

Benoit Prioux

benoit.prioux@gmail.com

Asciidoctor et Java

Me

 @binout

 <https://github.com/binout>



- Développeur Java depuis presque 10 ans
- Technical Leader chez Lectra, numéro un mondial des solutions dédiées à l'industrie du textile (machines et logiciels)
- Membre du Bordeaux JUG
 - Conférence bdx.io le 17 octobre !



Vous n'aimez pas écrire de la doc ?



<http://www.npr.org/blogs/13.7/2013/12/10/249726951/the-infinite-monkey-theorem-comes-to-life>

Une solution

Utiliser un langage de **balisage** : markdown, wiki, LaTeX, asciidoc, ...

Une solution

Utiliser un langage de **balisage** : markdown, wiki, LaTeX, asciidoc, ...

parce que :

Une solution

Utiliser un langage de **balisage** : markdown, wiki, LaTeX, asciidoc, ...

parce que :

- on se concentre plus sur le fond que sur la forme

Une solution

Utiliser un langage de **balisage** : markdown, wiki, LaTeX, asciidoc, ...

parce que :

- on se concentre plus sur le fond que sur la forme
- c'est du texte, donc un éditeur classique suffit

Une solution

Utiliser un langage de **balisage** : markdown, wiki, LaTeX, asciidoc, ...

parce que :

- on se concentre plus sur le fond que sur la forme
- c'est du texte, donc un éditeur classique suffit
- on peut gérer l'historique avec un SCM

Une solution

Utiliser un langage de **balisage** : markdown, wiki, LaTeX, asciidoc, ...

parce que :

- on se concentre plus sur le fond que sur la forme
- c'est du texte, donc un éditeur classique suffit
- on peut gérer l'historique avec un SCM
- à partir d'une même source, on peut publier vers plusieurs formats

Une solution

Utiliser un langage de **balisage** : markdown, wiki, LaTeX, asciidoc, ...

parce que :

- on se concentre plus sur le fond que sur la forme
- c'est du texte, donc un éditeur classique suffit
- on peut gérer l'historique avec un SCM
- à partir d'une même source, on peut publier vers plusieurs formats
- on a un peu l'impression de hacker ;-)

Asciidoc

- Langage créé en 2002
- Syntaxe proche de markdown mais plus évoluée
- Plus polyvalent qu'un wiki
- Plus simple que LaTeX
- 1^{ère} implémentation du processeur en Python



Use AsciiDoc for document markup. Really. It's actually readable by humans, easier to parse and way more flexible than XML.

Linus Torvalds

Exemple

```
= Hello, Jug Summer Camp!  
Benoit Prioux <benoit.prioux@gmail.com>
```

Introduction à <http://asciidoc.org>[AsciiDoc].

```
== Première Section
```

- * foo
- * bar

```
== Deuxième Section
```

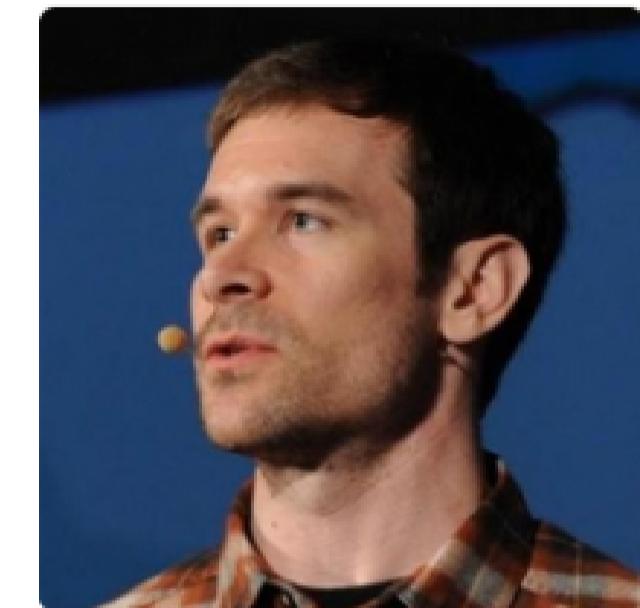
1. item 1
2. item 2

NOTE: C'est l'heure de la démo !

Asciidoctor

<https://github.com/asciidoctor>

- Implémentation open-source écrit en Ruby
- Permet de convertir des fichiers asciidoc vers différents backends : docbook, html5, epub, pdf (et même deckjs)
- 12 août 2014 : sortie de la version 1.5.0
 - 1^{ère} version majeure après 2 ans de développement
 - 50 contributeurs, 1800 commits, 1500 tests
 - introduit des évolutions de la syntaxe asciidoc, tout en gardant la compatibilité



Dan Allen
mojavelinux

Ils utilisent déjà asciidoctor



O'REILLY®



**Et si je ne connais pas Ruby, c'est
pas pour moi ?**

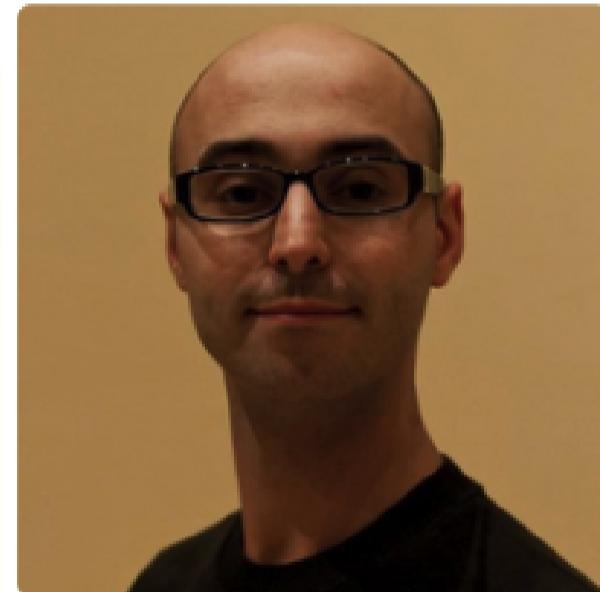
**Et si je ne connais pas Ruby, c'est
pas pour moi ?**



Et si je ne connais pas Ruby, c'est pas pour moi ?



+



Alex Soto
lordofthejars

= AsciidoctorJ

Asciidoctorj en action !

Asciidoctorj en action !

```
Asciidoctor asciidoctor = Asciidoctor.Factory.create();  
  
Options options = options().backend("html5").get();  
String rendered = asciidoctor.convert("*Gras* ou  
_italique_?", options);  
  
System.out.println(rendered);
```

Asciidoctorj en action !

```
Asciidoctor asciidoctor = Asciidoctor.Factory.create();  
  
Options options = options().backend("html5").get();  
String rendered = asciidoctor.convert("*Gras* ou  
_italique_?", options);  
  
System.out.println(rendered);
```

Console

```
<div class="paragraph">  
<p><strong>Gras</strong> ou <em>italique</em> ?</p>  
</div>
```

Asciidoctor et Maven

pom.xml

```
<plugin>
  <groupId>org.asciidoctor</groupId>
  <artifactId>asciidoctor-maven-plugin</artifactId>
  <version>1.5.0</version>
  <executions>
    <execution>
      <id>output-html</id>
      <phase>generate-resources</phase>
      <goals>
        <goal>process-asciidoc</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Asciidoctor et Gradle

build.gradle

```
buildscript {  
    repositories {  
        jcenter()  
    }  
  
    dependencies {  
        classpath 'org.asciidoctor:asciidoctor-gradle-  
plugin:1.5.0'  
    }  
}  
  
apply plugin: 'org.asciidoctor.gradle.asciidoctor'
```

Asciidoctor et Ant (unofficial)

<https://github.com/binout/asciidoctor-ant>

```
<target name="doc">
    <!-- on force le chargement de Asciidoctor par Ant --
>
    <taskdef
resource="net/jtools/classloadertask/antlib.xml"
classpath="lib/ant-classloadertask.jar"/>
    <classloader loader="thread"
classpath="lib/asciidoctor-ant.jar"/>

    <taskdef name="asciidoctor"
classname="org.asciidoctor.ant.AsciidoctorAntTask"/>
    <asciidoctor sourceDirectory="src/asciidoc"
outputDirectory="build/docs"/>
</target>
```

Asciidoctor est extensible ...



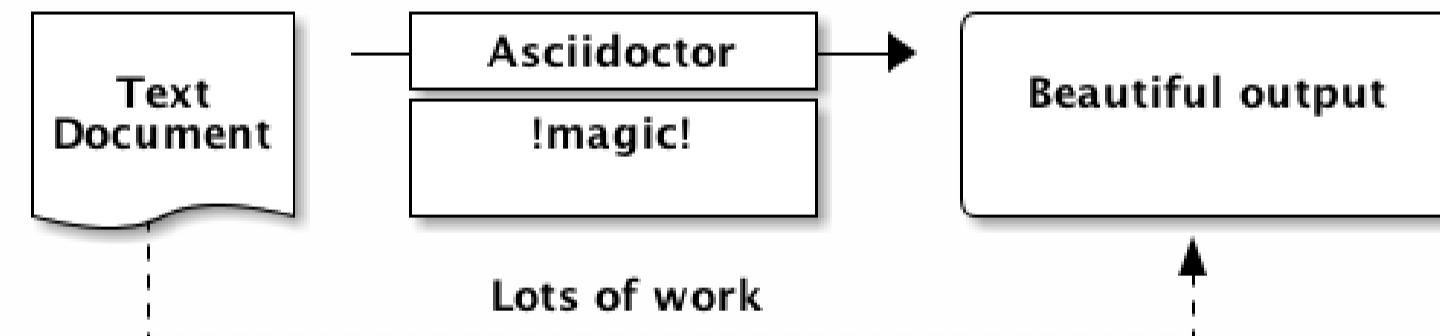
asciidioctor-diagram

asciidoct-diagram



asciidoc-diagram

```
[ditaa]
-----
+-----+ +-----+-----+ /-----\
|     | --- Asciidoc ---> | | |
| Text | +-----+ |Beautiful output|
| Document| | !magic! | |
| {d}| | | |
+-----+ +-----+ \-----/
:
| Lots of work |
+-----+
-----
```



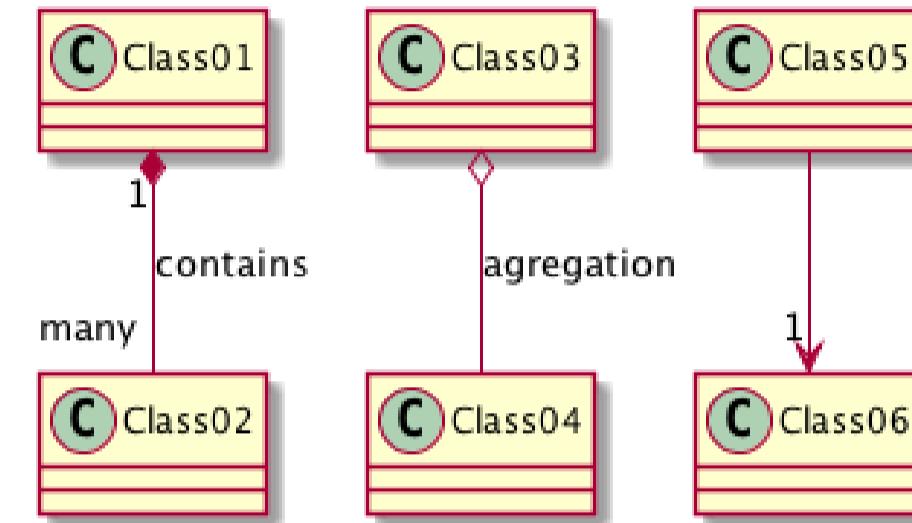
asciidioctor-diagram et UML

asciidocor-diagram et UML

```
[plantuml]
-----
Class01 "1" *-- "many" Class02 : contains
Class03 o-- Class04 : aggregation
Class05 --> "1" Class06
-----
```

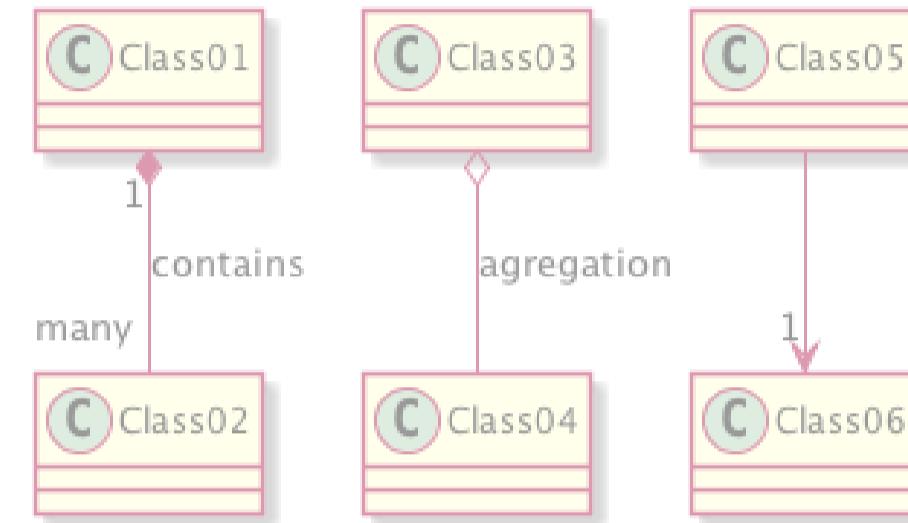
asciidioctor-diagram et UML

```
[plantuml]
-----
Class01 "1" *-- "many" Class02 : contains
Class03 o-- Class04 : aggregation
Class05 --> "1" Class06
-----
```



asciidocdoctor-diagram et UML

```
[plantuml]
-----
Class01 "1" *-- "many" Class02 : contains
Class03 o-- Class04 : aggregation
Class05 --> "1" Class06
-----
```

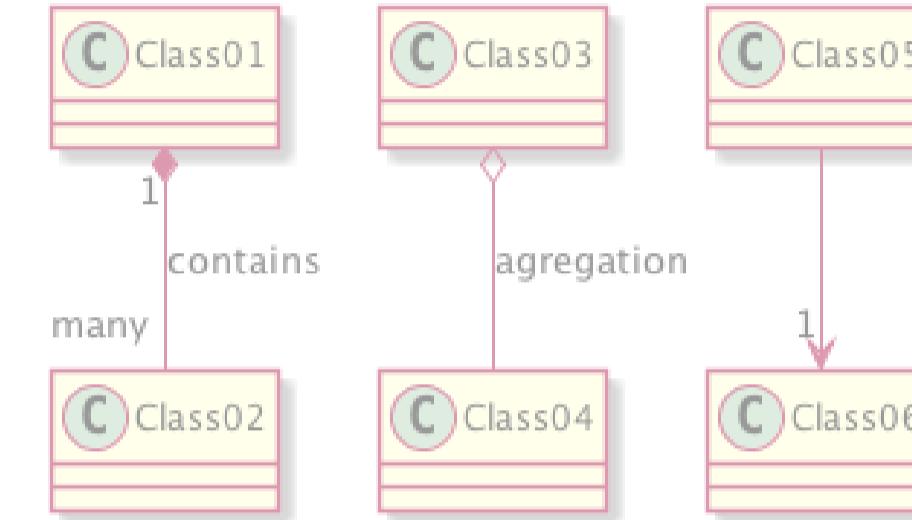


```
[plantuml]
-----
Client -> Server: Authentication Request
Server --> Client: Authentication Response

Client -> Server: Another authentication Request
Client <-- Server: another authentication Response
-----
```

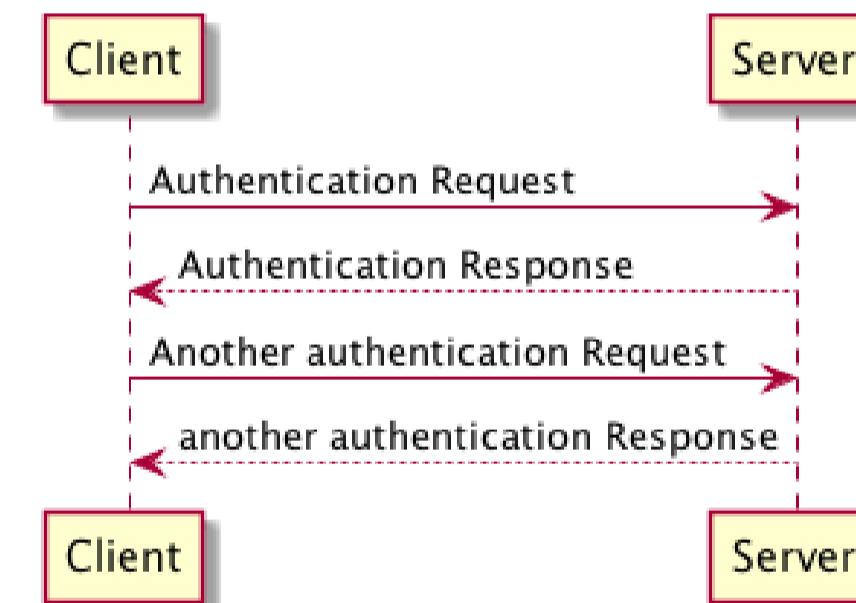
asciidocdoctor-diagram et UML

```
[plantuml]
-----
Class01 "1" *-- "many" Class02 : contains
Class03 o-- Class04 : aggregation
Class05 --> "1" Class06
-----
```



```
[plantuml]
-----
Client -> Server: Authentication Request
Server --> Client: Authentication Response

Client -> Server: Another authentication Request
Client <-- Server: another authentication Response
-----
```



Extensions ruby en java

- TorqueBox RubyGems Maven Proxy Repository

gem asciidoctor-diagram

```
<dependency>
  <groupId>rubygems</groupId>
  <artifactId>asciidoctor-diagram</artifactId>
  <version>1.2.0</version>
  <type>gem</type>
  <scope>provided</scope>
<dependency>
```

AsciidoctorJ et extensions ruby

asciidoc-maven-plugin

```
<plugin>
  <groupId>org.asciidoctor</groupId>
  <artifactId>asciidoctor-maven-plugin</artifactId>
  <version>${asciidoctor.version}</version>
  <configuration>
    <gemPath>${project.build.directory}/gems-
provided</gemPath>
    <requires>
      <require>asciidoctor-diagram</require>
    </requires>
  </configuration>
  ...
</plugin>
```

**Et si je veux faire une extension, je
dois coder en ruby ?**

Et si je veux faire une extension, je dois coder en ruby ?

Nouveau : on peut coder des extensions directement en Java, Groovy, Scala

Et si je veux faire une extension, je dois coder en ruby ?

Nouveau : on peut coder des extensions directement en Java, Groovy, Scala

<http://mrhaki.blogspot.fr/2014/08/awesome-asciidoc-write-extensions-using.html>

Et si je veux faire une extension, je dois coder en ruby ?

Nouveau : on peut coder des extensions directement en Java, Groovy, Scala

<http://mrhaki.blogspot.fr/2014/08/awesome-asciidoc-write-extensions-using.html>

```
twitter:binout[]
```

Et si je veux faire une extension, je dois coder en ruby ?

Nouveau : on peut coder des extensions directement en Java, Groovy, Scala

<http://mrhaki.blogspot.fr/2014/08/awesome-asciidoc-write-extensions-using.html>

```
twitter:binout[]
```

devient :

Et si je veux faire une extension, je dois coder en ruby ?

Nouveau : on peut coder des extensions directement en Java, Groovy, Scala

<http://mrhaki.blogspot.fr/2014/08/awesome-asciidoc-write-extensions-using.html>

```
twitter:binout[]
```

devient :

```
<a href="http://www.twitter.com/binout">@binout</a>
```

<http://asciidoc.org/>

@asciidoc

