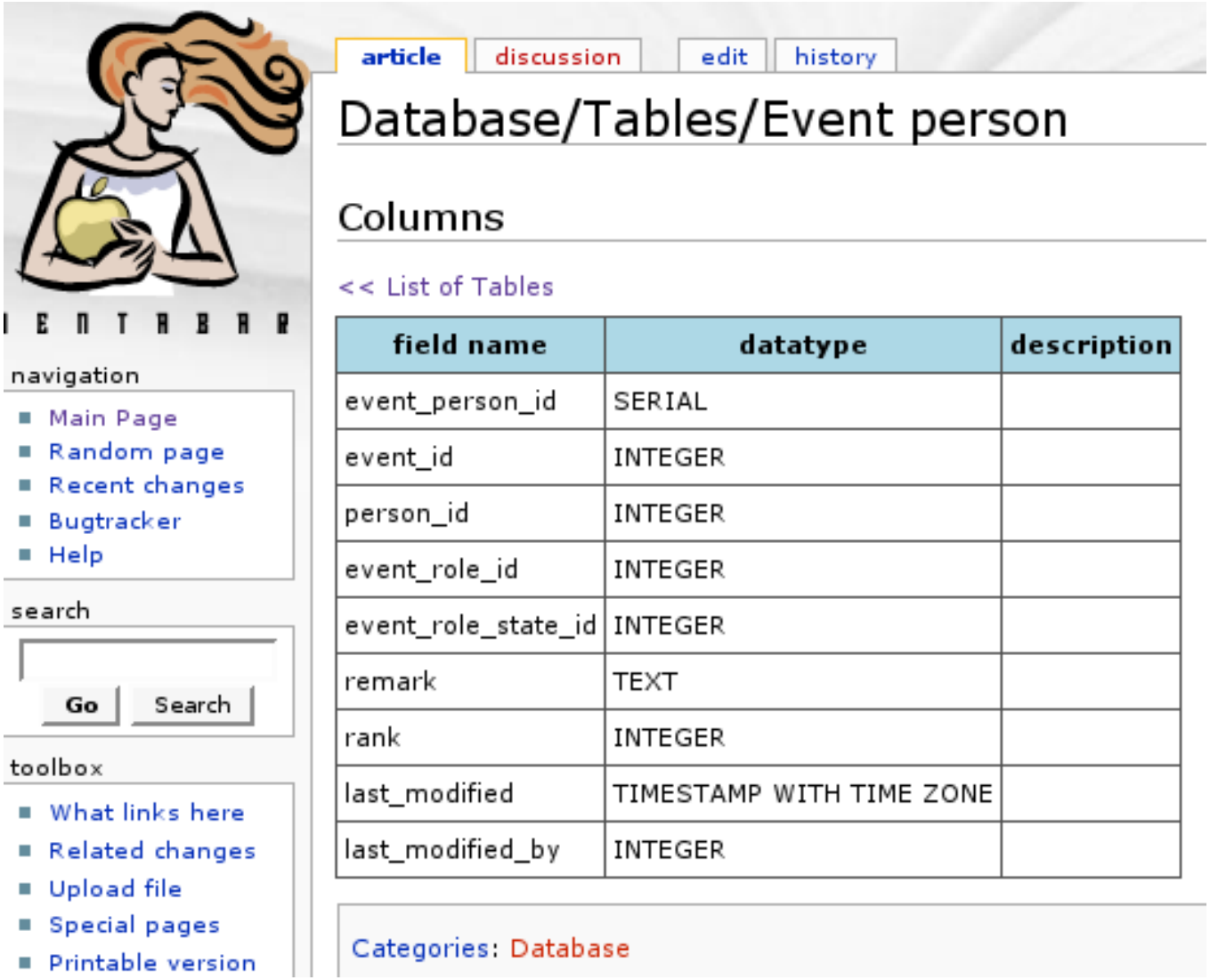
MediaWiki::Article#text (RW) [bearbeiten]

Article text, will be set by Article#reload

MediaWiki::Article::new(wiki, name, section = nil, load_text=true) [bearbeiten]

Create a new Article instance

Ruby-MediaWiki!



field name	datatype	description
event_person_id	SERIAL	
event_id	INTEGER	
person_id	INTEGER	
event_role_id	INTEGER	
event_role_state_id	INTEGER	
remark	TEXT	
rank	INTEGER	
last_modified	TIMESTAMP WITH TIME ZONE	
last_modified_by	INTEGER	