YetAnotherClipper

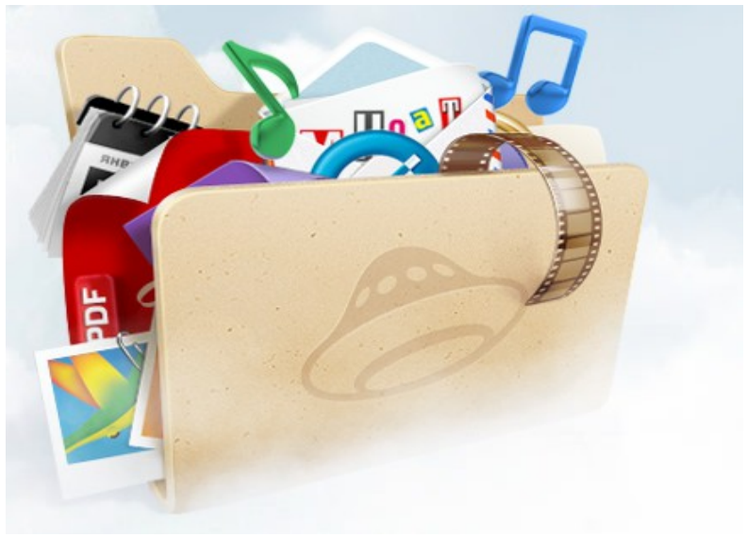
Толстопятов Всеволод
СПБГУ, 20.05.2014

Задача

В интернете можно найти много крутых статей/документов, порой объемных, которые хочется прочитать, но нету времени. Неплохо было бы хранить их где-нибудь в надежном месте, да еще и в удобоваримом виде

Постановка задачи

- Плагин к Хрому/Яндекс.Браузеру, который позволит пользователям сохранять документы на Яндекс.Диск
- Бэкэнд, который будет выкачивать документ, приводить его в нужный вид и сохранять куда надо
- Алгоритм, который будет выдирать из html только статью, избавляясь от мусора



Основные идеи

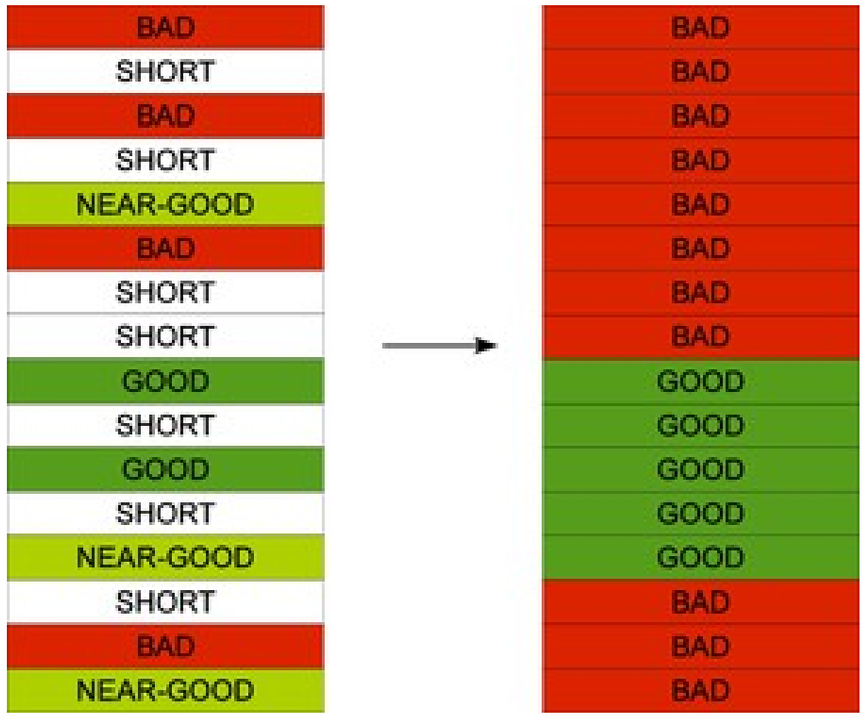
- Статья содержит текст и запятые
- HTML разбивает текст на блоки
- В статье плотность линков не очень большая
- Короткий блок со ссылкой — мусор
- Длинный блок с большим количеством ссылок скорее всего мусор
- Длинный блок с текстом и запятыми — скорее всего часть статьи (возможно комментарий)

Помехи

- Текст бывает не только в статье. Текст вообще много где бывает
- Статья может содержать в себе ссылки/картинки
- Комментарии нередко бывают большими (хабр) и ошибочно могут посчитаться статьей

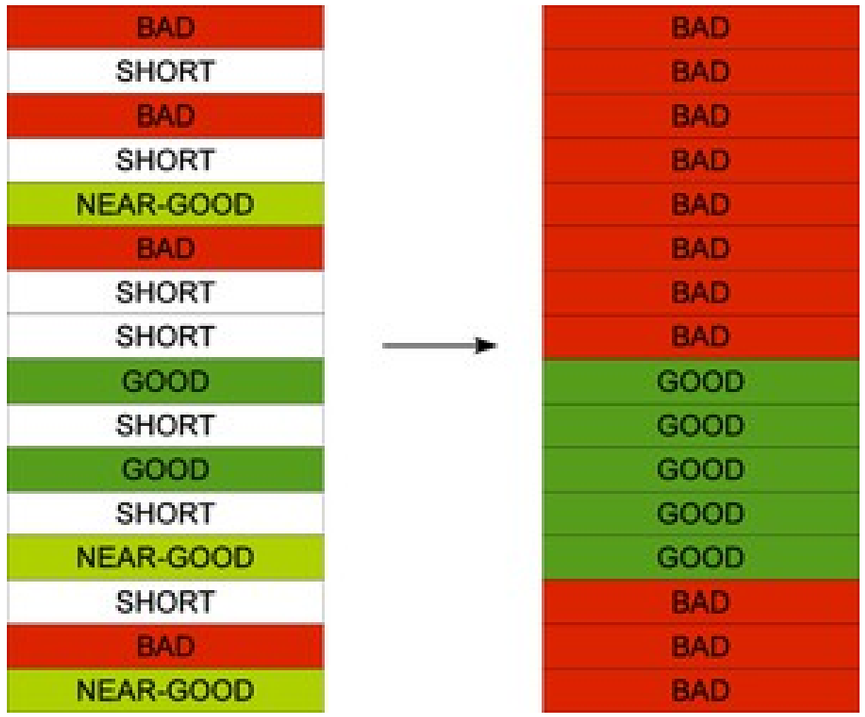
Главная идея

- Всё, включая html-тэги, любит сбиваться в кластеры



Главная идея

- Всё, включая html-тэги, любит сбиваться в кластеры



- Ссылка в статье — хорошо
- Ссылка в ссылках — плохо
- Высокая точность

Алгоритм. Подготовка

- Рассматриваем HTML как DOM-дерево
- Вводим несколько списков тэгов/атрибутов
  - Точно не статья: header, footer, .hidden, iframe etc.
  - Скорее всего не статья: comment, disqus, rss etc.
  - Скорее всего статья: article, body, content, entry etc.
- Вводим список того, что может содержать div, являющийся куском статьи: a, blockquote, img, ol, p, pre, etc.

Алгоритм. Первая стадия

- Начинаем обходить все узлы DOM-дерева
- Рассматриваем текущий узел
  - Если это div, содержащий «дозволенные» элементы, то помечаем его как параграф
  - Если это textNode (W3C), то помечаем как параграф
- На выходе получаем то же дерево, но некоторые узлы у него теперь помечены как возможные кандидаты на часть статьи

Алгоритм. Вторая стадия.

- Начинаем для каждого помеченного узла дерева считать некую метрику (эмпирическую)
- Посчитали метрику для текущего помеченного узла, прибавили её же к отцу, половину к деду, даже если они не помечены
- Получаем подмножество дерева с некоторым посчитанным весом для каждого узла
- Достаем оттуда узел с максимальным весом
- Утверждается, что это и есть статья в документе

Алгоритм. Метрика

- Задаем мап тэг -> вес
  - Div, p, pre — хорошо
  - H1-h6, form — плохо
- Смотрим на списки тэгов из первого шага
  - Если атрибут/тэг из «скорее всего кусок статьи», то увеличиваем вес
  - Если из «скорее всего нет», то уменьшаем
- Прибавляем вес за каждые 100 символов и запятые

Алгоритм. Тесты

- Хабр
- Лента
- The New York Times
- Блоги на blogspot
- Блоги на wordpress
- Ресурсы, достижимые из Яндекс.Новостей
- Википедия (тут не всё так гладко)

Алгоритм. Результат

Luxoft хабраиндекс 160,25

Профиль | Блог | Подписчики +6

21 мая 2013 в 10:34

Обзор java.util.concurrent.* tutorial

Блог компании Luxoft, JAVA*, Программирование*

В повседневной работе не так уж часто приходится сталкиваться с пакетом для многопоточности java.util.concurrent. Иногда существуют проектные ограничения по использованию java 1.4.2, где нет данного пакета, но чаще всего хватает обычной синхронизации и не требуется ничего сверхъестественного. К счастью, периодически возникают задачи, заставляющие немного пораскинуть мозгами и либо написать велосипед, либо порыться в javadoc'ах и найти что-то более подходящее. С велосипедом проблем нет — просто берешь и пишешь, благо ничего суперсложного в многопоточности нет. С другой стороны, меньше кода — меньше багов. Тем более, что на многопоточность никто в здравом уме юнит тестов не пишет, т.к. это уже полноценные интеграционные тесты получаются со всеми вытекающими последствиями.

Что выбрать для конкретного случая? В условиях запарки и deadline'ов довольно сложно охватить весь java.util.concurrent. Выбирается что то похожее и вперед! Так, постепенно, в коде появляются ArrayBlockingQueue, ConcurrentHashMap, AtomicInteger, Collections.synchronizedList(new LinkedList()) и другие интересности. Иногда правильно, иногда нет. В какой то момент времени начинаешь осознавать, что более 95% стандартных классов в java вообще не используются при разработке продукта. Коллекции, примитивы, перекладывание байтиков с одного места на другое, hibernate, spring или EJB, еще какая то библиотека и, вуаля, приложение готово.

Чтобы хоть как то упорядочить знания и облегчить вхождение в тему, ниже идет обзор классов для работы с многопоточностью. Пишу прежде всего как шпаргалку для себя. А если еще кому сгодится — вообще замечательно.

Для затравки

Сразу приведу пару интересных ссылок. Первая для тех, кто немного плавает в многопоточности. Вторая для «продвинутых» программеров — возможно тут найдётся что-нибудь полезное.

- Java Concurrency Tutorials
- Concurrent Programming in Java by Doug Lea

Немного об авторе пакета java.util.concurrent

Если кто хоть когда-нибудь открывал исходники классов java.util.concurrent, не могли не заметить в авторах Doug Lea (Даг Ли), профессора Oswego (Осуиго) университета штата Нью Йорк. В список наиболее известных его разработок попали java collections и util.concurrent, которые в том или ином виде отразились в существующих JDK. Также им была написана dlmalloc имплементация для

Информация о компании

671 подписчик

69 постов

Блог

Международный Agile консорци
Agile, Lean и все-все-все
15 мая 2014 в 13:39

CLRium: до конца регистрации
5 мая 2014 в 10:55

Мастер-класс по нутрям .Net Fr
30 апреля 2014 в 16:17

20-22 мая тренинг ICAgile Certif
Professional в Санкт-Петербург
28 апреля 2014 в 16:26

Получение указателя на объек
16 апреля 2014 в 16:36

DEV Labs 2014. ОНЛАЙН конфер
JAVA разработчиков
9 апреля 2014 в 13:49

Стратегии интеграции JavaFX
8 апреля 2014 в 10:31

Проблемы с кодом? Помогите ко
лучший код. Советы от Дино Э
11 марта 2014 в 15:11

Пятый вебинар технологическо
Luxoft: Канбан – альтернативн
успеху
7 марта 2014 в 12:39

Онлайн конференция для .NET

Алгоритм. Результат

Обзор java.util.concurrent.* / Блог компании Luxoft / Хабрахабр

В повседневной работе не так уж часто приходится сталкиваться с пакетом для многопоточности java.util.concurrent. Иногда существуют проектные ограничения по использованию java 1.4.2, где нет данного пакета, но чаще всего хватает обычной синхронизации и не требуется ничего сверхъестественного. К счастью, периодически возникают задачи, заставляющие немного пораскинуть мозгами и либо написать велосипед, либо порыться в javadoc'ах и найти что-то более подходящее. С велосипедом проблем нет — просто берешь и пишешь, благо ничего суперсложного в многопоточности нет. С другой стороны, меньше кода — меньше багов. Тем более, что на многопоточность никто в здравом уме юнит тестов не пишет, т.к. это уже полноценные интеграционные тесты получаются со всеми вытекающими последствиями.

Что выбрать для конкретного случая? В условиях запарки и deadline'ов довольно сложно охватить весь java.util.concurrent. Выбирается что то похожее и вперед! Так, постепенно, в коде появляются ArrayBlockingQueue, ConcurrentHashMap, AtomicInteger, Collections.synchronizedList(new LinkedList()) и другие интересности. Иногда правильно, иногда нет. В какой то момент времени начинаешь осознавать, что более 95% стандартных классов в java вообще не используются при разработке продукта. Коллекции, примитивы, перекладывание байтиков с одного места на другое, hibernate, spring или EJB, еще какая то библиотека и, вуаля, приложение готово.

Чтобы хоть как то упорядочить знания и облегчить вхождение в тему, ниже идет обзор классов для работы с многопоточностью. Пишу прежде всего как шпаргалку для себя. А если еще кому сгодится — вообще замечательно.

Для затравки

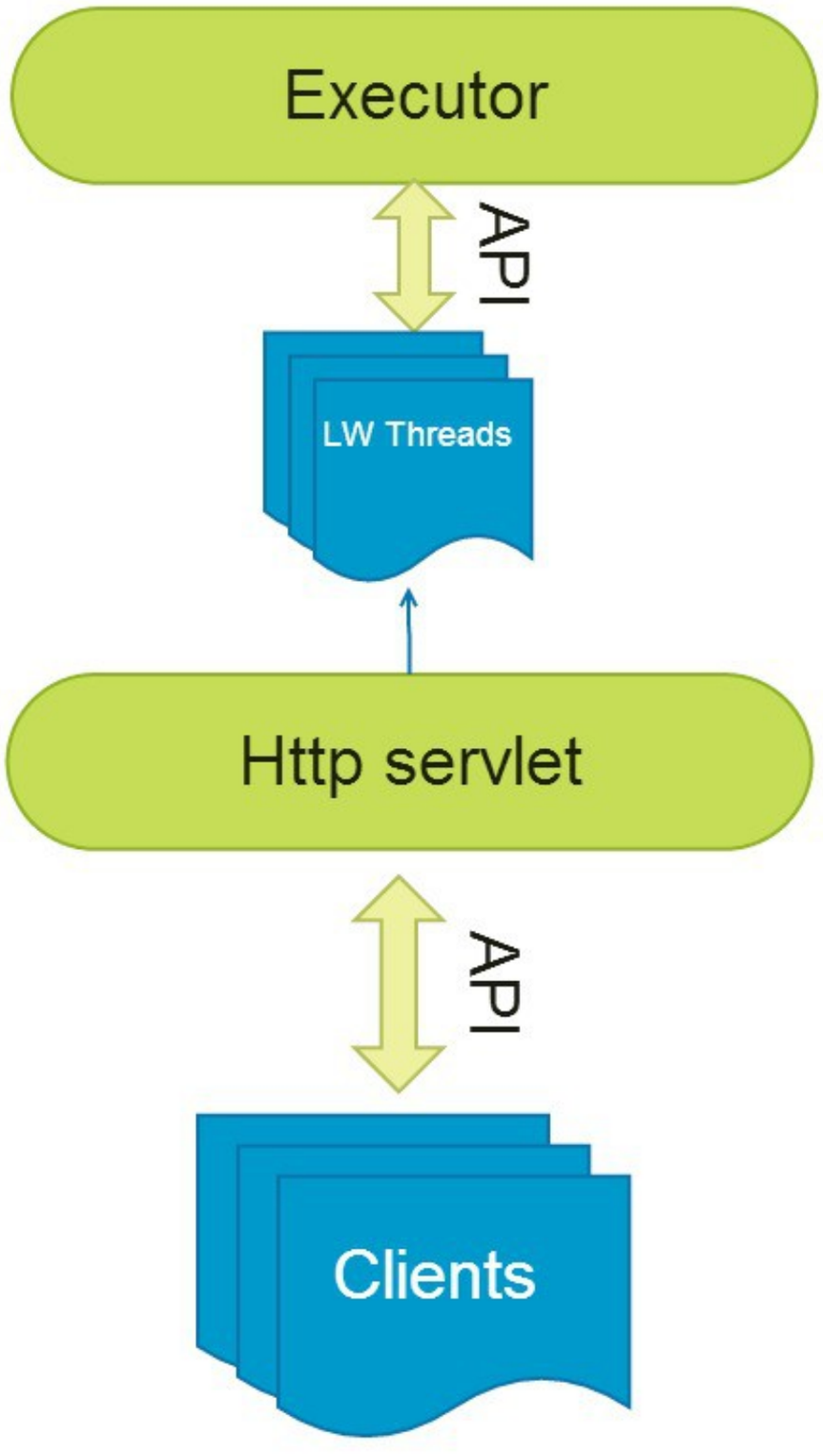
Архитектура

- Задача — не только придумать/написать алгоритм, но и сделать полноценный продукт со всеми вытекающими
- Плагин к браузеру: javascript, минимум логики:
  - Авторизация через OAuth, сохранение токена
  - Проверка валидности токена
  - Отправка запроса бэкэнду
- Сервис:
  - Java, любые сторонние библиотеки
- Требования к сервису:

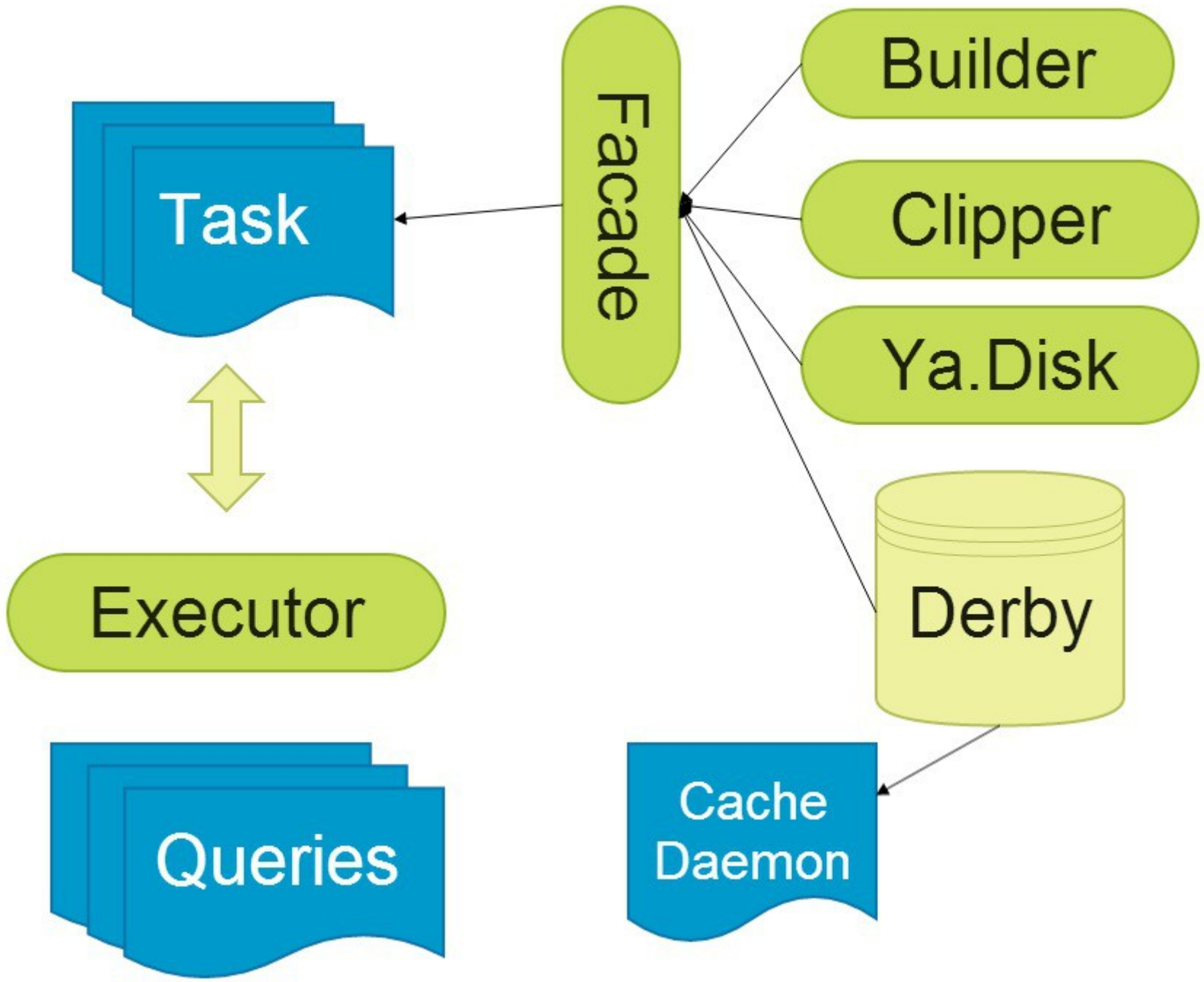
  /* Сказать словами */

Архитектура. Сервис.

- Jetty в роли асинхронного http-сервера
- DI-контейнер — Spring
- Log4j логгирует все более-менее осмысленные действия
- Jsoup для парсинга HTML
- Derby, потому что im memory
- Maven как средство сборки
- JUnit



Архитектура. Executor



Итог

- Полноценный сервис, готовый для использования
- Опыт промышленной разработки
- Крутой алгоритм
- Куча технологий