基礎プログラミングII

第8回　CGIプログラム (2)・LaTeX

今回学ぶこと

1.データを永続させるCGI
2.CGIへの画像投稿
3.LaTeX

今回学ぶこと

1.データを永続させるCGI

2.CGIへの画像投稿

3.LaTeX

CGI変数の寿命

CGIでは、webページで入力された値はCGIスクリプトの終了とともに消えてしまう。

PStoreクラス

現在の変数（数値、文字列、配列、ハッシュなど）の値をそのままの形でファイルに保存できる。

PStoreクラスにハッシュの値を保存する例

```
require 'pstore'
x = PStore.new("値保存ファイル")
x.transaction do
  x["foo"] = x["foo"] || Hash.new # x["foo"] が空なら新規ハッシュ代入
  保存したい変数 = x["foo"]
    :
    :
end
```

PStoreクラスの利用法

「名前」と「ひとこと」を読み込んで出力するRubyプログラム：

```
#!/usr/koeki/bin/ruby
# coding: utf-8

word = Hash.new

print "名前は?: "
name = gets.chomp
print "ひとこと: "
word[name] = gets.chomp

for who, wd in word        # または word.each do |who, wd|
  printf("%sさんのひとこと「%s」\n", who, wd)
end
```

PStoreクラスの利用法

実行例：

```
% ./word.rb
名前は?: taro
ひとこと: hoge
taroさんのひとこと「hoge」
% ./word.rb
名前は?: hanako
ひとこと: やほー
hanakoさんのひとこと「やほー」
% ./word.rb
名前は?: John
ひとこと: Hey, Jude
Johnさんのひとこと「Hey, Jude」
```

最後でハッシュに含まれるキーと値のペアを全て出力しているが、プログラムを一度動かしても１つのキーと値しか登録されないので、１人の言葉しか出てこない。

PStoreクラスの利用法

PStoreを使って、そのときのハッシュ全体の値を別ファイル（以下の例ではword.db）に保存して毎回読み込むようにする。

```
#!/usr/koeki/bin/ruby
# coding: utf-8

require "pstore"
x = PStore.new("word.db")
x.transaction do
  x["word"] = x["word"] || Hash.new
# x["word"] ||= Hash.new でも可
  word = x["word"]

  print "名前は?: "
  name = gets.chomp
  print "ひとこと: "
  word[name] = gets.chomp

  for who, wd in word
    printf("%sさんのひとこと「%s」\n", who, wd)
  end
end
```

PStoreクラスの利用法

実行例：

```
% ./word.rb
名前は?: taro
ひとこと: hoge
tarooさんのひとこと「hoge」
% ./word.rb
名前は?: hanako
ひとこと: やほー
hanakoさんのひとこと「やほー」
tarooさんのひとこと「hoge」
% ./word.rb
名前は?: John
ひとこと: Hey, Jude
Johnさんのひとこと「Hey, Jude」
hanakoさんのひとこと「やほー」
tarooさんのひとこと「hoge」
```

以前に起動された値をword.dbファイルに自動的に保存し、前回の値を引き継ぐことができる。

データを保存するCGIの実行

CGIプログラムにデータファイルを書かせるためには当該ディレクトリを他人でも書き込めるように設定する。

1. データを書き込むディレクトリを別途作成する。

```
mkdir ~/public_html/mycgi/data
```

2. 誰にでも書き込みできる既存ファイルを消すのはファイル所有者のみ、という属性をディレクトリに設定する。

```
chmod 1777 ~/public_html/mycgi/data
```

3. ~/public_html/mycgi/にword-pstore-cgi.rbを保存し、chmod +xする。

実用CGIスクリプトへ

自分用データベース

PStoreクラスを利用して、何かの飲み物を飲んだときの感想を登録できるデータベースを作る。入力名として、

飲み物の名前	item
感想	comment

を利用する。

実用CGIスクリプトへ

入力フォームの出力と、現在登録されているデータの出力を両方とも行うようなCGIプログラムにする。

CGIプログラムの構成

1. HTMLヘッダ等の出力
2. データ入力用フォームの出力
3. その時点で何か値が入力されていたらそれをハッシュに追加
4. 既存データの値一覧を出力

今回学ぶこと

1.データを永続させるCGI

2.CGIへの画像投稿

3.LaTeX

CGIへの画像の投稿

HTMLフォームinput要素の**type = “file”**を利用することで、CGIへ画像を投稿できる。

multipart/form-data

文字列や選択オプションを送信するフォーム：

```
<form method="POST" action="./hoge.rb">
 一言メモ: <input name="var" size="40"><br>
 <input type="submit" value="SEND!">
 <input type="reset" value="reset">
</form>
```

multipart/form-data

ファイルの中身を投稿するには、フォームのenctype属性をmultipart/form-dataに変える。

```
<form method="POST" enctype="multipart/form-data"
action="./hoge.rb">
 一言メモ: <input name="var" size="40"><br><br>
 画像を選んでね: <input name="image" type="file">
 <input type="submit" value="SEND!">
 <input type="reset" value="reset">
</form>
```

multipart/form-data

HTMLでmultipart/form-dataを指定して入力させた値を受け取るCGIスクリプト：

```
require "cgi"
cgi = CGI.new(:accept_charset => "UTF-8")


memo = cgi["var"].read                          # CGI変数 var の受け取り(text)
photo_filename = cgi["image"].original_filename # ファイル名の受け取り
photo_size = cgi["image"].size                  # 送信ファイルサイズ取得
photo_data = cgi["image"].read                  # 送信ファイルデータの受け取り
```

実際のCGIフォームでの送信値を得るにはreadメソッドを介して行う。

multipart/form-data

読み取った画像データをファイルに保存：

```
open("ファイル名", "w") do |out|
  out.write cgi["image"].read
end
```

ファイルを保存するディレクトリはWebサーバ権限で書き込めるようにしておく(chmod 1777)。

今回学ぶこと

1.データを永続させるCGI

2.CGIへの画像投稿

3.LaTeX

マークアップ言語

HTML言語のように「文書タイトル、見出し、本文、箇条書き」などの情報を文書自体に決められた記法で埋め込んでいくもの

LaTeX（ラテフ）

論文やレポート記事などを出版物と同じ品質で作成されるために設計されたマークアップ言語

LaTeXによる文書作成手順

1. 本文（ソース）作成
2. 組み版処理（タイプセット）
3. 印刷

LaTeXによる文書作成手順

LaTeXソース

↓

DVIファイル

Device Independent（デバイス非依存）ファイル

↓

PDFファイル

Portable Document Format（汎用文書形式）ファイル

↓

プリントアウト

テキストエディタ (Emacsなど)

タイプセッタ (uplatexコマンド)

PDF変換 (dvipdfmx コマンド)

PDFビューア (evince コマンド等)

PostScript変換 (dvipsコマンド)

LaTeXによる文書作成手順

1. 最初にLeTeXの文法に則った形で論文やレポートのソースを書く。
2. このファイルを保存し、タイプセッタにかける。

```
uplatex foo.tex && dvipdfmx foo
```

3. 文法エラーが報告された場合はエラーの出た行番号を見てソースを修正する。
4. エラーが出ずにタイプセットとPDF変換が完了した場合は、ソースファイルと同じディレクトリにPDFファイルができているので、これを開き仕上がりイメージを確認する。

```
evince foo.pdf &
```

5. 文書が完成するまで2から4を繰り返す。

LaTeXソースの構造

```
\documentclass[uplatex]{jsarticle}
\begin{document}
こんにちは!
\end{document}
```

jsarticle: 文書スタイル (他にjsbookやjsreportなど)
document環境：この中に本文を記述

LaTeXの文法

- LaTeX処理システムに特別に与える記号はバックスラッシュで始める
- バックスラッシュと英単語で始まるものを**マクロ**という
- マクロには{ }で括って引数を与えることがある
- \begin{}と\end{}に囲まれた部分を**環境**と呼ぶ。

文書スタイル

- jsarticle: 日本語による短い記事
- jbook: 本（日本語）
- jreport: レポート（日本語）
- tarticle: 縦書き文書（日本語）

など

document環境

\begin{document}, **\end{document}**で文書本体の開始と終わりを宣言する。
ファイルの先頭から\begin{document}までの部分を**プリアンブル**という。

```
\documentclass[uplatex]{jsarticle}
（プリアンブル）
\begin{document}
（文書本体）
\end{document}
```

LaTeXソースの例

\title{タイトル}

文書のタイトルを指定する。プリアンブルに記述する。\\は改行を意味する。

```
\documentclass[uplatex]{jsarticle}
\title{地球温暖化を食い止めよう}
\author{東北公益文科大学\\公益太郎}
\begin{document}
\maketitle
\section{はじめに}
やばそうよね。
\section{海水面上昇}
大丈夫なのか湊町酒田!?
\end{document}
```

LaTeXソースの例

\author{著者}

文書の著者を指定する。プリアンブルに記述する。

```
\documentclass[uplatex]{jsarticle}
\title{地球温暖化を食い止めよう}
\author{東北公益文科大学\\公益太郎}
\begin{document}
\maketitle
\section{はじめに}
やばそうよね。
\section{海水面上昇}
大丈夫なのか湊町酒田!?
\end{document}
```

LaTeXソースの例

\maketitle

この位置に文書のタイトルを出力する。通常document部の先頭に指定する。

```
\documentclass[uplatex]{jsarticle}
\title{地球温暖化を食い止めよう}
\author{東北公益文科大学\\公益太郎}
\begin{document}
\maketitle
\section{はじめに}
やばそうよね。
\section{海水面上昇}
大丈夫なのか湊町酒田!?
\end{document}
```

LaTeXソースの例

\section{セクションタイトル}

文書の節を開始する。

```
\documentclass[uplatex]{jsarticle}
\title{地球温暖化を食い止めよう}
\author{東北公益文科大学\\公益太郎}
\begin{document}
\maketitle
\section{はじめに}
やばそうよね。
\section{海水面上昇}
大丈夫なのか湊町酒田!?
\end{document}
```

LaTeX見出し

見出しの区切りの意味の大きいものから順に

1. \part
2. \chapter
3. \section
4. \subsection
5. \subsubsection
6. \paragraph
7. \subparagraph

ただし、\part, \chapterは文書スタイルjsarticleでは使えない

記号入力時の注意

いくつかの記号はそのまま入力できない

入れたい記号	LaTeXへの入力
%	\%
#	\#
$	\$
_	_
|	\textbar
~	\~{}または\textascciitilde
^	\^{}または\textasciicircum
<	$<$
>	$>$
\	$\backslash$または\textbackslash
{	\{
}	\}
丸つき文字	\textcircled{文字}

記号入力時の注意

その他、どうしてもそのまま入れたい文字は**\verb**で入れたい文字を挟む。例えば、

```
\verb|\<>|
```

とすれば、\<>がそのまま出力される。

itemize環境

箇条書きの出力：

```
\begin{itemize}
\item 出身地
\item 好きなもの
\end{itemize}
```

オンラインLeTeXエディター Overleaf (オーバーリーフ)

ブラウザ上でLaTeXファイルの編集や共有ができる。

https://ja.overleaf.com

まとめ

- CGIでデータを永続させるにはPStoreクラスを用いる。
- input要素のtype = “file”を利用すると、CGIへ画像を投稿できる。
- LaTeXを用いると、出版物のような形式の整った文書を作成できる。