Темы [исследовательских] работ от компании JetBrains

Дмитрий [Юрьевич] Булычев

СПбАУ

20 сентября 2012

Санкт-Петербург

Гуманистический интерфейс для Git

Git (`http://git-scm.com`) — мощное средство контроля версий с непростым характером.

Цель: приручить Git.

Средство: графовое представление и GUI.

Примеры: GitX (`http://gitx.frim.nl`), SmartGit (`http://www.syntevo.com/smartgit/index.html`).

Задачи: Изучить Git на уровне эксперта; реализовать визуализацию репозитория в виде графовой структуры и основные нетривиальные операции (rebase/merge/reorder/join/split) в виде операций на этом графовом представлении.

Командная строка для IntelliJ IDEA

Командные строки входят в моду:

- Командная строка в FireFox (https://hacks.mozilla.org/2012/08/new-firefox-command-line-helps-you-develop-faster);
- GitHub Command Bar (https://github.com/blog/1264-introducing-the-command-bar);
- Наверное, что-то ещё.

Задача: Разработать плагин для управления IntelliJ IDEA с помощью командной строки.

Примеры (умозрительные):

- `font -family Arial -size 15`
- `${templateName} "MyClassMyInterface"`
- `move ${methodName} -above ${methodName}`
- `go ${file} ${method}`
- `debugger breakpoints -off`
- и т.д.

Паттерны эффективной генерации в JS

JS стал играть важную роль в народном хозяйстве.

Может быть рассмотрен как привлекательная целевая платформа, для которой стоит генерировать код.

JS — динамический, слаботипизированный => не живет без хитрых оптимизаций.

Разные JSVM делают эти оптимизации по-разному; одна и та же JSVM делает эти оптимизации по-разному в разных версиях.

Задачи:

- изучить способы оптимизации, применяемые разными JSVM (V8, Gecko и т.д.);
- определить паттерны эффективной генерации JS-кода;
- определить способ определения паттернов эффективной генерации JS-кода на будущее.

Регулярная аппроксимация строковых значений

Условия: дана императивная программа, в ней выделено выражение строкового типа.

Найти: регулярное выражение, которое аппроксимирует значение этого строкового выражения при выполнении этой программы на любых данных.

Способ: статический анализ различного ассортимента.

Зачем? Чтобы статически проверять ошибки в генерируемом управляющем коде (SQL-запросы, HTML-странички и пр.)

Синтаксические расширения для сопрограммности

Сопрограммы — это старинный способ взаимодействия кусочков кода.

Не особенно использовались по техническим причинам.

Могут пригодиться в определенных контекстах. Появляются в различных вариантах в современных языках программирования.

Задача: попробовать реализовать какие-то варианты сопрограммности в виде синтаксических расширений существующих языков.

Зачем? Чтобы иметь возможность непротиворечиво добавлять такие возможности в существующие языки.

Batches и Type Providers

Batch — специальная модель взаимодействия с удаленными источниками данных. Разделение локального и удаленно работающего кода на синтаксическом уровне с обеспечением семантического контроля.

Type providers — особенность F# 3.0 (появилась чуть больше года назад). Позволяет создавать новые типы данных для связи с внешним миром.

Задача: изучить то и другое; научиться их делать самому; определить, что нужно от языка, чтобы в него можно было вставить такое.

Зачем? Вдруг пригодится.

Спасибо за внимание!

- Вопросы?
- Предложения?
- Пожелания?

Адрес для связи: Dmitri.Boulytchev@jetbrains.com