# Литературное программирование в Clojure с помощью Emacs/org/babel/cider

Maxim Bazhenov

1 февраля 2016 г.

## Содержание



## 1 Совместимые версии org-mode/babel, cider/lein nrepl middleware

Не всегда cider работает правильно с org/babel, не релиз версии cider часто не совместимы с текущей версией org/babel, ниже даны версии ПО которые работают на моей машине

Работающие версии ПО:

- org-mode/babel 20160125
- cider/lein nrepl middleware 0.10.0

# 2 Настройка

## 2.1 org-mode/babel

### 2.1.1 Базовые настройки

Следующие настройки, на мой взгляд, делают работу более удобной, их надо поместить в ~/.emacs

- Размер отступа внутри блока с кодом

  ```
  (setq-default org-edit-src-content-indentation 2)
  ```

- TAB внутри блока кода будет работать также как если бы он работал в основном режиме для языка программирования блока

  ```
  (setq-default org-src-tab-acts-natively t)
  ```

- Раскрашивать код внутри блока кода

  ```
  (setq-default org-src-fontify-natively t)
  ```

- Не спрашивать каждый подтверждения при выполнении блока кода

  ```
  (setq-default org-confirm-babel-evaluate nil)
  ```

### 2.1.2 Поддержка Clojure

Загружаем поддержку Clojure для babel и настраиваем его использовать cider для исполнения кода из блоков кода Clojure.

```
(require 'ob-clojure)
(setq-default org-babel-clojure-backend 'cider)
```

В принципе, установка org-babel-clojure-backend не обязательна если cider подключен раньше org-mode в ~/.emacs

### 2.1.3 Экспорт (для ubuntu/texlive)

Настройка экспорта в babel гораздо более комплексная. Экспорт проходит в 2 этапа, сначала в *.tex файл, а потом из *.tex в *.pdf с помощью pdflatex. Основные проблемы возникают на втором этапе, часто pdflatex не может обработать LaTeX-файл полученный на первом этапе, так как он содержит ошибки - не подключены какие-то LaTeX-пакеты, неправильно указана кодировка, и т.п. Во время экспорта в дирректории с *.org файлом создаётся *.log файл с журналом процесса экспорта. Если экспорт не работает, то разбираться надо начиная с файла журнала, ищем сообщения об ошибках или предупреждения, и исправляем одно за одним ;). Если экспорт проходит успешно, то файл журнала удаляется автоматически.

Ниже приведён более или менее рабочий вариант с приемлемой поддержкой русского языка и расцветкой блоков кода.

Без пакета ox-latex переменные org-latex* недоступны, так что запрашиваем пакет

```
(require 'ox-latex)
```

Для поддержки русского сначала необходимо настроить список соответствий входных кодировок Emacs и LaTeX, так чтобы LaTeX-файл содержал инструкцию `\usepackage[utf8x]{inputenc}`.

```
(add-to-list 'org-latex-inputenc-alist '("utf8" . "utf8x"))
```

Также выходной LaTeX-файл должен включать пакет cmap и babel (с опциями english,russian). Мы настраиваем список org-latex-packages-alist, который используется для указания пакетов включаемых в каждый LaTeX файл. Формат элемента списка ("optionspackage"SNIPPET-FLAG), флаг нужен чтобы указать что пакет также необходим для преобразования LaTeX-фрагментов (snippet) в изображения для включения в не LaTeX-файлы, в web-страницы, например. Я не очень понимаю какие LaTeX-пакеты для этого нужны, поэтому всегда указываю `t`.

```
(add-to-list 'org-latex-packages-alist '("" "cmap" t))
(add-to-list 'org-latex-packages-alist '("english,russian" "babel" t))
```

Дальше идёт настройка подсветки синтаксиса при экспорте. org-latex поддерживает два способа:

- с использованием LaTeX-пакета listings (требует также пакет color)

- с использованием LaTeX-пакета minted (требует внешней утилиты python pygments)

Поддержка Clojure в пакете listings рудиментарная, однако не требует установки дополнительных программ кроме LaTeX и соответствующих пакетов. Метод minted даёт более красочные результаты, однако требует установки внешней утилиты python pygments.

Настройка listings

```
(setq-default org-latex-listings t)
(add-to-list 'org-latex-packages-alist '("" "listings" t))
(add-to-list 'org-latex-packages-alist '("" "color" t))
```

Настройка minted

```
(setq-default org-latex-listings 'minted)
(add-to-list 'org-latex-packages-alist '("" "minted"))
(setq-default org-latex-pdf-process '("pdflatex -shell-escape -interaction nonstopmode
```

Так как LaTeX-пакет minted использует стороннюю программу python pygments для формирования HTML-кода с расцветкой, то LaTeX-файл будет включать инструкции `\write18{...}`, позволяющие запускать внешние программы из LaTeX-файла (при конвертации в PDF), это считается небезопасным и поддрежка `\write18` в pdflatex и аналогах по умолчанию отключена. Чтобы её включить надо передать pdflatex ключ -shell-escape. Единственный способ который я нашел это полное переопределение переменной org-latex-pdf-process, я просто взял значение по умолчанию и добавил везде ключ -shell-escape.

В документации к `org-latex-listings` указано, что при использовании метода `minted`, LaTeX-пакет `minted` надо добавлять с опцией `newfloat`

```
(add-to-list 'org-latex-packages-alist '("newfloat" "minted"))
```

однако у меня так не заработало, при конвертации из LaTeX в PDF происходила ошибка, в файле журнала экпорта была указана ошибка что пакет `newfloat` не найден. Всё заработало после того как я убрал опцию `newfloat`. Чем грозит удаление этой опции, мне пока не ясно.

По умолчанию ссылки в PDF-документе подсвечиваются синим цветом, мне так не нравится, я предпочитаю чёрный. Кроме того, поддержка русского языка, а точнее unicode кодировки, в LaTex-пакете `hyperref`, который используется для создания гиперссылок и содержания PDF-документа, отключена по умолчанию. Её можно включить, правильно

настроив генерируемый LaTeX-заголовок, для этого он должне содержать инструкцию `\usepackage[unicode]{hyperref}`, однако инструкция `\usepackage[]{hyperref}` (без опции `unicode`) включается в заголовок генерируемого LaTeX-файла по умолчанию. Она входит в список пакетов по умолчанию в переменой `org-latex-default-packages-alist`, изменять которую не рекомендуется. Другой способ настроить кодировку в LaTeX-пакете `hyperref` это использовать инструкцию `\hypersetup{...}`, шаблон которой, содержится в переменной `org-latex-hyperref-template`, вот её мы и используем для настройки кодировки и цвета ссылок.

```
(setq-default org-latex-hyperref-template "\\hypersetup{\n unicode=true,\n pdfauthor={
```

### 2.2 Cider

- Подключаем Cider и указываем ему скрывать служебные буферы с списке буферов emacs (так просто чтоб не засорять список)

```
(require 'cider)
(setq-default nrepl-hide-special-buffers t)
```

### 2.3 Leiningen

Для правильной работы Cider необходимо указать Leiningen загружать cider nrepl middleware при создании REPL. Для этого необходимо соответствующим образом настроить профиль repl. Открываем или создаём ~/.lein/profiles.clj и добавляем в него следующие настройки:

```
{:repl {:dependencies [[org.clojure/tools.nrepl "0.2.12"]]
        :plugins [[cider/cider-nrepl "0.10.0"]]}}
```

Разумеется, версии tools.nrepl и cider/cider-nrepl могут меняться со временем. Не уверен что tools.nrepl нужен.

thi-ng/babel

## 3 Дополнительные нюансы

При составлении данного документа я наткнулся на ошибки экспорта, несмотря на все эти настройки конвертация в PDF завершалась ошибкой. Причина была в том что мой текст содержит LaTeX-инструкции, которые в нём даны просто для описания, а не для исполнения. Такие инструкции должны быть вставлены в текст PDF-документа как есть, без интерпритации. Есть по крайней мере 3 способа как это сделать:

- Выделять такие инструкции символами ~ или =, например `~\some-tex-instruction~`
- Использовать org-mode опции для emacs-буфера: `OPTIONS: tex:verbatim`
- Установить в `/.emacs` переменную: `(setq-default org-export-with-latex "verbatim")}`