Communiqué de presse

Le smartphone, vecteur privilégié pour intéresser les jeunes à la programmation.

Strasbourg, 2 septembre 2019 - **L'apprentissage de la programmation réinventé grâce au *toolkit Atlas*, un outil sans pareil pour écrire facilement des programmes directement utilisables sur smartphone.**

Des programmes dotés d'une interface graphique

Avec le *toolkit Atlas*, un débutant écrira des programmes pourvus d'une véritable interface graphique. Celle-ci s'ouvre automatiquement dans un navigateur lors du lancement du programme, facilitant son accès.

Utiliser ses programmes avec son smartphone

L'apprenti-développeur pourra utiliser son programme sur son smartphone grâce à une *URL* générée par le *toolkit Atlas*. Cette *URL* est affichée sous forme de code *QR* par le programme, pour faciliter son ouverture sur son smartphone.

Partager l'accès à ses programmes

En transmettant cette *URL* à ses amis, parents, professeurs…, ces derniers pourront utiliser son programme sur leur propre smartphone. Tout comme une véritable application web, le programme est capable de gérer, de manière transparente pour le développeur, plusieurs utilisateurs simultanément.

Accès facile et rapide aux programmes

Grâce au *toolkit Atlas*, ses programmes sont automatiquement et instantanément utilisables de n'importe où sur internet. Nul besoin de configurer son smartphone, la box internet ou un quelconque routeur, ni de déployer le programme sur un serveur distant. Connectez simplement à internet l'ordinateur sur lequel s'exécute le programme, et son *URL* suffit pour y avoir accès.

Disponible dans plusieurs langages

Le *toolkit Atlas* est conçu pour accompagner l'apprentissage de la programmation quel que soit le langage utilisé. Actuellement, le *toolkit Atlas* existe pour *Java*, *Node.js*, *PHP*, *Python* et *Ruby*. L'*API* est la même quel que soit le langage, et, à l'avenir, encore plus de langages bénéficieront du *toolkit Atlas*.

Communiqué de presse

Un *toolkit* léger

Quel que soit le langage, le *toolkit Atlas* ne pèse que quelques dizaines de kilooctets et tourne facilement sur des machines peu puissantes, comme le *Raspberry Pi Zero W*. Le *toolkit Atlas* conférera une nouvelle dimension aux programmes de pilotage de circuits électroniques ou de robots (*Poppy Ergo Jr*, p. ex.).

Des standards éprouvés et répandus

Il suffit de savoir créer une page web basique pour pouvoir utiliser le *toolkit Atlas*, qui a été conçus pour s'appuyer sur des standards faciles à apprendre, et pour lesquels on trouve multitude de tutoriels de tous niveaux. En outre, ces standards font partie de la culture générale de tout développeur.

Un *toolkit* facile à utiliser et libre

Les exercices et exemples de programmes proposés par les tutoriels d'apprentissage de la programmation peuvent facilement être modifiés pour tirer profit *toolkit Atlas*. L'ensemble du *toolkit Atlas* étant disponible sous licence libre, il peut facilement être modifié pour être adapté à des besoins particuliers.

Termux

Pour les plus motivés disposant d'un appareil sous *Android*, ils peuvent y installer une application nommée *Termux*, un émulateur de terminal, grâce auquel ils disposeront, sur leur tablette ou leur smartphone, d'un véritable environnement de développement capable d'exécuter tous leurs programmes.

Pour aller plus loin

Sur le site dédié au *toolkit Atlas* (https://atlastk.org), des exemples de programmes développés avec le *toolkit Atlas* (en particulier l'application du projet *TodoMVC*) sont disponibles, ainsi que des démonstrations en ligne qui permettent de tester le *toolkit Atlas* dans un navigateur web, sans rien avoir à installer.

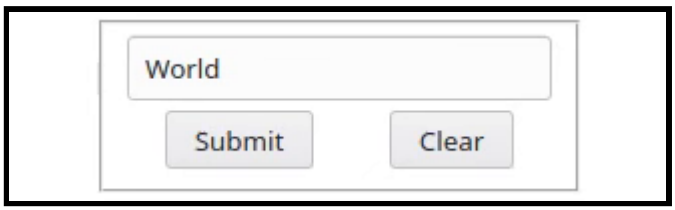
Vous y trouverez également la procédure d'installation détaillée pour chacun des langages pour lequel le *toolkit Atlas* est disponible, ainsi qu'une description détaillée de son *API*. Les liens vers le dépôt *GitHub* de chacun des composants du *toolkit Atlas* y sont également indiqués.

Ce projet a été développé par Claude SIMON, ingénieur logiciel résidant près de Strasbourg, que vous pouvez joindre pour tout renseignement complémentaire à : simon.claude@q37.info.

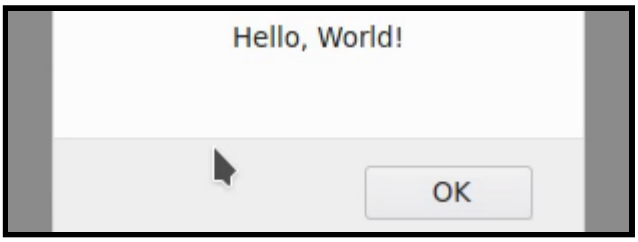
Pour la dernière version de ce communiqué : https://q37.info/s/kai4eehi.

Annexe

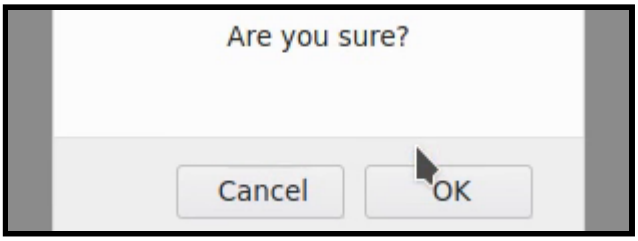
À titre d'exemple, voici à quoi ressemble un *Hello, World!* développé avec le *toolkit Atlas* (pour une demonstration en ligne : https://q37.info/s/oe1hipoh) :



Après un appui sur le bouton *Submit* :



Après un appui sur le bouton *Clear* :



Vous trouverez ci-dessous le code source pour les différents langages pour lesquels le *toolkit Atlas* est disponible ; le code *HTML* a été simplifié pour en réduire la taille.

- Version *Java*
- Version *JavaScript* pour *Node.js*
- Version *Perl*
- Version *Python*
- Version *Ruby*

Version *Java*

```
import info.q37.atlas.*;

class Hello extends Atlas {
  private static String body =
    "<input id=\"input\" data-xdh-onevent=\"Submit\"/>" +
    "<button data-xdh-onevent=\"Submit\">Submit</button>" +
    "<button data-xdh-onevent=\"Clear\">Clear</button>";

  @Override
  public void handle(String action, String id)
  {
    switch( action) {
    case "":
      dom.inner("", body);
      break;
    case "Submit":
      dom.alert("Hello, " + dom.getContent("input") + "!" );
      break;
    case "Clear":
      if ( dom.confirm("Are you sure ?") )
        dom.setContent("input", "");
      break;
    }

    dom.focus("input");
  }

  public static void main(String[] args) throws Exception {
    launch(() -> new Hello());
  }
}
```

Pour l'exécuter :

- `git clone https://github.com/epeios-q37/atlas-java`
- `cd atlas-java`
- y stocker le code source ci-dessus dans un fichier nommé `Hello.java`
- `javac -cp Atlas.jar Hello.java`
- puis, en fonction du système d'exploitation :
  - *Windows* : `java -cp .;Atlas.jar Hello`
  - autres : `java -cp .:Atlas.jar Hello`

Voir également https://github.com/epeios-q37/atlas-java.

Version *JavaScript* pour *Node.js*

```
const atlas = require( 'atlastk' );

const body = `
<input id="input" data-xdh-onevent="Submit"/>
<button data-xdh-onevent="Submit">Submit</button>
<button data-xdh-onevent="Clear">Clear</button>
`;

const callbacks = {
    "": (dom, id) => dom.inner("", body,
      () => dom.focus("input")),
    "Submit": (dom, id) => dom.getContent("input",
      (name) => dom.alert("Hello, " + name + "!",
        () => dom.focus("input"))),
    "Clear": (dom, id) => dom.confirm("Are you sure ?",
      (answer) => {
        if (answer) dom.setContent("input", "");
        dom.focus("input");
      })
};

atlas.launch(() => new atlas.DOM(), callbacks);
```

Pour l'exécuter :

- `git clone https://github.com/epeios-q37/atlas-node`
- `cd atlas-node`
- y stocker le code source ci-dessus dans un fichier nommé `Hello.js`
- `npm install`
- `node Hello.js`

Voir également https://github.com/epeios-q37/atlas-node.

Version *Perl*

```perl
use Atlas;

my $body = '
<input id="input" data-xdh-onevent="Submit"/>
<button data-xdh-onevent="Submit">Submit</button>
<button data-xdh-onevent="Clear">Clear</button>
';

sub acConnect {
    my ($hello, $dom) = @_;

    $dom->inner("",$body);
    $dom->focus("input");
}

sub acSubmit {
    my ($hello, $dom) = @_;

    $dom->alert("Hello, " . $dom->getContent("input") . "!");
    $dom->focus("input");
}

sub acClear {
    my ($hello, $dom) = @_;

    if ( $dom->confirm("Are you sure?") ) {
        $dom->setContent("input", "");
    }

    $dom->focus("input");
}

my %callbacks = (
    "" => \&acConnect,
    "Submit" => \&acSubmit,
    "Clear" => \&acClear,
);

Atlas::launch(\%callbacks);
```

Pour l'exécuter :

- `git clone https://github.com/epeios-q37/atlas-perl`
- `cd atlas-perl/examples`
- y stocker le code source ci-dessus dans un fichier nommé `Hello.pl`
- `perl -I atlastk Hello.pl`

Version *Python*

```
import atlastk as Atlas

body = """
<input id="input" data-xdh-onevent="Submit"/>
<button data-xdh-onevent="Submit">Submit</button>
<button data-xdh-onevent="Clear">Clear</button>
"""

def acConnect(this, dom, id):
  dom.inner("", body)
  dom.focus("input")

def acSubmit(this, dom, id):
  dom.alert("Hello, " + dom.getContent("input") + "!")
  dom.focus("input")

def acClear(this, dom, id):
  if ( dom.confirm("Are you sure?") ):
    dom.setContent("input", "")
  dom.focus("input")

callbacks = {
  "": acConnect,
  "Submit": acSubmit,
  "Clear": acClear,
}

Atlas.launch(callbacks)
```

Pour l'exécuter :

- stocker le code source ci-dessus dans un fichier nommé `Hello.py`
- `pip install atlastk`
- `python Hello.py`

Voir également https://github.com/epeios-q37/atlas-python.

Version *Ruby*

```ruby
require 'Atlas'

$body =
<<~HEREDOC
<input id="input" data-xdh-onevent="Submit"/>
<button data-xdh-onevent="Submit">Submit</button>
<button data-xdh-onevent="Clear">Clear</button>
HEREDOC

def acConnect(userObject, dom, id)
  dom.inner("", $body)
  dom.focus("input")
end

def acSubmit(userObject, dom, id)
  dom.alert("Hello, " + dom.getContent("input") + "!")
  dom.focus("input")
end

def acClear(userObject, dom, id)
  if dom.confirm?("Are you sure?")
    dom.setContent("input", "")
  end
  dom.focus("input")
end

callbacks = {
  "" => method(:acConnect),
  "Submit" => method(:acSubmit),
  "Clear" => method(:acClear),
}

Atlas.launch(callbacks)
```

Pour l'exécuter :

- `git clone https://github.com/epeios-q37/atlas-ruby`
- `cd atlas-ruby`
- y stocker le code source ci-dessus dans un fichier nommé `Hello.rb`
- `ruby Hello.rb`

Voir également https://github.com/epeios-q37/atlas-ruby.