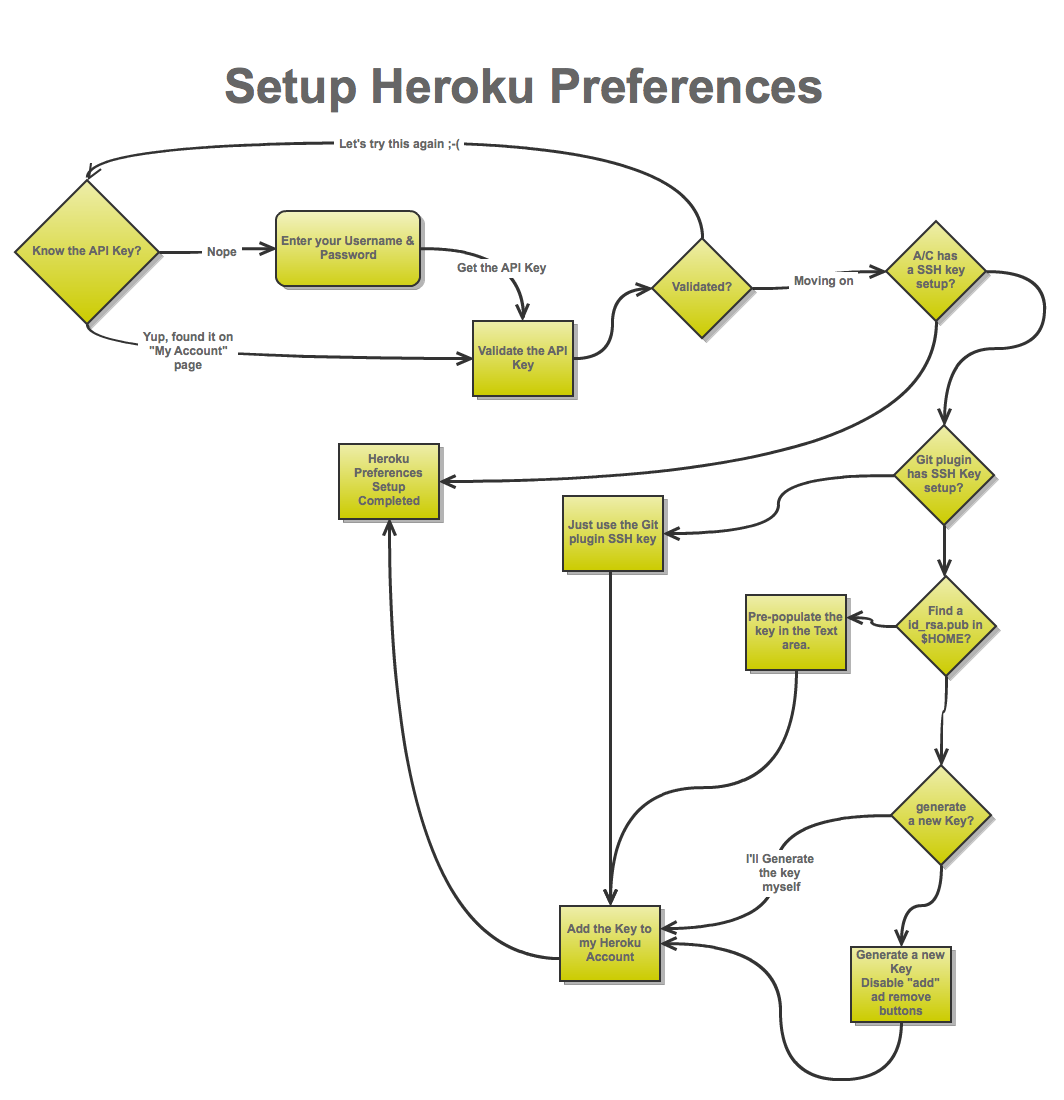
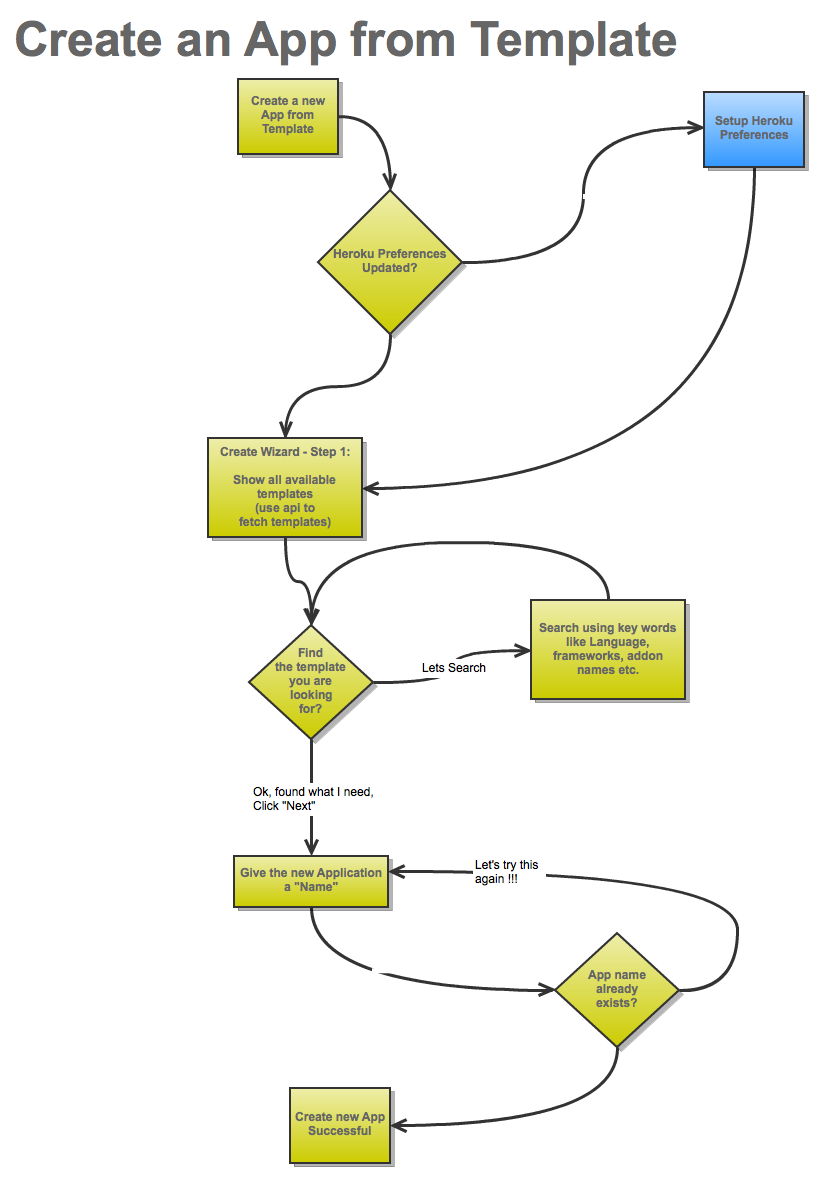
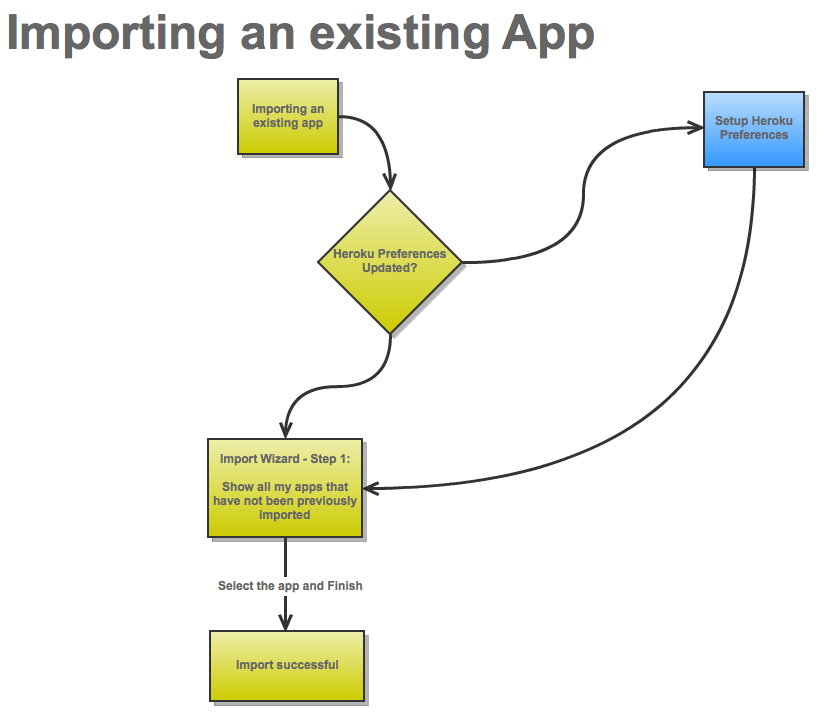
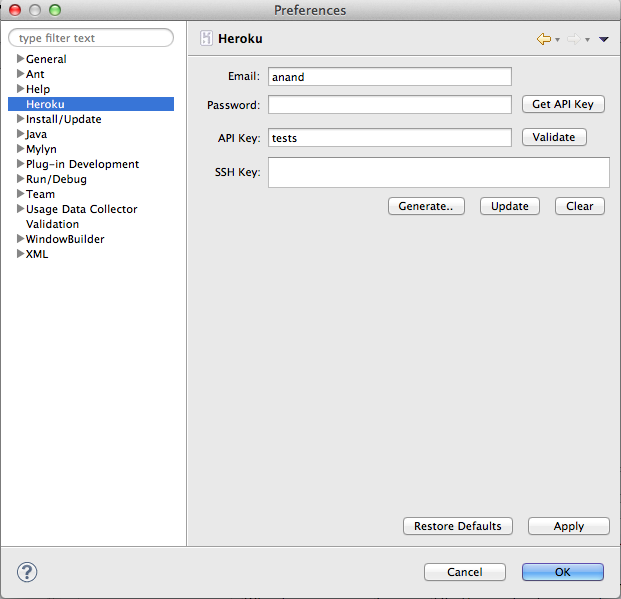
1. Overall Developer Experience Flow







1. Setup



4

2

3

5

7

6

1

|  |  |
| --- | --- |
| 1 | Accessing Heroku Preferences |
| Description | This allows for the developer to navigate to the preferences for Heroku. At this time this is limited to the API Key and the SSH key for “git” repo related tasks. |
| Validation | None. |
| Post Event | None. |

|  |  |
| --- | --- |
| 2 | Get API Key |
| Description | If the developer does not know their API key before hand, they can put in username/password and click on this button to fetch the API key. |
| Validation | * Username should be required * Password is required |
| Event Action | On click should perform a “/login” call with the username and password to fetch the API key. |
| Post Event | If the login is successful, the API Key should be stored in the Eclipse Workspace preferences.  The name of the preference should be “heroclipse” (Note: a “.prefs” is appended automatically) |

|  |  |
| --- | --- |
| 3 | Validate |
| Description | Validates if the API key value is valid. This only really applicable when a developer puts their API key without getching it using the “Get API Key” button. |
| Validation | * API key field is required |
| Event Action | On click, should perform a “list apps” API call to ensure that it returns without any errors. |
| Post Event |  |

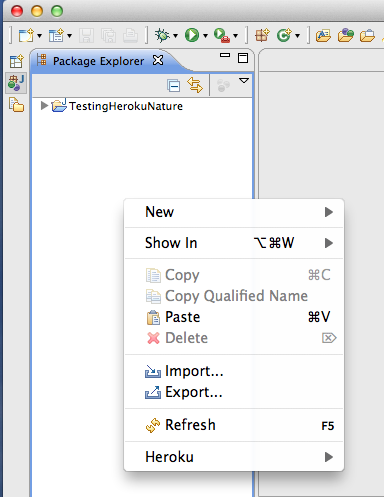
|  |  |
| --- | --- |
| 4 | SSH Key |
| Description | Pre-poplulates the SSK key from the location that was previously used. In case this is the first time, it should look at $HOME/.ssh/id\_rsa.pub and populate the same |
| Validation |  |
| Event Action |  |
| Post Event |  |

|  |  |
| --- | --- |
| 5 | Generate Key **(This is not required as Eclipse offers a SSH key generation mechanism).** |
| Description | Generates a SSH key pair that will be used for communicating with the Git repo. |
| Validation |  |
| Event Action | On click, generate a SSH key (similar to what ssh-keygen) does. |
| Post Event | Store the SSH key pair under the workspace preferences. It should be stored in the “heroclipse.prefs” file. |

|  |  |
| --- | --- |
| 6 | Update |
| Description | Adds the SSH key to the corresponding Heroku account. |
| Validation |  |
| Event Action | On click send a “/keys/add” API call with the SSH details |
| Post Event | Display a “success” or “error” dialog when the key is successfully added to the Heroku account. |

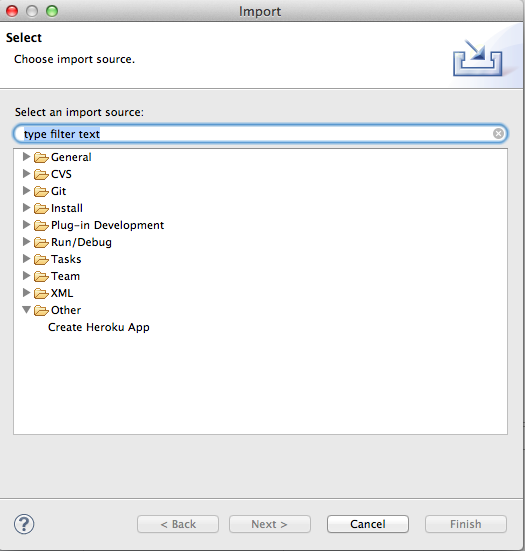
|  |  |
| --- | --- |
| 7 | Clear |
| Description | Clears all SSH keys on the corresponding Heroku account. |
| Validation |  |
| Event Action | Clears out all SSH Keys from the corresponding Heroku account |
| Post Event | Display a “success” or “error” dialog when keys are cleared from the Heroku account. |

1. Import an Existing Heroku App



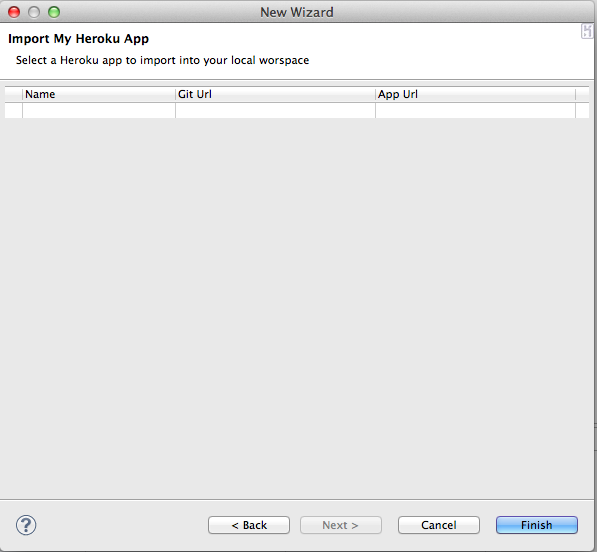
8

|  |  |
| --- | --- |
| 8 | Import |
| Description | Initiates the Import process for an existing Heroku app |
| Validation |  |
| Event Action | None. |
| Post Event | Displays the Import Wizard. |



9

|  |  |
| --- | --- |
| 9 | Import Heroku App |
| Description | Initiates the Import process for an existing Heroku app |
| Validation |  |
| Event Action | A new “import wizard” needs to be displayed under a “Heroku” category with a single sub item called “Existing Heroku App” |
| Post Event | Displays a list of available apps associated with the Heroku Account |



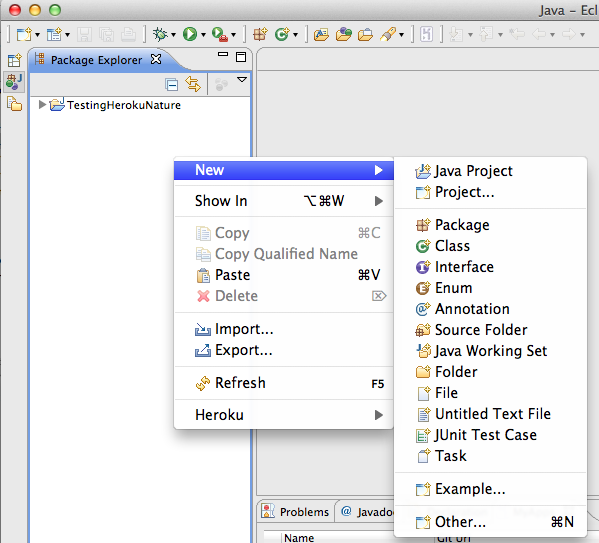
11

10

|  |  |
| --- | --- |
| 10 | Import Heroku App – Step 2 |
| Description | Lists all available Heroku Apps that have not previously been imported into the current workspace. |
| Validation | If there is no API Key or if the API Key is invalid, display an error dialogue with the suitable message. |
| Event Action | All the apps associated with the app. The following about each app should be listed in the table.  Application Name  Application URL  Git URL (???)  Framework : |
| Post Event | The “Finish” button should only be enabled if a row in the list of apps is selected. |

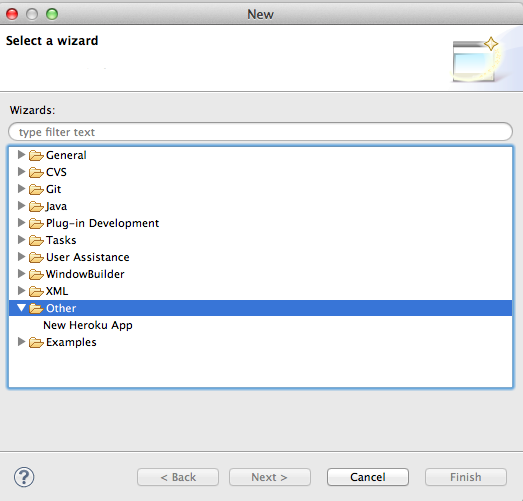
|  |  |
| --- | --- |
| 11 | Import Heroku App – Step 3 |
| Description | Completes the “import” of an existing Heroku app into the developer’s workspace. |
| Validation | A app should be selected in the list of apps available for “import” |
| Event Action | Should do the following:   1. Do a “git clone” of the corresponding App URL 2. Create the .project with JavaNature & Java builder to the .project file. [TBD Need to provide a template] 3. If the app is a Java project using a pom.xml, it should also add a “Maven Nature” & Maven builder to the .project file [TBD Need to provide a template] 4. Generate a “.classpath” file associating the default runtime environment as well as the maven classpath container [TBD Need to provide a template] 5. The import process should be done as a background process. |
| Post Event | Once the project is imported successfully the new project should appear in the list of projects in the workspace and should also compile/build successfully. |

1. Create new App from Template



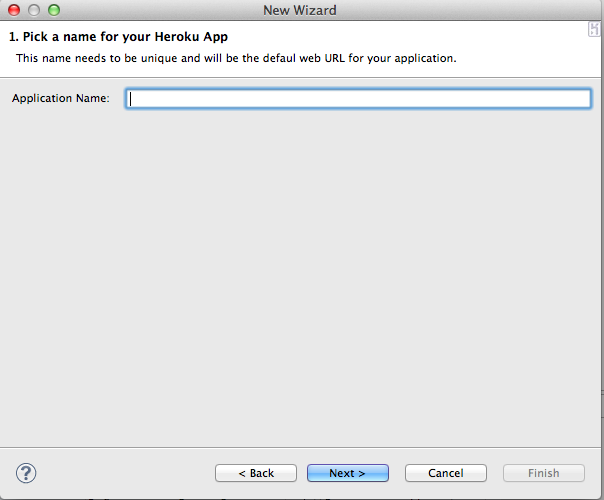
12

|  |  |
| --- | --- |
| 12 | Create Heroku App |
| Description | Initiates the navigation to create a new Heroku app from a predefined template. |
| Validation |  |
| Event Action | Opens the “Select a wizard” standard Eclipse Window |
| Post Event |  |



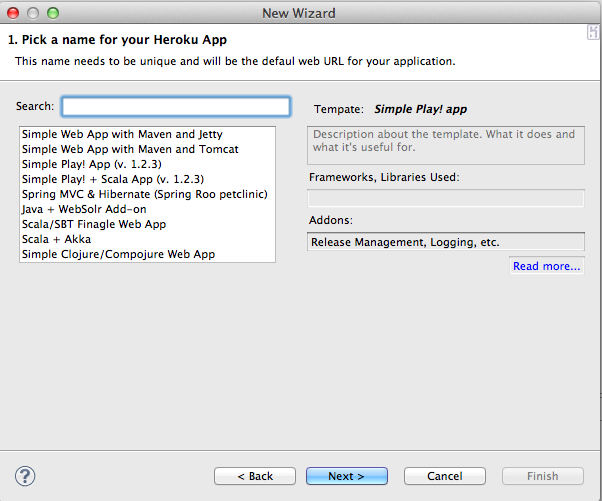
13

|  |  |
| --- | --- |
| 13 | Create Heroku App – Step 2 |
| Description | This starts a new Heroku App creation process. |
| Validation |  |
| Event Action | Opens the “Create new App” wizard |
| Post Event |  |



13

|  |  |
| --- | --- |
| 13 | Create Heroku App – Step 3 |
| Description | Pick an application name for the new application. |
| Event | Single Click |
| Validation | If there is no valid API Key found in the workspace preferences, on click of “Next”, the developer should see a “Error” dialog to setup their API key through preferences. |
| Event Action | Opens the “Select an App Template” page |
| Post Event | Create an app and ensure that the name entered by the developer is available.  If the name is not available, display an “Error” dialog. |



14

15

17

16

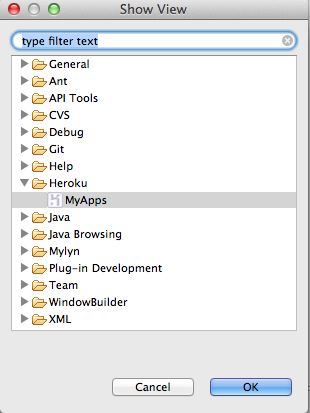
|  |  |
| --- | --- |
| 14 | List of templates |
| Description | Allows for a list of pre-created templates that has the base code etc. to get started with an application on Heroku. |
| Validation |  |
| Event Action | The list of app templates will be fetched from <apptempaltes.herokuapp.com>  TBD: For first release may be the list of templates are a finite list with the meta data (i.e. template name etc. )hardcoded into the plugin. |
| Post Event |  |

|  |  |
| --- | --- |
| 15 | Search for a template |
| Description | Allows developers to search for app templates and pick the right one. |
| Validation |  |
| Event Action | The list of app templates will be fetched from <apptempaltes.herokuapp.com> based on the search criteria.  Search will be performed once the “Return/ Enter” key is pressed on the text box. |
| Post Event | List the templates that match the search criteria. |

|  |  |
| --- | --- |
| 16 | Template detail |
| Description | View detail about the app template |
| Validation |  |
| Event Action | Once a developer clicks on a specific template, the details of the template are displayed. The following details would be displayed:   * Template Name * Language * Description: TBD (Need to put a max character limit) * Public Github Repo * Addons : If the template uses addons (??? Not sure if this should be a template) * Read more: If there’s a devcenter article/public URL that talks more about the template. |
| Post Event |  |

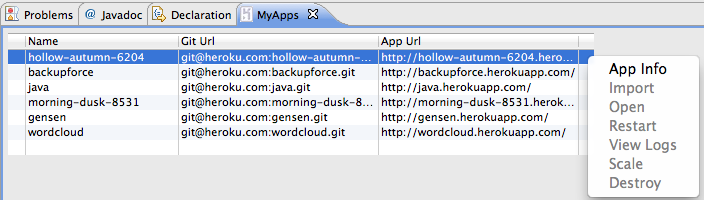
|  |  |
| --- | --- |
| 17 | Finish App creation |
| Description | This completes the app creation process. |
| Validation |  |
| Event Action | Depending on the app template selected, this would do a “git clone” of the corresponding template.  TBD: If the template requires additional entries in the IDE settings that needs to be added as well. For the most part in Java this should not be an issue as pom.xml would take of dependencies automatically. |
| Post Event | App is created and available on the project workspace. |

1. View my Heroku Apps



18

|  |  |
| --- | --- |
| 18 | View my Heroku Apps |
| Description | This lists all available apps that are associated with this Heroku account.  The view is opened through **Window > Open View > Heroku > My Heroku Apps** |
| Validation | If no API key is found or is invalid, should display an error dialog. |
| Event Action | List of all the apps associated with this Heroku account should be displayed. The following attributes are displayed   * App Name * Application URL * Git Url * App Status (Up, Idle, Down) |
| Post Event |  |

1. Manage my App

18

to

24

|  |  |
| --- | --- |
| 18 | View App Info |
| Description | Double clicking on the app would display application details. |
| Validation |  |
| Event Action | Open “App Info” in a new Window. Refer to section #7 |
| Post Event |  |

|  |  |
| --- | --- |
| 19 | Import |
| Description | To import an existing app on the Heroku account into the workspace |
| Validation | If the app is already imported into the workspace, this option should be disabled. |
| Event Action | Should do the same function as “import wizard”. The import process should be done as a background process. |
| Post Event | New Project with associated project files and nature should be created in the current workspace |

|  |  |
| --- | --- |
| 20 | Open |
| Description | Will open application URL in the embedded browser or the primary browser configured in Eclipse. |
| Validation |  |
| Event Action |  |
| Post Event |  |

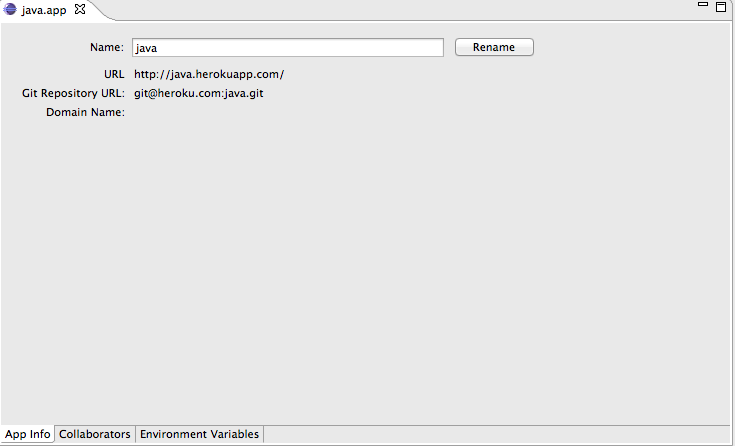
|  |  |
| --- | --- |
| 21 | Restart |
| Description | Will restart all processes associated with the app |
| Validation |  |
| Event Action | Will invoke a “restart” API call for the app that is currenly selected. |
| Post Event | Display updated status of the processes under the app. |

|  |  |
| --- | --- |
| 22 | View Logs |
| Description | To display application logs |
| Validation |  |
| Event Action | Will invoke the “logs” API call and fetch the logs for the application. Depending on the selection, the logs fetched should be for one of the following:   * The entire application (equivalent of `heroku logs –t`) * A specific process (equivalent of `heroku logs –p <processname>`) |
| Post Event | Logs should be displayed in the “Console” view that comes with Eclipse. |

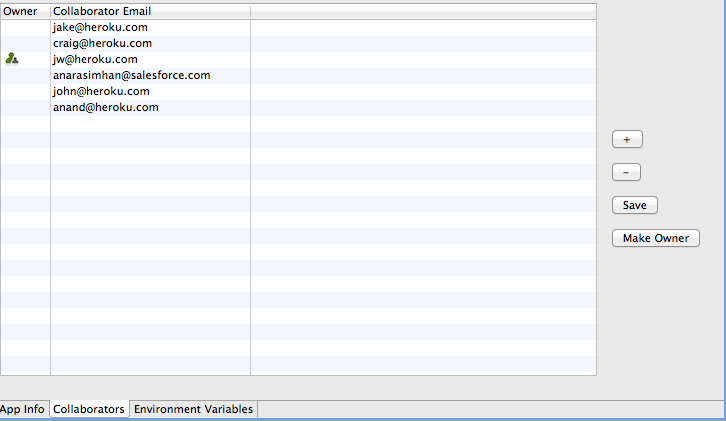
|  |  |
| --- | --- |
| 23 | Scale |
| Description | To scale up or down the app or processes |
| Validation |  |
| Event Action | Will invoke the “scale” API call and scale the application appropriately.  If the menu is clicked against a “process” under the application, a modal dialog box should open and the current # of processes should be displayed.  If the menu is clicked against the “application” and there is only 1 process type defined, the # of processes should be pre populated. |
| Post Event | Display status of the scale operation in the console logs. |

|  |  |
| --- | --- |
| 24 | Destroy |
| Description | To remove/destroy an app |
| Validation |  |
| Event Action | Should display a “Confirmation” dialog with the message “Destroy will remove your application add addons and data associated with the application permanently. Do you still want to destroy this app?”  If the developer clicks “Yes”, the “destroy” API call should be invoked against the app. |
| Post Event | The view should be refreshed to display the new list of apps |

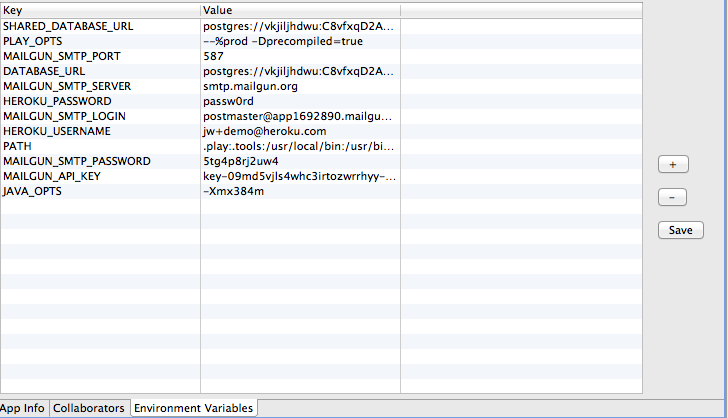
1. View App Info



1. Add Collaborators



1. Add/Remove Environment Variables



1. Error Handling

The plugin should catch any known “exceptions” and display a suitable error message. In case of any unknown exceptions, a “generic” error message along with the detailed stack trace should be displayed to the end-user. All exceptions should be logged to the workspace log for any troubleshooting.

When an error or exception occurs, the error should be written to Eclipse’s workspace log file so that developers can use the same for any trouble shooting that may be required.

**Note**: Exceptions such as “null pointer exceptions”, “array out of bounds”, “connection exceptions” should be gracefully handled and a suitable error message should be displayed to the end users.

1. Logging

The plugin should log any operations that are performed on the UI into the “console”. The API calls do not need to be logged. For e.g. if the developer is “importing” an app, the plugin should log:

Import [app-name] into Workspace.

Detected Java Maven application.

Adding Maven Nature to imported project

Heroku application import completed.

The exact format of these log messages will be detailed out as part of the design iteration.

1. Background Tasks

The plugin should leverage the Eclipse Background Tasks capability when possible. For e.g. “Scale”, “Import an App” etc. should be executed as background tasks so that the developer does not have to wait and can continue on with other tasks that they may perform.