Example diagrams

Table of Contents

File tree view

Sadly, this is only available in AsciidocFX for now. So use the editor to produce the image, then import it as-is in your document.

```
cg-asciidoctor-sample
 pom.xml
 src
  docs
   asciidoc
    asciidoc-syntax-reference.adoc
    images
     pause.png
     play.png
     sunset.jpg
    themes
     cg-theme.yml
     favicon.png
     logo-pdf.png
    _includes
     attrs-charref.adoc
     attrs-env.adoc
     ex-table-data.adoc
     ex-table.adoc
  scripts
   asciidocHistory.sh
```

```
root
 photos
  camp.gif
  festival.png
  balloon.jpg
 videos
  car-video.avi
  dance.mp4
  dance01.mpg
  another video.divx
  school videos
   firstday.flv
 documents
  jsfile.js
  powerpoint.ppt
  chapter-01.asc
  archive-db.zip
  .gitignore
  README
  configuration.conf
 etc.
```

Sequence diagram

```
== Initialization ==

Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

== Repetition ==

Alice -> Julie: Another authentication Request
Alice <-- Julie: another authentication Response
```

More examples : http://plantuml.com/sequence-diagram

Use cases diagram

```
left to right direction
skinparam packageStyle rectangle
actor customer
actor clerk
rectangle checkout {
  customer -- (checkout)
  (checkout) .> (payment) : include
  (help) .> (checkout) : extends
  (checkout) -- clerk
}
```

More examples : http://plantuml.com/use-case-diagram

Classes diagram

```
class BaseClass

namespace net.dummy #DDDDDD {
    .BaseClass <|-- Person
    Meeting o-- Person

    .BaseClass <|- Meeting
}

namespace net.foo {
  net.dummy.Person  <|- Person
  .BaseClass <|-- Person

  net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person
```

More examples : http://plantuml.com/class-diagram

Activities diagram

```
start
:ClickServlet.handleRequest();
:new page;
if (Page.onSecurityCheck) then (true)
  :Page.onInit();
  if (isForward?) then (no)
    :Process controls;
    if (continue processing?) then (no)
      stop
    endif

    if (isPost?) then (yes)
      :Page.onPost();
    else (no)
      :Page.onGet();
    endif
    :Page.onRender();
  endif
else (false)
endif

if (do redirect?) then (yes)
  :redirect process;
else
  if (do forward?) then (yes)
    :Forward request;
  else (no)
    :Render page template;
  endif
endif

stop
```

More examples : http://plantuml.com/activity-diagram-beta

Components diagram

```
package "Some Group" {
  HTTP - [First Component]
  [Another Component]
}

node "Other Groups" {
  FTP - [Second Component]
  [First Component] --> FTP
}

cloud {
  [Example 1]
}

database "MySql" {
  folder "This is my folder" {
    [Folder 3]
  }
  frame "Foo" {
    [Frame 4]
  }
}

[Another Component] --> [Example 1]
[Example 1] --> [Folder 3]
[Folder 3] --> [Frame 4]
```

More examples : http://plantuml.com/component-diagram

States diagram

```
scale 600 width

[*] -> State1
State1 --> State2 : Succeeded
State1 --> [*] : Aborted
State2 --> State3 : Succeeded
State2 --> [*] : Aborted
state State3 {
  state "Accumulate Enough Data\nLong State Name" as long1
  long1 : Just a test
  [*] --> long1
  long1 --> long1 : New Data
  long1 --> ProcessData : Enough Data
}
State3 --> State3 : Failed
State3 --> [*] : Succeeded / Save Result
State3 --> [*] : Aborted
```

More examples : http://plantuml.com/state-diagram

Objects diagram

```
object Object01
object Object02
object Object03
object Object04
object Object05
object Object06
object Object07
object Object08

Object01 <|-- Object02
Object03 *-- Object04
Object05 o-- "4" Object06
Object07 .. Object08 : some labels
```

More examples : http://plantuml.com/object-diagram

User interface

```
@startsalt
{+
{* File | Edit | Source | Refactor
 Refactor | New | Open File | - | Close | Close All }
{/ General | Fullscreen | Behavior | Saving }
{
    { Open image in: | ^Smart Mode^ }
    [X] Smooth images when zoomed
    [X] Confirm image deletion
    [ ] Show hidden images
}
[Close]
}
@endsalt
```

More examples : http://plantuml.com/salt

Deployment diagram

```
actor actor
agent agent
boundary boundary
cloud cloud
component component
control control
database database
entity entity
file file
folder folder
frame frame
interface  interface
package package
queue queue
artifact artifact
rectangle rectangle
storage storage
usecase usecase
skinparam rectangle {
    roundCorner<<Concept>> 25
}

rectangle "Concept Model" <<Concept>> {
    rectangle "Example 1" <<Concept>> as ex1
    rectangle "Another rectangle"
}
node node1
node node2
node node3
node node4
node node5
node1 -- node2
node1 .. node3
node1 ~~ node4
node1 == node5
```