The **pmdb** Package

D. P. Story
Email: `dpstory@acrotex.net`

processed June 7, 2021

Contents

```
⟨*package⟩
```

1 Introduction

This package addresses the issue of a poor-man's database. Educators who use LaTeX to construct exams and homework sometimes have a collection of problems. Each problem is in its own TEX file. The educator creates a document and `\inputs` several of these packaged questions. This package attempts to provide some "user interface" to the questions and provides a mechanism of selecting which questions are to be included in the document.

What does this package do? For a document that inputs content using the LaTeX command `\input`, the same content can be input using the command `\pmInput` (capitalized). When content is input by `\pmInput`, a check box is created in the margin at the insertion point of the content. The check boxes so created can be checked or cleared. When the user clicks on push button provided by this package, a list of all *selected* `\input` statements are listed in the JavaScript console. This list can then be copied and pasted into another document the author is developing. If you Ctrl+Click on a check box, the associated content is opened in the default TEX editor. In this way, the document author can see a typeset of the content and decide whether the content should be included in the developing document.

The DB stage: When you have a collection of questions (or content) in various files and want to use this package to `\input` them into your document, the following comments are apropos:

PDF creators: Any PDF creator current in the LaTeX world is valid for use with this package.

PDF viewers: The ideal viewer is Acrobat, however, Adobe Reader and PDF-XChange Editor can also be used. In the case of Adobe Reader, there is an annoying security dialog box that appears each time you use the Ctrl+Click feature of the check box; the Ctrl+Click feature does not WORK with PDF-XChange Editor.

The production stage: After the document has been assembled (using pmdb), and you build your final document (perhaps an exerquiz quiz), the end user can use an appropriate PDF reader. If an exerquiz quiz is used, then a minimum of Adobe Acrobat Reader is required.

2 The main code

```
\edef\th@dquoteCat{\the\catcode`\"}
\catcode`\"=12\relax
\newif\ifpmdbmode \pmdbmodetrue
```

2.1 Package options and package requirements

`dbmode` The default option is `dbmode`. When in effect, check boxes appear in the margins at each `\pmInput` point.

```
\DeclareOption{dbmode}{\pmdbmodetrue}
```

`!dbmode` A convenient way to turn off the creation of the check boxes is to simply place an exclamation point (!) in front of the `dbmode` option.

```
\DeclareOption{!dbmode}{\pmdbmodefalse}
```

`tight` When this option is taken, the checkboxes are tight against the text box area.

```
\newif\ifpmdbtight \pmdbtightfalse
\DeclareOption{tight}{\pmdbtighttrue}
```

`!tight` The default for the package, the checkboxes are placed to extreme left (or right) in the margins.

```
\DeclareOption{!tight}{\pmdbtightfalse}
\ProcessOptions
\RequirePackage{eforms}
```

2.2 Special attention to Thor

One motivation for this package is to support the thorshammer package, to that end we make the following assignment, if Thor is not present. This is to prevent stoppage: if you are inputting a `\RespBoxEssay` question that is accompanied by the `\essayQ` command, defined in thorshammer.

```
\def\pmdb@ckThor{\@ifundefined{essayQ}{\let\essayQ\@gobble}{}}
\AtBeginDocument{\pmdb@ckThor}
```

2.3 Some switches

Some Booleans and counters. The `\ifpmdbFP` switch is set to true when the path to the resource is a full path; otherwise, it is set to false. The `\ifpmdbDQs` switch is set to true of double-quotes are detected; otherwise it is false.

```
\newif\ifpmdbFP \pmdbFPfalse
\newif\ifpmdbDQs \pmdbDQsfalse
```

2.4 Form field creation

`\editSourceOn` `\editSourceOff` In version dated 2021/01/03, we introduce a button to edit the resource directly in the default application. If can be turned on with `\editSourceOn` and off again with `\editSourceOff`. The default is off. These commands can be placed anywhere in the document and affect all subsequent insertions of `\cbSelectInput`.

```
\newif\ifeditSource \editSourcefalse
\def\editSourceOn{\editSourcetrue}
\def\editSourceOff{\editSourcefalse}
\newcount\pmdb@Cnt
```

2.4.1 Marginal edit source creation

First up is the design for the edit pushbutton or link annotation.

`\editSourceBtn[`⟨*options*⟩`]{`⟨*wd*⟩`}{`⟨*ht*⟩`}` The button to show the current referenced content in the default application.

```
\newcommand{\editSourceBtn}[3][]{\def\@editSourceBtn##1{%
    \pushButton[\A{/S/Launch/F(##1)}\protect\A#1
    ]{editPb.\the\pmdb@Cnt}{#2}{#3}}}
```

`\editSourceBtn[`⟨*options*⟩`]{`⟨*wd*⟩`}{`⟨*ht*⟩`}{`⟨*txt*⟩`}` The link version of the button.

```
\newcommand{\editSourceLnk}[4][]{\def\@editSourceLnk##1{%
    \raisebox{1bp}{\setLinkBbox[\A{/S/Launch/F(##1)}
      \protect\A#1]{#2}{#3}[c]{\makebox[#2][l]{\thinspace#4}}}}}
```

Declare the appearance of this push button and link

```
\editSourceBtn[\TU{View in default editor}\S{S}]{11bp}{11bp}
\editSourceLnk[\linktxtcolor{red}\H{N}]{11bp}{11bp}{E}
```

`\useEditBtn` `\useEditLnk` We provide a control for setting the annot used to edit the source: `\useEditBtn`, a pushbutton is used; `\useEditLnk`, a link is used. The default is to use a pushbutton.

```
\def\useEditBtn{\def\@editSourceBorL{\@editSourceBtn}}
\def\useEditLnk{\def\@editSourceBorL{\@editSourceLnk}}
\useEditBtn
```

2.4.2 Marginal checkbox creation

`\cbSelectInput{`⟨*path*⟩`}` creates a check box with a tool tip of ⟨*path*⟩ The mouse up action `\ccBoxMU` fixes up relative paths, and defines a Ctrl+Click action. When the check box is so clicked, the ⟨*path*⟩ is opened in the default browser. The ⟨*path*⟩ can be relative or absolute. This command is used within `\insertCkBx`; its ⟨*path*⟩ argument is passed to it from `\insertCkBx`.

```
\def\pmCBPresets#1{\def\pm@CBPresets{#1}}
\pmCBPresets{}
```

Here is the check box that appears in the margins and the edit source button `\ifeditSource` is true.

```
\def\cbSelectInput#1{\mbox{\checkBox[\TU{#1}\presets{\pm@CBPresets}
  \cmd{\bParams{#1}{\the\pmdb@Cnt}\eParams}
  \AAmouseup{\ccBoxMU}]{pmdbCkBx.\the\pmdb@Cnt}{11bp}{11bp}{On}%
  \ifeditSource\olBdry\@editSourceBorL{#1}\fi}\global
  \advance\pmdb@Cnt\@ne
}
```

`\insertCkBx{`⟨*method*⟩`}` The argument of this macro describes the method of inserting the checkbox. The default definition works well for a straightforward document, where you are inputting ordinary LaTeX code (such as sections or chapters).

```
\def\insertCkBx#1{\def\@insertCkBx##1{#1}}
```

`\pmAlignCB` `\altCBMargins` Placement of check boxes in the margin. `\pmAlignCB` controls the marginpar placement. `\altCBMargins` alternates the margin placement, forces the check box to the extreme left (on odd pages) and extreme right (on even pages).

```
\def\setCBsMarg{%
  \ifpmdbtight
    \if@reversemargin
      \def\pmAlignCBAlt{\ifodd\value{page}\leavevmode
        \hfill\else\fi}\else
      \def\pmAlignCBAlt{\ifodd\value{page}\else\hfill\fi}\fi
  \else
    \if@reversemargin
      \def\pmAlignCBAlt{\ifodd\value{page}\hfil\else\hfil\fi}\else
      \def\pmAlignCBAlt{\ifodd\value{page}\hfill\else\fi}\fi
  \fi
}
\def\altCBMargins{\let\pmAlignCB\pmAlignCBAlt}
\def\pmAlignCB{%
  \if@reversemargin
    \ifpmdbtight\hfill\else\fi
  \else
    \ifpmdbtight\else\hfill\fi
  \fi
}
\@ifundefined{chapter}{}{\AtBeginDocument{\setCBsMarg\altCBMargins}}
```

This is the default declaration. It works well when you are inputting content that goes into horizontal mode. We insert the check box at the beginning of the first paragraph. When you are inputting files that come into a list environment, this method does not work satisfactory.

3 \pmInput states

The switch `\ifqzInput` is set to true when `\InputQuizItems` is expanded; otherwise, it is set to false.

```
\newif\ifqzInput \qzInputfalse
```

`\InputParas` This macro declares that the next `\pmInput` macros are for paragraph content. This is the default state of the package.

```
\def\InputParas{\global\qzInputfalse
  \insertCkBx{\ifpmdbmode
  \everypar{\marginpar{\pmAlignCB
    \cbSelectInput{##1}}\global\everypar{}}\fi}}
```

Set the default to be `\InputParas`

```
\InputParas
```

`\InputProbs` The `\InputProbs` is a command that declares the next `\pmInput` is for questions for an eqexam document. Currently this command is still under development.

```
\def\InputProbs{\global\qzInputfalse
  \insertCkBx{\ifpmdbmode
  \def\prior@questionsHook{\marginpar
    {\pmAlignCB\cbSelectInput{##1}}}\fi}}
```

`\InputQuizItems` The macro declares that the next `\pmInput` is for items in a `quiz` environment of exerquiz

```
\newcount\saveQNo \saveQNo\z@
\def\pmHook@qzItems{%
  \let\item@pmOld\item
  \def\item@pmNew{\item@pmOld\itemhook\let\item\item@pmOld}%
  \let\item\item@pmNew}
\def\InputQuizItems{\global\qzInputtrue
  \let\pmHook\pmHook@qzItems
  \saveQNo\z@
  \insertCkBx{\def\cbInQzMargin{\cbSelectInput{##1}}}%
  \ItemHook{\leavevmode\ifpmdbmode
  \ifnum\saveQNo<\value{eqquestionnoi}%
    \marginpar{\pmAlignCB\cbInQzMargin}\fi
  \saveQNo=\arabic{eqquestionnoi}\fi}}
```

`\InputItems` This command declares that the next `\pmInput` macros are for items in an list environment 2019/12/09 v0.4

```
\def\pmHook@item{\let\item@pmOld\item
  \def\item@pmNew{%
    \ifx\pmiarg\@empty
      \ifx\pm@Brk\ef@YES
        \def\pm@next{\item@pmOld[]}\else
        \let\pm@next\item@pmOld
      \fi
    \else
      \def\pm@next{\item@pmOld[\pmiarg]}%
    \fi\pm@next\itemhook\let\item\item@pmOld}%
  \let\item\item@pmNew
}
\def\ItemHook#1{\def\itemhook{#1}}
\def\InputItems{\global\qzInputfalse
  \let\pmHook\pmHook@item
  \insertCkBx{\def\cbInQzMargin{\cbSelectInput{##1}}}%
  \ItemHook{\leavevmode\ifpmdbmode
  \marginpar{\pmAlignCB\cbInQzMargin}\fi}}
```

Place checkbox and input ⟨*path*⟩

`\ckBxInput{`⟨*path*⟩`}` Places the check box and inputs the ⟨*path*⟩.

```
\let\pmHook\relax
\def\ckBxInput#1{\@insertCkBx{#1}%
  \ifpmdbDQs\def\donext{\pmHook\input{"#1"}}\else
    \def\donext{\pmHook\input{#1}}\fi
  \donext}
```

3.1 Other pushbutton creations

This package provides two form fields that are used for the DB step.

`\displayChoices[`⟨*options*⟩`]{`⟨*wd*⟩`}{`⟨*ht*⟩`}` inserts a push button whose action is to display the selections in the console window. The argument ⟨*wd*⟩ can be empty, in which case, the width of the field is determined from the `\CA` key.

```
\newcommand{\displayChoices}[3][]{\pushButton[\TU{\displayChoice@TU}
  \CA{\displayChoice@CA}#1\AAmouseup{\sldInputs}\protect\AA
]{sldInputs}{#2}{#3}}
```

`\displayChoiceCA`
`\displayChoiceTU`

Easy access to the `\CA` and `\TU` keys, for language localization, are provided. These are `\displayChoiceCA{`⟨*text*⟩`}` and `\displayChoiceTU{`⟨*text*⟩`}`.

```
\def\displayChoiceCA#1{\def\displayChoice@CA{#1}}
\def\displayChoiceTU#1{\def\displayChoice@TU{#1}}
\displayChoiceCA{Display Choices}
```

```
\displayChoiceTU{Display all choices in the console window}
```

`\clrChoices[`*⟨options⟩*`]{`*⟨wd⟩*`}{`*⟨ht⟩*`}` inserts a push button whose action is to clear all check boxes (and underlying JavaScript variables) created by this package. The argument *⟨wd⟩* can be empty, in which case, the width of the field is determined from the `\CA` key.

```
\newcommand{\clrChoices}[3][]{\pushButton[\TU{\clrChoices@TU}
  \CA{\clrChoices@CA}#1\AAmouseup{\clrAction}\protect\AA
]{Inputs}{#2}{#3}}
```

`\clrChoicesCA`
`\clrChoicesTU`

Easy access to the `\CA` and `\TU` keys, for language localization, are provided. These are `\clrChoicesCA{`*⟨text⟩*`}` and `\clrChoicesTU{`*⟨text⟩*`}`.

```
\def\clrChoicesCA#1{\def\clrChoices@CA{#1}}
\def\clrChoicesTU#1{\def\clrChoices@TU{#1}}
```

The default declarations for these two.

```
\clrChoicesCA{Clear Choices}
\clrChoicesTU{Clear all check boxes created by pmdb}
```

3.2 Defining the `\pmInput` command

`\pmInput*[`*⟨arg⟩*`]{`*⟨path⟩*`}` is the main user-interface for inputting a file; here, the macro's name is `\pmInput`, ultimately it calls `\input` with the same path. Paths with spaces must be enclosed in double quotes (`\pmInput{my cool problem.tex}`) and the extensions must always be used. It the `*` option is specified, `\pmInput` gobbles up all remaining arguments and does nothing otherwise. The optional argument *⟨arg⟩* is only obeyed when `\InputItems` is in affect; it is passed on to the underlying `\item`, as in `\item[`*⟨arg⟩*`]`. The *⟨path⟩* argument is the path to the content to be input; if the *⟨path⟩* contains spaces, then the path must be enclosed in double quotes (`"my cool path/file.tex"`).

```
\newcommand\@gobbleOR[2][]{}
\def\pmInput{\@ifstar{\@gobbleOR}{\pmInput@i}}
\def\pmInput@i{\@ifnextchar[%]
  {\let\pm@Brk\ef@YES\inputConta}
  {\let\pm@Brk\ef@NO\inputConta}}
\let\pm@Brk\ef@NO
\def\inputConta{\bgroup\@makeother\"\inputContb}
\newcommand\inputContb[2][]{\egroup\def\pmiarg{#1}\inputConti#2;;}
```

Determine if double quotation marks are used.

```
\def\inputConti{\@ifnextchar"%
  {\global\pmdbDQstrue\removedqs}
  {\global\pmdbDQsfalse\removesemis}}
\def\removedqs"#1";;{\inputContii{#1}}
\def\removesemis#1;;{\inputContii{#1}}
```

Determine if this is a full path, we do this by searching for a colon (`:`). Following the search for the colon, pass on to the final step of `\doinput`.

```
\def\inputContii#1{\isItFullPath#1:\@nil\doinput{#1}}
```

A command to detect presence of a colon.

```
\def\isItFullPath#1:#2\@nil{%
  \def\@rgii{#2}\ifx\@rgii\@empty
    \global\pmdbFPfalse\else
    \global\pmdbFPtrue\fi}
```

Final step, if the switch `\ifpmdbmode` is true, we insert the check box `\ckBxInput`; otherwise, we pass ⟨*path*⟩ to `\input`.

```
\def\doinput#1{\ifpmdbmode\def\donext{\ckBxInput{#1}}\else
  \ifpmdbDQs\def\donext{\input{"#1"}}\else
    \def\donext{\input{#1}}\fi\fi
  \donext}
```

During the development of this package, the original command name used was `\Input`. There are a few users that use this old definition; the command `\Input` is defined in other package, in particular in the srcltx. So we allow the use of `\Input` if `\Input` is not otherwise defined.

```
\def\pmInputWarni{\PackageWarningNoLine{pmdb}{The command
  \string\Input\space is already defined.\MessageBreak
  The checkboxes may not appear in the margins.\MessageBreak
  Use the supported command \string\pmInput\space instead}}
\def\pmInputWarnii{\PackageWarningNoLine{pmdb}{Letting
  \string\Input\space to \string\pmInput. You are \MessageBreak
  encouraged to use the supported\MessageBreak
  command \string\pmInput\space instead}}
\def\pmInputChk{\@ifundefined{Input}{\let\Input\pmInput
  \pmInputWarnii}{\pmInputWarni}}
\AtBeginDocument{\pmInputChk}
```

4 Field JavaScript

`\ccBoxMU` This is the JavaScript action of the check box, used in `\cbSelectInput` This code uses two parameters `\p(1)=`⟨*path*⟩ and `\p(2)=`⟨*cnt*⟩ (`\the\pmdb@Cnt`).

```
\begin{defineJS}[\makeesc\@\makecmt\%]{\ccBoxMU}
@ifpmdbFP%
event.target.userName=("@p(1)");
@else%
```

This part of the code is Windows specific. Don't know enough about Mac OS to form the proper path.

```
// device independent path
var pos=this.path.lastIndexOf("/");
var thispath=this.path.substring(0,pos+1);
```

```
/<drive>/user/documents/.../myfolder/
```

```
pos=thispath.indexOf("/",1);
var drive=thispath.substring(1,pos);
var platform=app.platform;
```

Windows platform

```
if (platform=="WIN")
```

```
<drive>:/user/documents/.../myfolder/
```

```
  thispath=drive+":/"+thispath.substring(pos+1);
```

MacOS platform: I'm not familiar with the MacOS file system, so we'll just assume it is the same as with Windows.

```
if (platform=="MAC")
```

```
<drive>/user/documents/.../myfolder/
```

```
  thispath=drive+"/"+thispath.substring(pos+1);
event.target.userName=thispath+("@p(1)");@fi
if (event.modifier){
  event.target.checkThisBox(0,!event.target.isBoxChecked(0));
  try {
    aebTrustedFunctions(this,aebLaunchURL,%
{cURL: "file:///"+event.target.userName});
  } catch(e) {
    console.show();
    console.println("The Ctrl+Click action is not supported, %
installation of aeb\_pro.js or aeb-reader.js is required.");
  }
```

(2020/04/29) Add a shift event for quizzes.

```
}
if (event.shift) {
event.target.checkThisBox(0,!event.target.isBoxChecked(0));%
@ifqzInput
  this.gotoNamedDest("@currQuiz."+(@p(2)+1));@fi
} else {
  if (event.target.isBoxChecked(0)){
    _oSPaths["pmdbCkBx.@p(2)"]=%
[("@p(1)"),@ifpmdbDQs true@else false@fi];
    _aInputs[@p(2)]=true;
    _numInputs++;
  }else{
    _oSPaths["pmdbCkBx.@p(2)"]=null;
    _aInputs[@p(2)]=false;
    _numInputs--;
  }
  event.target["_boxState"]=!!event.target.isBoxChecked(0);
}
\end{defineJS}
```

`\sldInputs` This is the mouse up action for a push button. It lists all selected content and displays them in the console window of Acrobat/Reader, used in `\displayChoices`.

```
\begin{defineJS}[\makecmt\%]{\sldInputs}
console.clear();console.show();
if (_numInputs==0) console.println("No inputs selected");
else {
  for(var i=0;i<_aInputs.length;i++){
    if (!!_aInputs[i]){
      if(_oSPaths["pmdbCkBx."+i][1])
        console.println('\\\\input\{\\"'+(_oSPaths["pmdbCkBx."+i][0])%
+'\\"\}');
      else
        console.println('\\\\input\{'+(_oSPaths["pmdbCkBx."+i][0])%
+'\}');
    }
  }
}
\end{defineJS}
```

`\clrAction` Mouse up action to clear the checkboxes and to re-initialize internal internal JS variables. Used in `\clrChoices`.

```
\begin{defineJS}{\clrAction}
var _oSPaths=new Object;
var _aInputs=new Array;
var _numInputs=0;
this.resetForm("pmdbCkBx");
\end{defineJS}
```

5 Document JavaScript

```
\begin{insDLJS}{mrki}{Supporting JavaScript for pmdb}
var _oSPaths=new Object;
var _aInputs=new Array;
var _numInputs=0;
\end{insDLJS}
```

```
\catcode`\"=\th@dquoteCat
⟨/package⟩
```

6 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

7 Change History

v0.4 (2019/12/09)
  General: Added `\InputItems` 6
    Modified to work when no points are specified 5
v0.5 (2020/04/29)
  General: Added `event.shift` for quizzes 9
v0.6 (2021/01/03)
  General: Added `\editSourceOn` and `\editSourceOff` 3
  `\editSourceBtn`: Added `\editSourceBtn` 3
    Added `\editSourceLnk` 3
    Added `\useEditBtn` and `\useEditLnk` 4
    Added hard wired action to launch the default application. 4
v1.0 (2021/06/07)
  General: Upgrading v0.6 to v1.0 for publication 2