



PM1/PT Ruby: Einbinden externer Dateien

von Bibliotheksklassen und eigenen
Klassen und eigenen Scripten



require lädt Ruby-Programmtext in eigene Programme

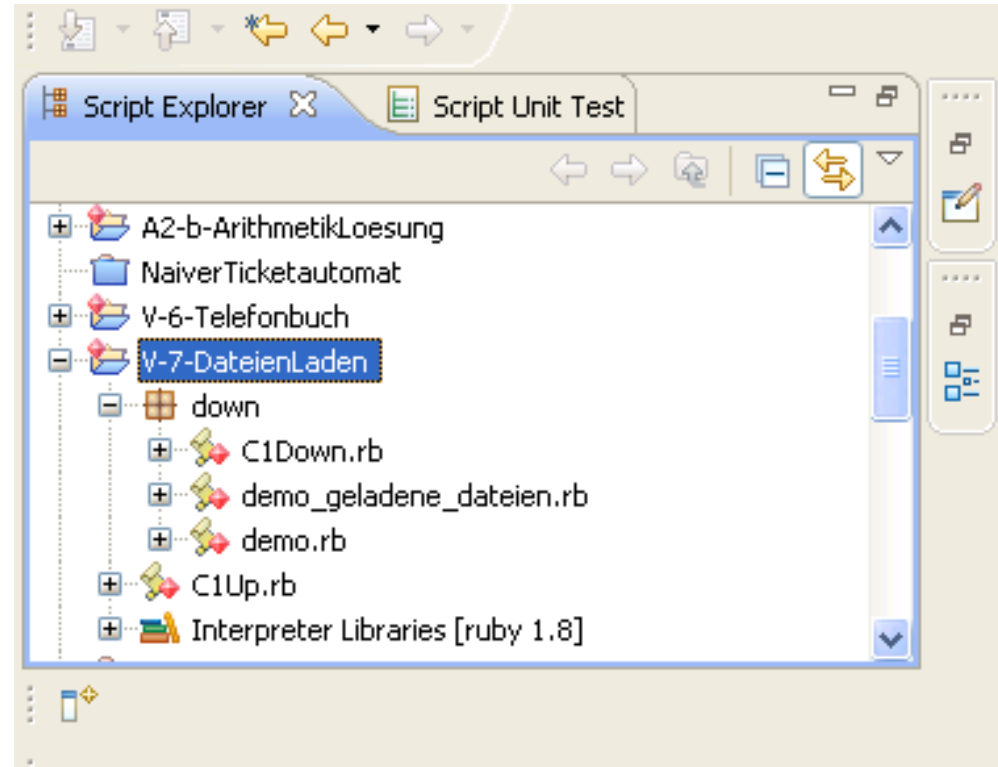
- Werden in einer Klasse externe Bibliotheksklassen verwendet, dann müssen diese, solange sie nicht zum Ruby Standardbibliothek gehören, zuerst geladen werden.
- Gleiches gilt auch für Klassen und Skripte, die Sie selber geschrieben haben.
- *require <Name des Programms ohne Dateiendung>* lädt eine Datei und übersetzt diese zum Zeitpunkt des Ladens.
- *require* lädt ein Programm nur genau einmal. Ruby merkt sich, welche Programme bereits geladen wurden.
- Die geladenen Programme können in der Variable *\$"* nachgeschaut werden.
- Der Name der Datei kann absolut oder relativ adressiert werden.
- **Absolut:** Es wird immer der vollständige Dateiname unter Angabe des Laufwerks angegeben.
- **Relativ:** Der Dateiname wird relativ zu den Verzeichnissen angegeben, die im Ruby **load path** eingetragen sind.
- **Eclipse** trägt das **Projektverzeichnis** in den Ruby load path ein.



Beispiel: Laden eigener Dateien

- Gegeben die rechte Projektstruktur in Eclipse.
- Dann müssen in *demo.rb* die Dateien *C1Up.rb* und *C1Down.rb* wie folgt geladen werden:

```
require "C1Up"  
require "down/C1Down"  
# "../C1Up" wird nicht gefunden.  
# "down" ist nicht im load path  
require "../C1Up"
```






Beispiel: Ausgabe des „load path“ in Ruby

- Der Ruby load path wird in globalen Variablen gespeichert:
 - `$:`
 - `$LOAD_PATH`

```
puts "Load_Path inspizieren"
puts "$:"
puts $:
puts "$LOAD_PATH"
puts $LOAD_PATH

Load_Path inspizieren
$:
C:/Users/birgit/DATEN/HAW/vorlesungen/rubypr1/sose10
/workspace/v7-a DateienLaden
C:/Ruby187/lib/ruby/site_ruby/1.8
C:/Ruby187/lib/ruby/site_ruby/1.8/i386-msvcrt
C:/Ruby187/lib/ruby/site_ruby
C:/Ruby187/lib/ruby/vendor_ruby/1.8
C:/Ruby187/lib/ruby/vendor_ruby/1.8/i386-msvcrt
C:/Ruby187/lib/ruby/vendor_ruby
C:/Ruby187/lib/ruby/1.8
C:/Ruby187/lib/ruby/1.8/i386-mingw32
.
$LOAD_PATH
C:/Users/birgit/DATEN/HAW/vorlesungen/rubypr1/sose10
/workspace/v7-a DateienLaden
C:/Ruby187/lib/ruby/site_ruby/1.8
C:/Ruby187/lib/ruby/site_ruby/1.8/i386-msvcrt
C:/Ruby187/lib/ruby/site_ruby
C:/Ruby187/lib/ruby/vendor_ruby/1.8
C:/Ruby187/lib/ruby/vendor_ruby/1.8/i386-msvcrt
C:/Ruby187/lib/ruby/vendor_ruby
```





Beispiel: Eindeutigkeit von Namen

- Das Script `demo_geladene_dateien.rb`, gibt nach jedem `require` die neu geladene Datei aus.
- Der Pfad `"./C1Up"` steht für die gleiche Datei wie der Pfad `"C1Up"`. Dennoch wird die Datei `C1Up` zweimal geladen.
- Grund: `require` schreibt die Argumente als String in `$"`. Da der String `"C1Up"` ungleich `"./C1Up"` ist, wird hier die Datei `C1Up` zweimal geladen

```
bereits_geladen = Array.new("$")
require "C1Up"
p("$" -bereits_geladen)
```

```
bereits_geladen = Array.new("$")
require "down/C1Down"
p("$" -bereits_geladen)
```

```
bereits_geladen = Array.new("$")
# require schreibt seine Argumente
#   in "$", ohne
#   den Pfadnamen aufzulösen.
# C1Up wird mit nachfolgendem
#   require erneut geladen, da
#   der String ./C1Up nicht in "$"
#   enthalten ist.
require "./C1Up"
p("$" -bereits_geladen)
```



Ausgabe des letzten Beispiels

```
bereits_geladen = Array.new($")  
require "C1Up"  
p($" -bereits_geladen)
```

```
bereits_geladen = Array.new($")  
require "down/C1Down"  
p($" -bereits_geladen)
```

```
bereits_geladen = Array.new($")  
# require schreibt seine Argumente  
in $\", ohne  
# den Pfadnamen aufzuloesen.  
# C1Up wird mit nachfolgendem  
require erneut geladen, da  
# der String ./C1Up nicht in $\"  
enthalten ist.  
require \"/C1Up"  
p($" -bereits_geladen)
```

```
up loaded  
["C1Up.rb"]  
down/C1Down loaded  
["down/C1Down.rb"]  
up loaded  
["./C1Up.rb"]
```