Emacs Flashcards

Leon Rische

[2019-08-12 Mon 13:38]

Contents

1 Card Filters

any match front, back and answer

front match front

back match back

answer match answer

learning cards in the learning phase

deck cards by deck id

due cards due right now

leech leech cards

The query `cards?deck=127&front=test` matches all cards in deck 127 with a front side containing the text "test".

2 Leeches

Cards that have been rated as "again" more than eight times and have an ease < 2.0 are considered as **leeches**.

The ease is included in this classification so that cards can stop being leeches once their review performance has increased.

Leeches should be rewritten or split up into smaller cards to improve the review performance.

3 Design Goals

1. Open Source, works on Linux / BSD
2. Org mode syntax for cards
3. Easy integration of cards in existing org mode documents (incremental reading)
4. Card management & review from inside emacs
5. Independent spacing / ease calculation for each node of a cloze card

None of the existing solutions I've seen so far fits these.

org-drill is the one that comes closest and is a elegant solution in that it is fully contained within emacs.

Because ease levels and intervals are stored in the properties of a card, for cloze deletions, a random node is shown at a time and depending on the review result of this node, the interval and ease of the "parent" card are updated.

I'd like to have independent intervals and ease levels for **each node** of a cloze deletion, because some nodes might be harder to remember than others.

With shared intervals / eases, hard nodes are repeated to rarely and easy nodes are repeated to often.

In addition to that, searching a bunch of org files for cards that are due does not scale well, querying a database is much faster.

Adding a database into the setup can lead to problems where the state of the card in the file is different from the state of the card in the database, but I think the increased flexibility and performance is worth this trade-of.

4 Simple Cards

Simple cards come in two types, `normal` and `text_input`, the latter of which is useful for learning the spelling of words.

5 Double Cards

For each card of type `double`, two cards are created, one mapping from front to back, one mapping from back to front.

This is useful for learning words of a foreign language.

6 Cloze Cards

Cloze cards have only a front side.

Nodes are marked with either `{{text}}` or `{{text}{hint}}`. This syntax is inspired by the org mode link syntax.

If either `text` or `hint` contains `{, }`, the alternative syntax `«text»<hint»` can be used. This is useful for deletions containing LaTeX formulas.

There are two kinds of cloze cards, **deletions** and **enumerations**.

A card is created for each of the nodes, replacing `{{text}}` with `[...]` and `{{text}{hint}}` with `[hint...]`. The back of this card is the `text` of the node.

For deletions, one of the nodes is hidden at a time, for enumerations, one node and all after it are hidden.

For enumerations, nodes other than the current one are shown as `...` or `...hint...`.

Enumerations are well suited for learning lists of facts where the order is important, e.g. when learning the digits of π.

Combined with inline flashcards, this type of card can be used for **incremental reading**, wrapping paragraphs of text in inline flashcards and marking relevant pieces of information as nodes.

7 Inline flashcards

```
#+BEGIN_FLASHCARD :deck_id 164 :type deletion :id 139
The capital of {{France}{country}} is {{Paris}{city}}.
#+END_FLASHCARD
```

For the card types "normal" and "double", a horizontal line (`---`) can be used to split the front from the back.