# Lottozahlengenerator mit Ruby & Shoes

== About Shoes ==
Shoes ist ein Cross-Platform-Toolkit um grafische Oberflächen (GUIs) zu erstellen.
Das schöne an Shoes ist, das es sehr einfach zu erlernen, gut dokumentiert ist und sich sowohl unter Win als auch unter Unix perfekt an das aktuelle Fenstermanagement anpasst.

Hier ein einfaches Beispielfenster mit einem Button und einem event.

**Quellcode**

```
1. Shoes.app do
2. button("easy-coding") { alert("FTW!") }
3. end
```

Hier ist das manual zu finden.

== Ein Programm starten ==
Man benötigt keine Libs oder oder Zusatzgems die man einbinden muss um das Programm auszuführen. Zuerst muss man die ShoesApp herunterladen.
Wenn man die App startet sollte man ein solches Fenster zu sehen bekommen:
[Blockierte Grafik: http://i.imgur.com/shw28.png]

Man wählt einfach "Open an App" und wählt die *.rb Datei mit der Shoes.app Methode aus ... und schon sieht man das Ergebnis.

== Motivation ==
Unser Projekt soll am Ende so aussehen :
[Blockierte Grafik: http://i.imgur.com/JovBQ.png]

Mit einem Klick auf den Button werden neue Zahlen generiert .... nicht mehr, nicht weniger. (Natürlich dürft Ihr das Programm erweitern )

Um dieses Ergebnis zu erhalten, werden wir grade einmal 40 Zeilen Code benötigen !

== Der Lottozahlengenerator ==
Als erstes werden wir die GUI erstellen und uns danach um die Programmlogik kümmern.

=== GUI ===
Der Code für das leere Fenster mit Größenbegrenzung sowie Titel sollte selbsterklärend sein.

**Quellcode**

```
1. # GUI
2. Shoes.app :title => "Lottogenerator", :height => 120, :width => 250 do
3. end
```

Die draw-Methode zum zeichnen des Inhaltes kommt in unsere Lottoklasse.

## Quellcode

```
1. # Lottoklasse
2. class Lottogenerator
3. # Methode zum zeichnen des Inhaltes
4. def draw
5. end
6. end
```

Damit wir auch innerhalb unserer Lottoklasse auf das Shoesobjekt zugreifen können müssen wir die Instanz übergeben. Wir übergeben einfach im Konstruktor die Instanz, welche wir in der Klasse in der Instanzvariable @shoes speichern.

## Quellcode

```
1. # Lottoklasse
2. class Lottogenerator
3. def initialize shoes
4. @shoes = shoes
5. draw
6. end
7. def draw
8. end
9. end
10. # GUI
11. Shoes.app :title => "Lottogenerator", :height => 120, :width => 250 do
12. Lottogenerator.new(self)
13. end
```

Alles anzeigen

Die fertige draw-Methode sieht am Ende so aus:

## Quellcode

```
1. def draw
2. # Überschrift
3. @shoes.para "Lottozahlengenerator"
4. # Zahlen generieren (haben wir noch nicht implementiert)
5. lr = generate_lotto_numbers
6. # Ausgabe
7. @shoes.flow do
8. 6.times { |t| @shoes.para lr[t].to_s }
9. @shoes.para "Zusatzzahl: " + lr[6].to_s
10. end
11. # Button
12. @shoes.button "Lottozahlen generieren" do
13. @shoes.clear
14. draw
15. end
16. end
```

Alles anzeigen

Die para-Methode (Paragraph) gibt einen String aus.
Die Methode generate_lotto_numbers werden wir nachfolgend implementieren. Diese Methode gibt ein eindimensionales Array zurück.

In Shoes hat man 2 Methode um die Ausgabe in und mit "Slots" (Container) zu formatieren. Die eine heisst flow, die andere stack.
Wie der Name schon verrät ist flow (fließen) für die horizontale Ausrichtung und stack (Stapel) für die Vertikale Ausrichtung zuständig.

Es ist keine Pflicht Slots zu benutzen, man kann die Elemente natürlich auch direkt in das Fenster setzen.
Mehr über Slots gibts es hier!

Unsere Zahlen sollen von links nach rechts "fließen".
[Blockierte Grafik: http://i.imgur.com/p2SP6.png]

Wir nutzen hier einfach flow als Block und geben das Array aus.
Hier wird nicht die each-Methode genutzt, da das letzte Element im Array die Zusatzzahl ist, welche mit dem Vermerk "Zusatzzahl" ausgegeben werden soll, daher der Umweg über die times-Methode.

### Quellcode

```
1. @shoes.flow do
2. 6.times { |t| @shoes.para lr[t].to_s }
3. @shoes.para "Zusatzzahl: " + lr[6].to_s
4. end
```

(Wer noch nicht so viel oder garnicht mit Ruby gearbeitet hat, zur Info, alles ist ein Objekt, es gibt keine primitiven Datentypen. Somit ist auch die Zahl 6 ein Objekt auf die wir ganz einfach die Methode times anwenden können.)

Die Methode button gibt einfach einen Button mit dem angegebenen String aus. Der folgende Block beschreibt die Action des onclick-Events. Hier nutzen wir die clear-Methode um den Inhalt des Fensters zu löschen und rufen die draw Methode auf um alles , mit neuen Zahlen, neu zu zeichen.

### Quellcode

```
1. # Button
2. @shoes.button "Lottozahlen generieren" do
3. @shoes.clear
4. draw
5. end
```

Das ganze kann man auch anders schreiben.

### Quellcode

```
1. btn = @shoes.button "Lottozahlen generieren"
2. btn.click {
3. @shoes.clear
4. draw
5. }
```

Alle weiteren Elemente und Elementeigenschaften von Shoes findet man hier!

Wer die GUI schöner und bunter machen möchte guckt einfach im manual bei den styles und/oder bei den unterstützten colors.

=== Programmlogik ===
Zu guter letzt erstellen wir nur noch die generate_lotto_numbers-Methode, welche das Array mit den 6 Lottozahlen und der Zusatzzahl anlegt.

### Quellcode

```
1. def generate_lotto_numbers
```

```
2.  lotto = Array.new(6)
3.  # 6 aus 49
4.  lotto.map! {|l|
5.  # Stichwort Mathematik, Stochastik > ungeordnet, ohne zurücklegen
6.  begin
7.  randLottoInt = ((rand * 49) + 1).to_i
8.  end while lotto.include?(randLottoInt)
9.  l = randLottoInt
10. }
11. # Zusatzzahl
12. lotto.push(((rand * 9) + 1).to_i)
13. end
```

Alles anzeigen

Dazu speichern wir ein Array in der Variable Lotto und gehen mittels der map!-Methode, das Array durch und erstellen die Zufallszahlen.
Da auf einem Lottoschein jede Zahl nur einmal vorkommen darf, existiert innerhalb des map!-Blockes ein weiterer Block mit einer While-Schleife der solange eine Zufallszahl zwischen 1 und 49 sucht, wie die include?-Methode true zurückliefert, also die Zahl schon vorhanden ist.
Wenn dies nicht der Fall ist, dann wird die Zahl gespeichert.

Am Ende wird eine Zusatzzahl zwischen 1 und 9 generiert und mittels push an das Ende des Arrays angehängt.

## Thats it !

Man hat auch die Möglichkeit mit Shoes, das Ergebnis in eine ausführbare Shoes,OSX,Windows oder Linux Datei zu "packen". Darauf wird hier nicht weiter eingegangen.

== Das fertige Programm ==

### Quellcode

```
1.  # Lottoklasse
2.  class Lottogenerator
3.  def initialize shoes
4.  @shoes = shoes
5.  draw
6.  end
7.  def generate_lotto_numbers
8.  lotto = Array.new(6)
9.  # 6 aus 49
10. lotto.map! {|l|
11. # Stichwort Mathematik, Stochastik > ungeordnet, ohne zurücklegen
12. begin
13. randLottoInt = ((rand * 49) + 1).to_i
14. end while lotto.include?(randLottoInt)
15. l = randLottoInt
16. }
17. # Zusatzzahl
18. lotto.push(((rand * 9) + 1).to_i)
```

```
19.  end
20.  def draw
21.  # Überschrift
22.  @shoes.para "Lottozahlengenerator"
23.  # Zahlen generieren
24.  lr = generate_lotto_numbers
25.  # Ausgabe
26.  @shoes.flow do
27.  6.times { |t| @shoes.para lr[t].to_s }
28.  @shoes.para "Zusatzzahl: " + lr[6].to_s
29.  end
30.  # Button
31.  @shoes.button "Lottozahlen generieren" do
32.  @shoes.clear
33.  draw
34.  end
35.  end
36.  end
37.  # GUI
38.  Shoes.app :title => "Lottogenerator", :height => 120, :width => 250 do
39.  Lottogenerator.new(self)
40.  end
```

Alles anzeigen