Практическое занятие № 10

Планирование code-review

Цель: получение практических навыков подготовки среды совместной разработки для проведения инспекции программного года.

Норма времени: 2 часа.

Оборудование: Программное обеспечение, необходимое для работы: выбранные обучаемым средства просмотра видео; видеофайлы; доступ в Интернет.

Ход работы

Просмотр кода (англ. code review) или инспекция кода (англ. code inspection) – систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки. Целью просмотра является улучшение качества программного продукта и совершенствование навыков разработчика. В процессе инспекции кода могут быть найдены и устранены такие проблемы, как ошибки в форматировании строк, состояние гонки, утечка памяти и переполнение буфера, что улучшает безопасность программного продукта. Системы контроля версий дают возможность проведения совместной инспекции кода. Кроме того, существуют специальные инструментальные средства для совместной инспекции кода. Программное обеспечение для автоматизированной инспекции кода упрощает задачу просмотра больших кусков кода, систематически сканируя его на предмет обнаружения наиболее известных уязвимостей. В среде Github при работе с командами разработчиков программного обеспечения обычно при подготовке необходимо выполнить следующие действия:

- Добавить среду необходимых членов команды (организация и соавторы).
- Обеспечить возможность отправки версий и их слияния (Pull Requests).
- Обеспечить отслеживание ошибок (issues Github).
- Реализовать возможность комментариев к строкам и URL-запросов.

Как правило, существует два способа настройки в Github для совместной работы необходимых членов команды:

1. Создание организаций. Владелец организации может создавать множество организаций с разными уровнями доступа для различных репозиториев.

2. Добавление сотрудников. Владелец репозитория может добавлять коллабораторов с доступом Read + Write для одного репозитория

Создание организаций разработки. Если вы контролируете несколько команд и хотите установить разные уровни доступа для каждой команды с

различными членами и добавить каждого участника в разные репозитории, то организация будет наилучшим вариантом. Любая учетная запись пользователя Github уже может создавать бесплатные организации для репозиториев с открытым исходным кодом. Чтобы создать организацию, необходимо перейти на страницу настроек своей организации:

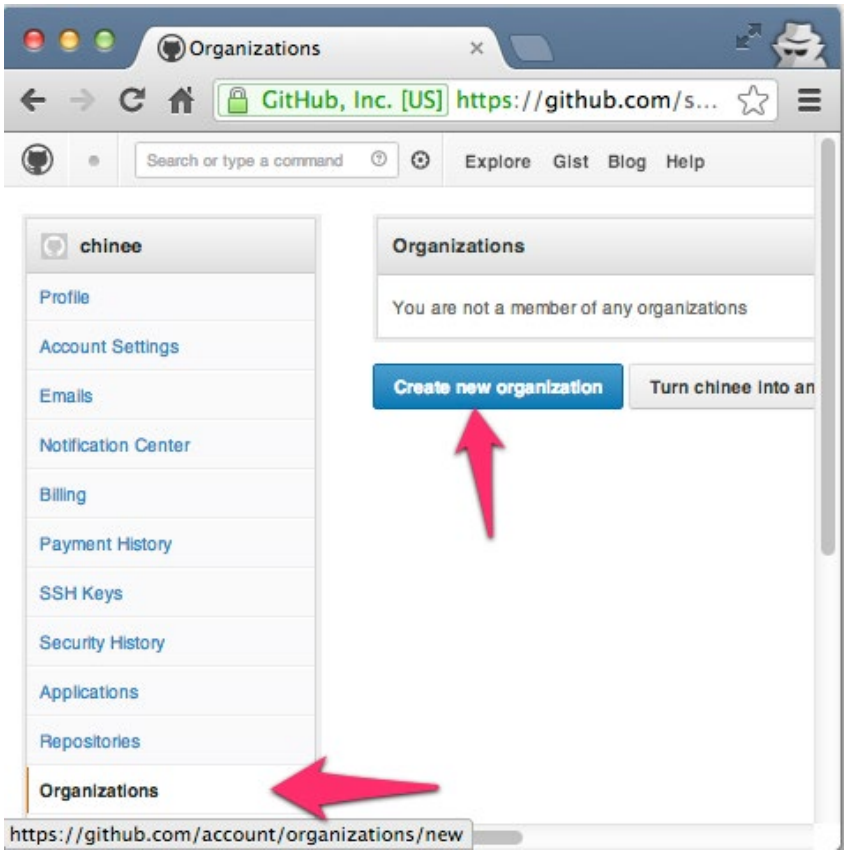

Рис.1 Добавление организации

Чтобы получить доступ к странице команд для вашей Органи-зации, необходимо перейти на http://github.com/organizations/[organization-name]/teams, или для создания новых организаций – https://github.com/organizations/[organization-name]/teams/new. Для создаваемых команд возможны три уровня доступа:

1. Pull Only: выборка и слияние с другим репозиторием или локальной копией. Доступ только для чтения.

2. Push and Pull: – доступно обновлением удаленного репозитория. Читайте + Запись.

3. Push, Pull & Administrative: (1), (2) добавляются права созданием команд, а также удаление аккаунтов организации. Чтение + запись + доступ администратора

4.

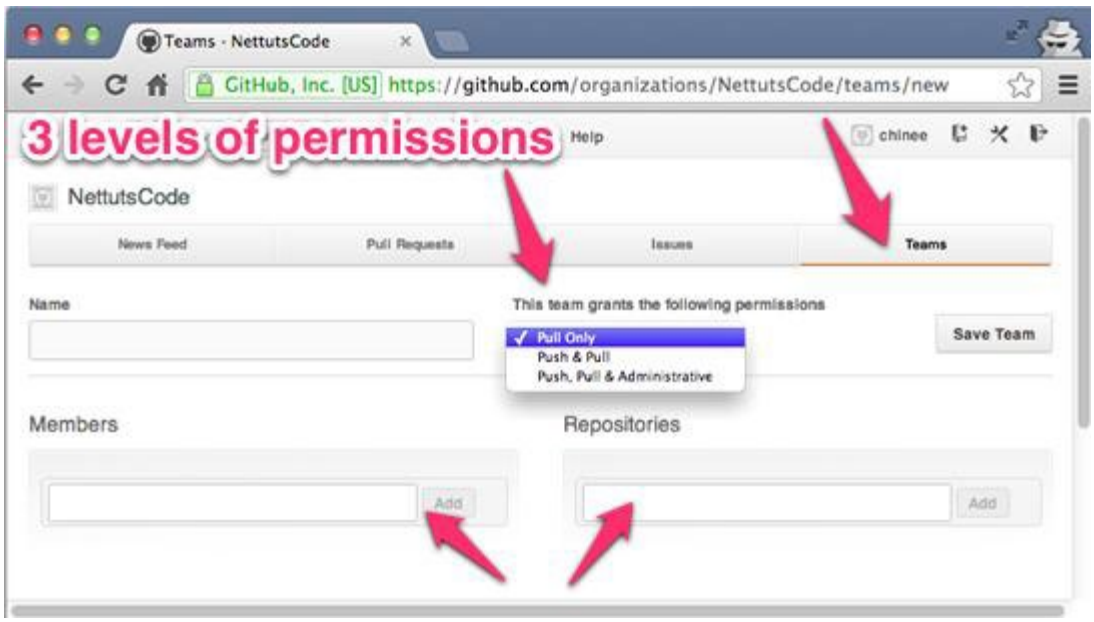

Рис. 2 Создание команд с различными уровнями доступа.

Настройка отправки и слияния (Pull Requests) Pull Requests – позволяет внести свой вклад в репозиторий, сделав его копию. Копия репозитария может быть отправлена (pull request) владельцу репозитория, чтобы объединить изменения кода. Сам pull request может включать обсуждение качества кода, функ-ций или даже общей стратегии. В Github есть две модели для создания копии репозитария. **Модель Fork & Pull** – используется в общедоступном репози-тории, на который у вас нет push-доступа

Share Repository Model – используется в частном репозито-рии, на который у нас есть push-доступ. В этом случае форк не требуется. Здесь мы видим рабочий процесс между двумя пользователями (repoowner и forked-repo-owner) для модели Fork and Pull: Определите репозиторий Github, в который вы хотите внести свой вклад, и нажмите кнопку «Fork», чтобы создать клон репозитория в вашей собственной учетной записи Github.

Рис.3 Вызов команды создания копии репозитария.

Это создаст точную копию репозитория в вашем собственном аккаунте.

Рис.4 Содержание действий в создании копии репозитария.

Выберите URL-адрес SSH, чтобы он запрашивал вашу парольную кодовую фразу SSH вместо имени пользователя и пароля каждый раз, когда вы делаете git push или git pull. Затем мы будем клонировать этот репозиторий на наш локальный компьютер:

```
1$ git clone [ssh-url] [folder-name]
2$ cd [folder-name]
```

Создадим новую ветку, чтобы внести очень простое изменение в файл readme.md:

```
1 $ git checkout -b [new-feature]
```

После внесения соответствующих дополнений для создания новых функций мы просто передадим новые изменения и проверку в ветку git master:

```
1$ git add
2$ git commit -m «information added in readme»
3$ git checkout master
```

Перейдем на ветку с новой задачей, а так же на псевдоним для удаленного репозитория. Затем мы будем пушить изменения с помощью git push [git-remote-alias] [branch-name]:

```
$ git branch
* master
readme
$ git remote -v origin git@github.com:[forked-repo-owner-username]/[repo-name].git
(fetch)
origin git@github.com:[forked-repo-owner-username]/[repo-name].git
(push)
$ git push origin readme
```

На исходной «развязанной» странице Github репозитория перейдем к ветке с новой функцией, а затем нажмите кнопку «Pull Request». После обсуждения возможно, что владелец «форкнутого» репозитория может

захотеть добавить изменения в новую функцию. В этом случае мы выберем одну и ту же ветку на нашей локальной машине, зафиксируем ее и запушим ее обратно на Github. Когда мы заходим на страницу запроса в оригинальном репозитории, он будет автоматически обновляться. Обзор кода С каждой фиксацией изменений Github позволяет использовать чистый интерфейс для общих комментариев или даже конкретных комментариев к отдельной строчке кода. Возможность делать комментарии или задавать вопросы по каждой отдельной строке ко-да очень полезна при проведении обзоров строка за строкой. Чтобы просмотреть встроенные комментарии, установите флажок в верхней части каждой фиксации.

Рис. 5 Просмотр встроенных комментариев

При обзоре кода могут быть использованы шаблоны URL-адресов, они предоставляют возможность отслеживать различия между фиксациями.

Шаблон сравнения веток – Compare branches / tags / SHA1 https://github.com/[username]/[repo-name]/compare/[starting-HA1]...[ending-SHA1].

Рис.6 Шаблон сравнения веток

Сравнение без пробелов: добавьте ?w=1 для сравнения URL-адресов.

Рис.7 Шаблон для сравнения адресов

Diff: добавьте .diff к URL-адресам сравнения, чтобы получить информацию о git diff в текстовом формате. Это может быть полезно для сценариев.

3. ПОРЯДОК ВЫПОЛНЕНИЯ И ЗАДАНИЕ ДЛЯ РАБОТЫ

1. Создайте репозиторий на сервисе Github.
2. Задайте несколько типов организаций с различными параметрами доступа.

3. Задайте несколько учетных записей для разработчиков.

4. Задайте необходимые параметры для отправки и слияния копии репозитория.

Контрольные вопросы:

1. Какие действия необходимо выполнить, чтобы подготовить среду разработки для инспекции кода?
2. Какие методы существуют для настройки Github для совместной работы нескольких разработчиков?
3. Что такое организация разработки в сервисе Github?
4. Какие уровни доступа могут быть заданы для организации в Github?
5. Какие параметры должны быть заданы при настройке отправки и слияния копии репозитория?
6. Какие модели применяются при создании копии репозитория?

Что такое «обзор кода»?