GROCON

GRODNO 2017

ВЯЧЕСЛАВ
ТРЕТЬЯК

КАК ЛОКАЛИЗОВАТЬ ПРОЕКТ и не сойти с ума

Я

- **2006** — начало
- **2009** — локализованное приложение под Android
- **2014** — локализовал веб-приложение ≈ 500 файлов (≈ 100 000 строк)
- Думал, что всё знаю и умею про локализацию
- **2017** — …

The Проект

- ≈ 3500 файлов
- ≈ 500 000 строк
- PHP, JavaScript, CoffeeScript, Twig, Pug, Html
- ZF2, Angular (1.x)
- Очень много нелокализованного текста

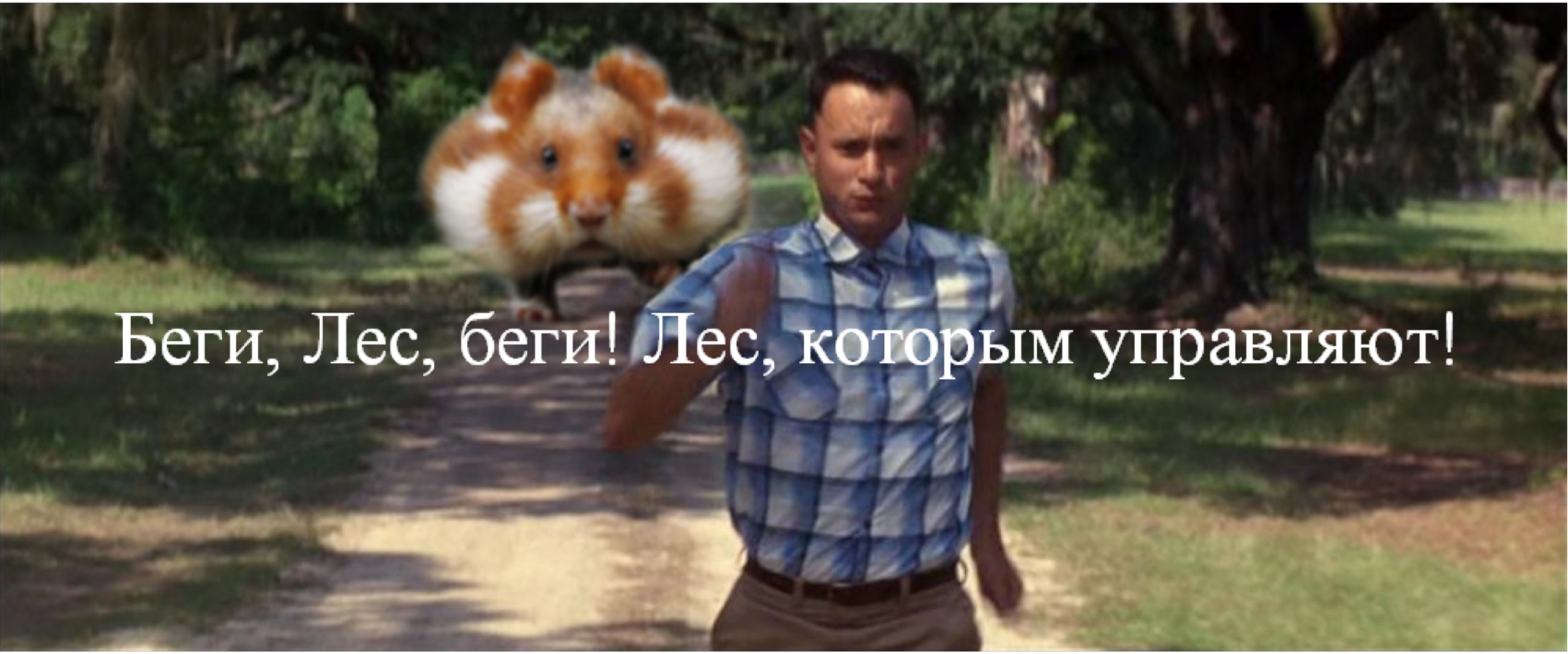
Всё как у всех

```
"Привет "
    + (user.isGuest() ? "Гость" : user.getName())
    + "!, онлайн "
    + onlineCount
    + " "
    + pluralize(onlineCount,
      "пользователь",
      "пользователя",
      "пользователей")
```

1–2–3

1. Моё видение хорошей интернационализации *(best practices)*
2. Что всё-таки делать, если уже есть нелокализабельный проект *(наш опыт)*
3. Резюме

МОЁ ВИДЕНИЕ ХОРОШЕЙ ИНТЕРНАЦИОНАЛИЗАЦИИ


Беги, Лес, беги! Лес, которым управляют!

Локализация vs Интернационализация

- Локализация — **перевод** проекта на другой язык
- Интернационализация — **написание кода**, который позволяет легко делать локализацию

Наивный подход

```
hi: "Привет"
guest: "Гость"
online: "онлайн"
users: one="пользователь", two="пользователя",
       many="пользователей"

t("hi")
    + " " + (user.isGuest()
        ? t("guest") : user.getName())
    + "!, " + t("online")
    + " " + onlineCount + p("users", online)
```

Не работает

- Одно и то же слово в разном контексте переводится по–разному:
  - Класс — как класс в школе (grade)
  - Класс — как поощрение пользователя (cool)
  - Класс — как css-класс (class)
- Может потребоваться поменять части предложения местами:
  - “Привет, Саша!”
  - “Alex, welcome to the system!”

Подстановка строк

```
hello_user: "Привет {username}!, онлайн {onlineCount}
{onlineStr}"
guest: "Гость"
user_plural: one="пользователь", two="пользователя",
             many="пользователей”

t("hello_user", {
    username: user.isGuest()
        ? t("guest") : user.getName(),
    onlineCount: onlineCount,
    onlineStr: p("user_plural", onlineCount)
})
```

Не совсем работает

- Не надо использовать повторно строки для разных экранов
- В разном контексте одинаковое по смыслу слово может переводиться по–разному

  (Yes: Да / Согласен, Удалить: Delete / Remove)
- На разных экранах одна и та же фраза на одном из языков может не влезть

Именуем ключи правильно

```
dashboard_welcome: "Привет {username}!, онлайн
{onlineCount} {onlineStr}"

dashboard_guest: "Гость"

dashboard_user_plural: one="пользователь",
                       two="пользователя",
                       many="пользователей"
```

Почти хорошо, но…

- “На сайте было 2 пользователя в течение последних 15 дней” — неудобно разделять **plurals** с самими строками
- “Возьмите карточку **{color}** цвета со стола”, где color это “красного”, “синего” или “желтого” — одно сообщение разделяется на несколько, переводчику не удобно
- **Plurals** + подстановки + **select**–подстановки в одном предложении — всё очень грустно 😔

ICU MessageFormat

```
dashboard_welcome: "Привет {hasUsername, select, yes
{{username}} other {Гость}}! онлайн {onlineCount,
plural, one {# пользователь} two {# пользователя}
many {# пользователей} other {# пользователя}}"

t("dashboard_welcome", {
    hasUsername: !user.isGuest(),
    username: user.getName(),
    onlineCount: onlineCount
})
```

CSS–классы — прокидываем

Нет:

```
'Hello <span class="cabinet-payments-multi-discount-popup__red-text">{username}</span>!'
```

Да:

```
'Hello <span class="{cls}">{username}</span>!'
```

Ссылки — по возможности прокидываем

```
"Нажмите {link}здесь{_link} для входа в систему."
```

Профит — может быть реальная ссылка,
а может и **span** с **onclick**

Работает, но…

- **Сообщение удалили из кода, но не из ресурсов**
- Удалили сообщение из ресурсов, но не из кода
- Добавили ключ сообщения в код, но в ресурсы — забыли
- В коде и в ресурсах сообщение есть, но в ключе описка
- Html–сообщение вставлено как текст — не “**привет**” болдом, а “`<strong>привет</strong>`”
- Фигак–фигак и в продакшен — в код захардкодили строку (vs добавили через ресурсы)

Статический анализ

- **В коде нет захардкоданных строк**
- Нет дубликатов ключей
- **Ключ используется в коде**
- В коде нет ключей, которые отсутствуют в ресурсах
- Строки правильно подставляются
  (html как html, текст как текст)
- Каждый конкретный ключ есть во всех языках

И ещё

- В большинстве систем локализации для каждого языка есть файл **key → value**
- Добавляешь строку в один язык — надо не забыть добавить во все остальные
- Идеальный проект — после программистов должны работать переводчики
- У кого есть этот идеальный проект? 😃

И ещё

```
'Application_Promo_EmailPlaceholder' => [
    'ru' => 'Введите email',
    'en' => 'Please, enter email',
],
```

- Программистам — удобней
- Для всего остального есть скрипты

ЧТО ВСЁ–ТАКИ ДЕЛАТЬ, ЕСЛИ УЖЕ ЕСТЬ НЕЛОКАЛИЗАБЕЛЬНЫЙ ПРОЕКТ


Translate!
Insert
Home
Delete
End
Shift

Наивный подход

Открывать по очереди каждый файл, и выносить строки в ресурсы

+ Достаточно раздеваться и работать

– Over 100500 часов

Если есть много денег

Нанять аутсорсеров

\+ Пусть аутсорсеры вкалывают, а мы будем отдыхать на Бали 😀

\- Нет гарантии нормального качества, надо тратить деньги

Реалистичный подход

Попытаться по максимуму автоматизировать процесс

\+ Значительно уменьшается доля ручного труда

\- Надо писать тулы, и всё равно, некая доля ручного труда остаётся

Почему не xgettext?

- Не умеет Pug и Html (а также CoffeeScript)
- Не понимает какие сообщеньки — это сообщеньки, а какие — служебные строки

Процесс эволюции

№1

- Проект на русском — любая строка, в которой есть русские символы подлежит локализации
- Скрипт регэкспами проходит по файлам, и пишет в консоль где и что не переведено
- Руками это всё заносится в ресурсы
- **Проблемы** — не всегда правильно выдирает сообщеньки (ну… regexp)

№2

- Скрипт различает типы файлов, и для PHP парсит через встроенный **`token_get_all()`**
- **Профит** — для PHP сообщеньки выдираются идеально (кроме конкатенации нескольких строк)
- **Проблемы** — лень руками вносить сообщения, тем более что часть ключа (префикс **Модуль_Компонент_**) поддаётся автогенерации

№3

- Автогенерация части ключа и режим автодополнения ресурсов выдираемыми сообщениями
- **Профит** — запустил, получил почти готовые локализационные файлы, остаётся только добавить последнюю часть ключа
- **Проблемы** — менять в исходниках текст на ключи надо руками, кажется, что это можно автоматизировать

№4

- Автозамена на регэкспах (для PHP)
- **Профит** — запустил генератор, подправил ключи, запустил автозамену, готово
- **Проблемы** — надоело дополнять ключи

```
...->add((new Element\Text('title'))
   ->setLabel('Название')) // ключ будет Title


$field = new Element\Text('address');
$field->setLabel('Адрес'); // ключ будет Address
```

№5

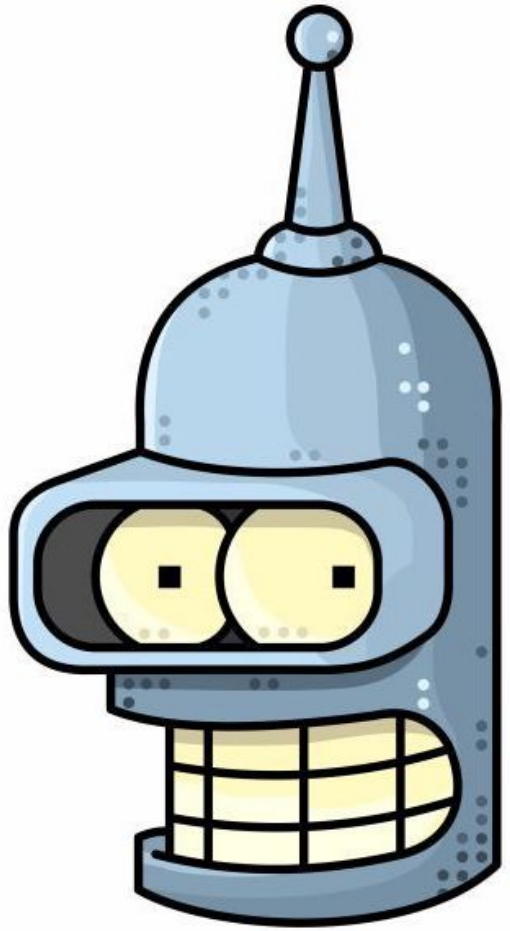
- Анализатор контекста (на основе конечных автоматов), в 70% угадывает название ключа
- **Профит** — для PHP-файлов интернационализация вообще сказка
- **Проблемы** — в pug–файлах надо пропускать комментарии, а на регэкспах это не очень хорошо делать; так же, когда есть много текста разбавленного html–ом, то регэкспы отрабатывают плохо

№6

- Подключили pug–парсер
- **Профит** — генерация — сказка
- **Проблемы** — PHP–файлы закончились, а для других типов нет автогеренатора ключа и автопатчера

№7

- На самом деле нет
- Можно подключить парсеры для остальных типов файлов, а для автогенерации ключей можно подключить Google Translate API
- Но это занимает время и стоит денег
- Лучше включить на пару недель режим биоробота, и всё допилить руками



Дальнейшая эволюция

- Переводчикам платят за знаки, а в проекте есть места, где строки можно объединить (например, кнопки “Сохранить” в админке)
- Скрипт для поиска и замены дубликатов
- **Профит** — избавляемся от дубликатов (**только от тех, от которых стоит избавиться**), экономим деньги на перевод (до дедупликации было ≈ 7000 сообщений, после — ≈ 5000)

Что 100% можно было бы сделать лучше

- Автозамена для PHP не на основе регэкспов, а через parse / re-print — меньше косяков
- Парсеры для Twig, JS и HTML — точное выдирание, автозамена, возможность автогенерации ключей
- Translation memory для автогенератора ключей
- Ещё более умная выдиралка, которая бы понимала конкатенацию
- Поправить CSS, чтоб в сообщениях не использовать классы, а только простые тэги

Что, вероятно, можно было бы сделать лучше

- Пусть ключом будет часть сообщения латиницей — “Введите имя” → “**`VvediteImja`**“
- Некрасиво, но можно генерировать ключи полностью автоматически
- Или всё–таки раскошелиться на Google Translate API

РЕЗЮМЕ

Резюме

- Проект с нуля — сразу интернационализацию!
  - Даже если кажется, что она никогда в жизни не понадобится
- Проект достался в наследство? Сначала определите объем работ
  - Это просто: **`find`** / **`cat`** / **`wc -l`**, подсчитайте количество файлов и количество строк

Проект достался в наследство?

- Небольшой — можно обойтись тулой, которая просто показывает в каких файлах есть нелокализованные строки
- Среднего размера — тула для поиска нелокализованных строк + пара джунов
- Большой — придётся автоматизировать процесс

Автоматизация

Самое важное — определение локализабельности строки

- Если проект с нелатинским языком, то всё просто
- Если латиница, то считать локализабельными строки, в которых есть пробелы и знаки пунктуации
- Может быть, определять из контекста

Автоматизация

- Узнайте какие типы файлов есть в проекте, сколько их, нужна ли им автоматическая обработка
- Продумайте механизм генерации ключей:
  - Латинизация сообщения
  - Перевод через Google Translate API
  - Полуручной режим

Автоматизация

- Парсилка под конкретный тип файла **vs** простой код на regexp–ах
- **Парсилка** — точное выдирание строк, автозамена строк на ключи, можно анализировать контекст (для “угадывания” ключа, типа сообщения или ещё чего-нибудь)
- **Регэкспы** — скрипт в десяток строчек и подходит для любых типов файлов

Автоматизация

- Запаситесь терпением — как бы вы не автоматизировали, всё равно будет доля нудного и монотонного ручного труда
- И не забудьте написать скрипты для статического анализа!

А теперь надо локализовать контент… 😃

Полезные ссылки

- **Пара слов об интернационализации приложений**
  https://habrahabr.ru/post/165705/

- **Android string.xml — несколько вещей, которые стоит помнить**
  https://habrahabr.ru/post/307798/

- **Formatting messages**
  http://userguide.icu-project.org/formatparse/messages

http://eightsines.com/me.html#talks