Eclipse Che Pair Programming Extension

1. Introduction

This extension enables pair programming feature for Eclipse Che. The extension enables following two features.

- Display cursors of other participants.
- Participants getting notified when the files which are not opened by them are getting modified at the moment.

2. Demo (https://youtu.be/OZNceE-sWto)

2.1. Requirements

- Java 8
- Maven 3
- Npm
- Docker

2.2. To make the demo up and running

2.2.1. Building the product

- Clone my forked repository (https://github.com/rnavagamuwa/che.git)
- Go into the cloned folder and build the product.
- Launch Eclipse Che
  Use this script to do the above steps
  (https://gist.github.com/rnavagamuwa/794ab734f84365a82268b87ed320ca1b)

2.2.2. Enable extension in Eclipse Che

- Create a workspace with Eclipse Flux. Select the Flux stack from the stacks library.
  (Url to Dockerfile :
  https://hub.docker.com/r/rnavagamuwa/eclipse-flux-for-pair-programming/~/dockerfile/)
- Start the workspace, Open in IDE and create a new project. You can simply select a sample project in the previous step.
- Start the flux custom command.
- Open a file to edit.
- Open a new browser windows to the same workspace, open the same file to edit.

3. Implementation

3.1. Communication Protocol

Eclipse Flux is used as the communication protocol for this extension. Eclipse Flux is a message based architecture for cloud based developer tooling. Every tools, called participants, will exchange JSON messages to follow a specific use-case. Several use-cases are available for demo: the pair-programming use case is one of them. Every participants will connect to a central Flux server and all the messages will be routed by this server. The server is implemented with node.js, websockets (socket.io) and RabbitMQ as a message broker. Each message is named and can be broadcasted or send to one specific participant.

In the pair programming use-case, when communicating using Eclipse Flux mainly two event types are being used.

- **liveResourceChanged**
- **liveCursorOffsetChanged** : This event type wasn't there in Eclipse Flux. This was added to the Eclipse Flux codebase, in order to support this plugin. (https://github.com/eclipse/flux/commits/master?author=rnavagamuwa)

Each message passing through flux will have the following information.

1. String fullPath;
2. String addedCharacters;
3. int offset;
4. int removeCharCount;
5. String project;
6. String resource;
7. String username;
8. String channelName;

A sample message is as follows

```
{
  "name": "liveResourceChanged",
  "args": [
    {
      "username": "USER",
      "project": "helloworld",
      "resource":
"src/main/java/com/codenvy/example/java/App.java",
      "offset": 120,
      "removedCharCount": 0,
      "addedCharacters": "h"
    }
  ]
}
```

Figure 1 : Sample message

3.2. CSS styles

- In this plugin annotations have been used to create the cursors. In order to do this provided api (org.eclipse.che.ide.api.editor.texteditor.HasTextMarkers) has been used. In this plugin five different CSS styles has been used as cursors (Up to five users are supported).
- CSS styles have been injected using StyleInjector class (***com.google.gwt.dom.client.StyleInjector***)

3.3. Document change event

- Document change event has been used got notified when there is a change in the document. When there is a change in the document, a new message will be sent via flux with the event name, "***liveResourceChanged***".
- Whenever flux receives a new message with the event name as "***liveResourceChanged***", a new annotation will be added.

3.4. Cursor offset change event

- Since cursor offset change is not same as the resource change, a new message type has been introduced into Eclipse Flux. Cursor offset change message will be sent to with the event name, "***liveCursorOffsetChanged***".
- "***org.eclipse.che.ide.flux.liveedit.CursorModelForPairProgramming***" is used to get notified when there is a cursor offset change. This class implements "***CursorModelWithHandler***" and "***CursorActivityHandler***" classes.

3.5. Remove and adding new cursor annotations

- "***CursorHandlerForPairProgramming***" class is used to track the annotation that has been used before. When there is a **cursor offset change** or a **live resource change** previous annotation is removed and a new annotation will be added.
- Removing and adding annotations are done using "***org.eclipse.che.ide.api.editor.texteditor.HasTextMarkers***" class.

3.6. Message flow diagram of the extension

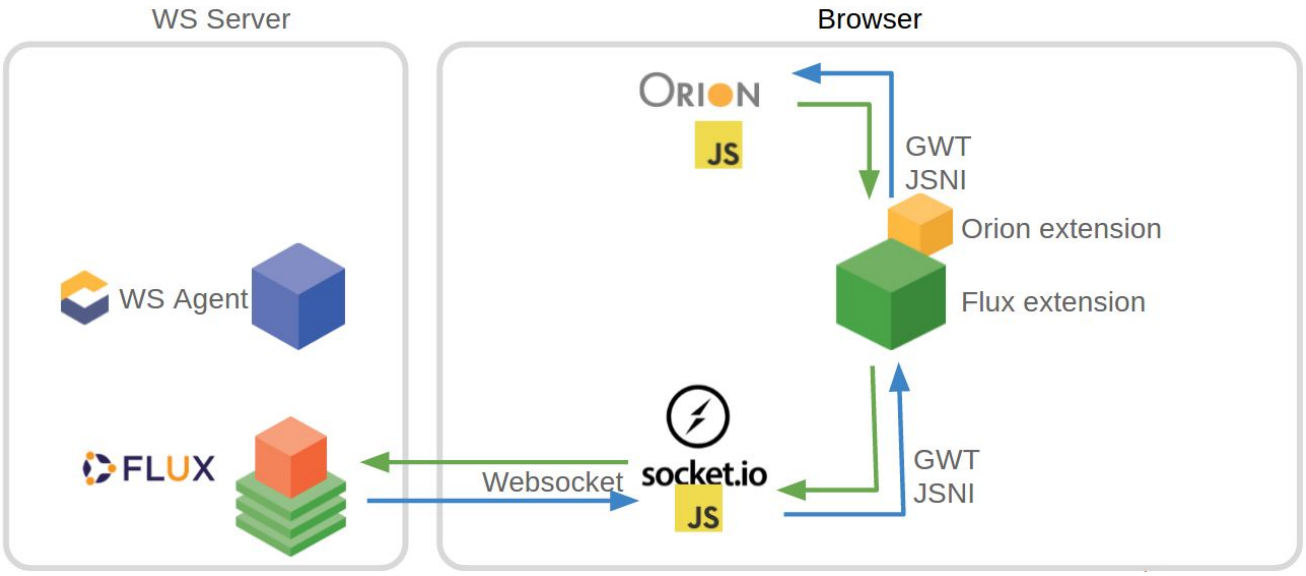

Figure 2 : message flow diagram of pair programming plugin

3.7. Important urls

- Pull request : https://github.com/eclipse/che/pull/2131
- Plugin : https://github.com/sunix/che-plugin-flux-live-edit
- GSoC final submission : http://www.rnavagamuwa.com/open-source/gsoc-2016-final-submission-pair-programming-with-eclipse-che-cloud-ide/

4. Work in progress

- This extension should be split into two.
    - One should be the pair programming extension. User should be able to use this extension with any communication protocol.
    - The other one should be the protocol which is used for communications. In this case it's Eclipse Flux.
- Collision management
- Convert the Flux runtime to an Eclipse Che agent.
- Add FS sync to the Flux runtime.
- Handle the message "***getLiveResourcesRequest***" and "***getLiveResourcesResponse***" to retrieve live document content that may not be saved.

5. Contributors

- Sun Seng David TAN sunix@sunix.org
- Randika Navagamuwa randikanavagamuwa@gmail.com