DeployGround:
A Framework for Streamlined Programming from API Playgrounds to Application Deployment

Jun Kato, Masataka Goto
National Institute of Advanced Industrial Science and Technology (AIST), Japan
VL/HCC 2018 Short paper (10 min talk)
Let me do the demo first!
Coding tutorials and references

• Much work on creating tutorials in the context of HCI
  [Chi et al., UIST ‘12] [Lafreniere et al., CHI ‘13] [Chi et al., UIST ‘13] [Kim et al., CHI ‘14]

• Only a handful of work on creating coding tutorials
  [Harms et al., IDC ‘13] [Head et al., VL/HCC ‘15] [Gordon et al., VL/HCC ‘15]

See https://junkato.jp/deployground for the complete list of references.

Contributions:
• A framework to create interactive web-based coding tutorials
• Its concrete implementation techniques
API documentations and tutorials

• Concrete usage information is beneficial
  [Robillard, 2009] [Hou et al., ICPC ’11] [Robillard et al., 2011] [Wang et al., MSR ’13]

• Executable example codes are especially beneficial
  [Subramanian et al., ICSE ’14]

• Some documentations and tutorials provide "playgrounds"
  [Khan Academy] [TypeScript Playground] [Vimeo API Playground] [W3School] [tutorialspoint] …
<html>
<body>
<p id="demo">Click the button to change the layout of this paragraph</p>
<script>
function myFunction() {
    var x = document.getElementById("demo");
    x.style.fontSize = "25px";
    x.style.color = "red";
}
</script>
<button onclick="myFunction()">Click Me!</button>
</body>
</html>
```c
#include <stdio.h>

int main() {
    /* printf() function to write Hello, World! */
    printf("Hello, world!");
    printf("Hello, world!");
}
```

Compiling the source code....
```
gcc main.c -o demo -lm -pthread -lgmp -lthread 2>&1
```

Executing the program....
```
demo
Hello!Hello, World!
```
Programming

Programming the "master" client

Once you have tokens, you can start writing JavaScript code. Please edit the code below (replace the dummy tokens with yours) and click the button.

```javascript
function onsinglewidgetloadReady(singlewidget) {
    // Show the music player
    var player = new SingleWidget.Player(
        {accessToken: 'foo' // Access token
        secretToken: 'bar' // Secret token
    );
    player.useMedia('https://youtube.com/watch?v=OMplYnLyaIg');
    player.addPlugin(new SingleWidget.Plugin.SongSync());
    // Start playing music when the page load completes
    player.on('loadevent', function() {
        if (player.isPlaying()) {
            player.play()
        }
    });
}
```

master test

Playback position: 0[ms]
API Playgrounds

- A part of coding tutorials (API documentations or tutorials)
- A code editor and its output sit next to each other
- The output can be interactively updated upon the user's request
Limitations of existing tutorials

• Learning APIs is supported well with the interactive playgrounds meanwhile…

• The programmer needs to leave the tutorial at some point
• S/he needs to re-start the development in their own environment.

Support for seamless post-learning experience is in need
Detailed limitations of existing tutorials

- Toy sandbox OR expensive sandbox
- Ephemeral code
- No support for deployment
- Little social interaction

What are these and how can we address them?
Click the button to change the layout of this paragraph

function myFunction() {
    var x = document.getElementById("demo");
    x.style.fontSize = "25px";
    x.style.color = "red";
}

button onclick="myFunction()">Click Me!</button>
Virtual Machine on the server
Programming

Programming the "master" client

Once you have tokens, you can start writing JavaScript code. Please edit the code below (replace the dummy tokens with yours) and click the button.

---

Sngle Sync tutorial (DeployGround-based tutorial)
Toy sandbox OR expensive sandbox?

Toy: cannot used for non-browser languages

Expensive: requires lots of server-side resources
LIMITATION (2)

Toy sandbox OR expensive sandbox?

1. Toy: cannot used for non-browser languages
2. Expensive: requires lots of server-side resources
3. Flexible: can potentially support any APIs
Pseudo runtime environment
for making a flexible sandbox

- requires manual implementation of the emulation layer
- yet has potential to emulate anything that can be represented (visualized) on a web browser

- e.g.,
  - Node.js-based web servers: usually requires dedicated domain names
  - Physical computing devices: usually requires purchasing modules
Ephemeral code?

Download each code editor content as a file

VM sessions cannot be (easily) exported
Ephemeral code?

1. Download each code editor content as a file
2. VM sessions cannot be (easily) exported
3. Save the entire workspace as a GitHub repo
No support for deployment?

HTML with JS files cannot be directly opened with modern web browsers

In more complex cases (e.g. Node.js-based projects), only instructions are presented
LIMITATION (3)

No support for deployment?

HTML with JS files cannot be directly opened with modern web browsers

In more complex cases (e.g. Node.js-based projects), only instructions are presented

Deploy to GitHub Gist or Heroku
Adaptive boilerplate for exporting files as an executable project

Deployment target
- Heroku
  http://....herokuapp.com/...
- RawGit
  http://cdn.rawgit.com/...

Online storage
- GitHub
  (Node.js projects)
- GitHub Gist
  (Static files)

Tutorial content
(Static HTML/CSS/JavaScript files)

Tutorial users

Tutorial developers

DeployGround (tutorial system)
Little social interaction?

"Achievement unlocked!" kind of posts can be potentially made on social networking services

Codebases and apps can be shared instantly
Reversible software engineering
to use people's outcome as educational resource

**Deployment target**
- Heroku
  - http://...herokuapp.com/...
- RawGit
  - http://cdn.rawgit.com/...

**Online storage**
- GitHub
  - (Node.js projects)
- GitHub Gist
  - (Static files)

**Tutorial content**
- (Static HTML/CSS/JavaScript files)

**Tutorial users**
- Tutorial users

**Tutorial developers**
- Tutorial developers

**DeployGround (tutorial system)**
In this tutorial, you will learn how to write a Node.js-based web server that hosts a "master" client and returns the HTML code that renders a video player serving as a "slave" client.

**DeployGround**

1) pseudo-runtime environment

2) adaptive boilerplate

3) reversible software engineering
DeployGround

- Live Programming
  1) pseudo-runtime environment
- Save and deploy (or download)
  2) adaptive boilerplate

Streamlined support

- Import
  3) reversible software engineering

Deployed applications

Execution results

User A

User B

Learn

Conventional tutorials

Develop

Programming environments

Deploy

←from learning

to deployment→
Preliminary user feedback

Asked 3 software engineers and 2 researchers to answer prequestionnaire and try out the tutorial

Asked 24 university students to form 6 groups and prototype applications in 2 days

- All of them successfully benefited from the framework
- Potential applications: APIs with expensive initial cost to try out
- Requests for more detailed views on save/deploy features
- Emulation is imperfect; more diverse examples are demanded
DeployGround framework

- **Pseudo runtime environment** enables live programming experience. Learners can enjoy testing the APIs. They accumulate their codebase during the tutorial.

- **Adaptive boilerplate** brings the experience out of the sandbox. In the end, they get the archived project files. The tutorial even helps them deploy the project.

- **Reversible software engineering** allows to gain benefits from social coding. All of the above experiences can be easily shared on the web, helping the other learners.
DeployGround:
A Framework for Streamlined Programming from API Playgrounds to Application Deployment

In this tutorial, you will learn how to write a Node.js-based web server that hosts a "master" client and returns the HTML code that renders a video player serving as a "slave" client.

Code editor

Appendix

Execution results

Deployment applications

User A

3) Reversible software engineering

User B

AIST
"Playground" implementations

<table>
<thead>
<tr>
<th>Performance</th>
<th>Browser-native</th>
<th>Huge latency</th>
<th>A little overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debuggability</td>
<td>Very low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Supported APIs:</td>
<td>Complete (browser-based APIs)</td>
<td>Complete (CUI-based APIs)</td>
<td>Partial (any emulatable APIs)</td>
</tr>
</tbody>
</table>
"Playground" implementations

<table>
<thead>
<tr>
<th>Performance</th>
<th>Browser-native</th>
<th>Huge latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debuggability</td>
<td>Very low</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Supported APIs:**
- Complete (browser-based APIs)
- Complete (CUI-based APIs)