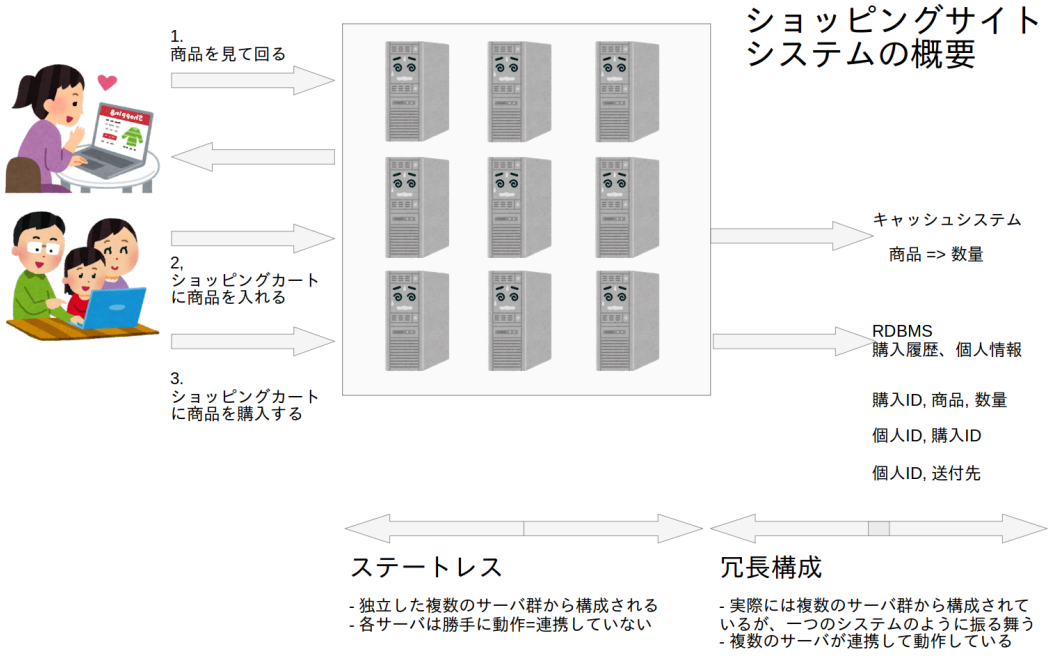
課題: www.pyのcart機能の実装

www.py v1.84のcart関数(190行目以降)の改造です。 とりあえずノーヒントで頑張ってみてほしいですね。

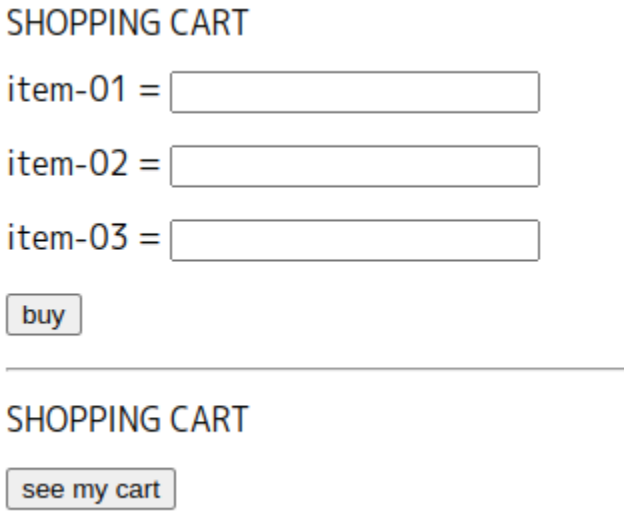
課題(第11回～第12回)

- 右図の2.の部分を、キャッシュシステムも用いて、きちんと作ります。 実現することは
  1. ショッピングカートに数量をいれて送信すると、その数量が返ること
  2. そして、この数量が、キャッシュに記録されていること
  - 実行例は次頁以降を参照
  - 第10回以降はCookieの利用が前提です。 この**「サーバがブラウザに一時的に与えるパスワード」相当のCookieが必須。** ？の人は第10回の演習パートを復習してください

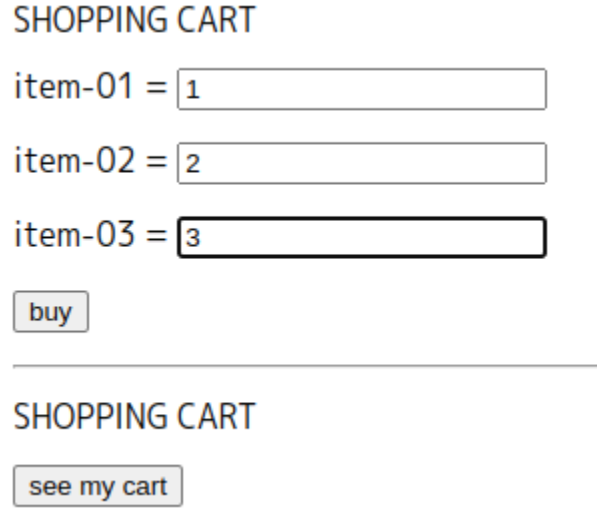


(脚注) www.pyの動作の見かけは今までと変わりません。ただし裏側は異なる状態です。 来週（第12回）、 1号機のwww.pyで保存した数量が2号機のwww.pyでも表示されることを確認します

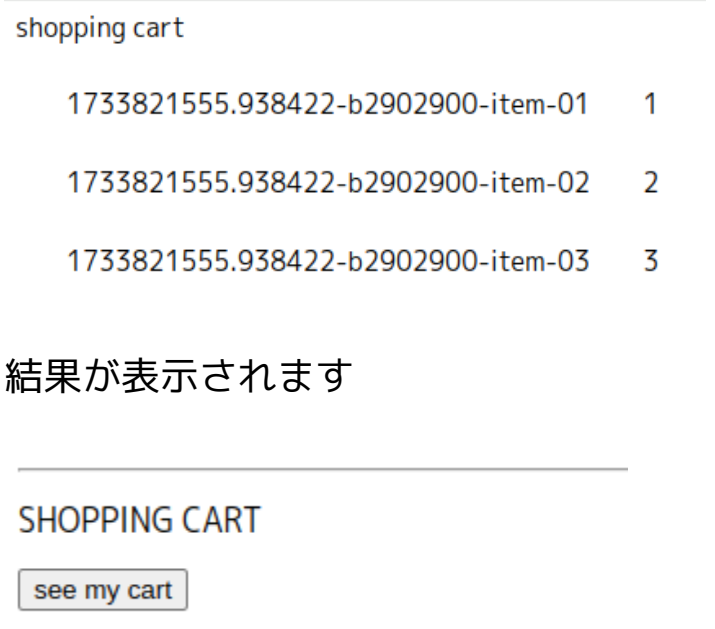
実行例(第11回)



ショッピングカートを開き



数量をいれ、送信すると



結果が表示されます

最初の画面に戻り、一番下「see my cart (カートを見る)」をクリックすると同じ結果が表示されるはずです

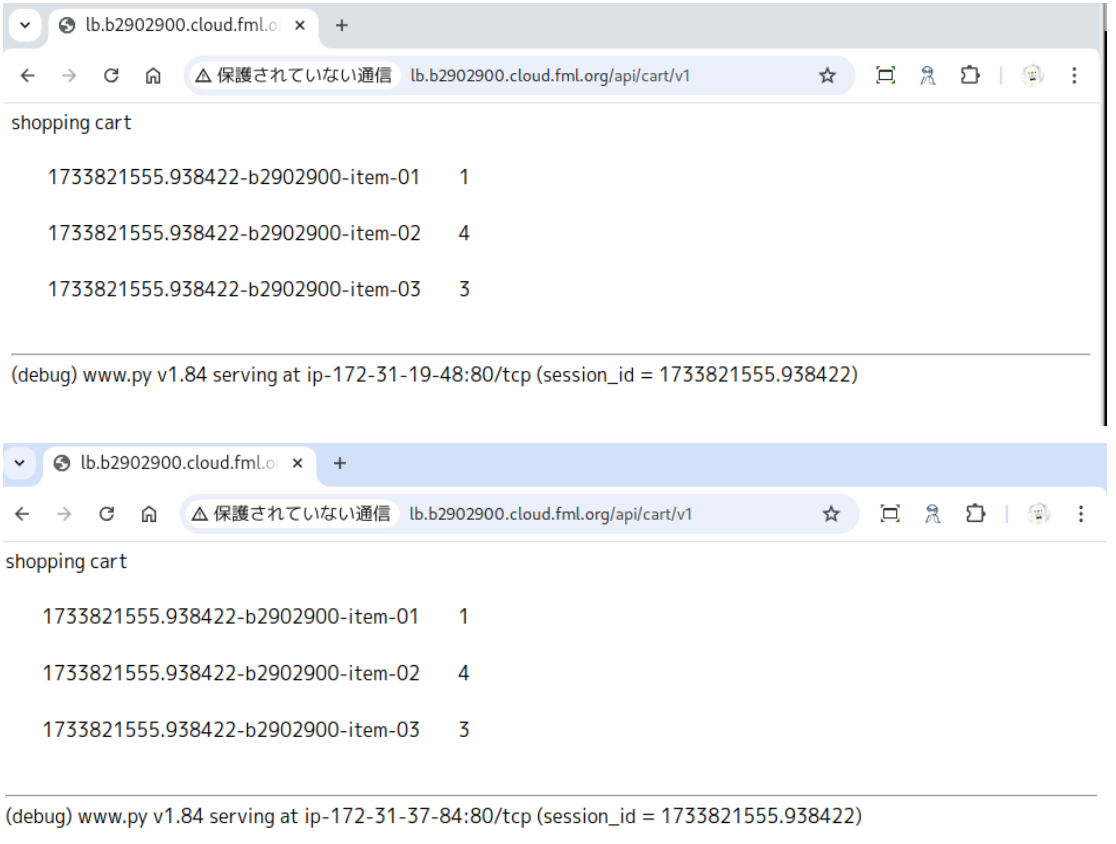
(脚注) 裏側の動作が正しいのかどうかは、 ブラウザの画面だけ見ても分かりません;-)

(1) リロードしても同じ出力です (2) 二号機でも(ホスト名以外)同じ出力です、次頁を参照

(2)は第12回分の予定ですけど、進められる人は進めてかまいません。 そのぶん最終課題に早く取り組めるというものです、うむ

実行例(第12回以降)

- 右図、**画面下部の(debug)行で、ホスト名だけが変わっている**ことが分かるでしょうか？
  - 二つのEC2を動かしてアクセスするので
  - 表示されるカートの情報は同じですが、
  - (debug) 行のホスト名とブラウザのURLは異なります



(脚注) どんな構成でも、この見栄えは同じです。 ただし、 このスクリーンショットでURLが同じなのはELB構成で取得したから

第11回の仕様: www.py を改造し、キャッシュにデータを保存する

```
http://学籍番号.cloud.fml.org/cart.html
```

- (a) 上のURL(cart.html)にアクセスし、
- (b) 商品の数量を入力
- (c) 送信する
- (d) Web APIサーバは受け取った数量をブラウザに返す
  - (裏側ではキャッシュに商品と数量の組を保存する) (NEW)
  - **ブラウザとのやりとりにはCookie Session IDをつける(実装済,開発は不要)**
- (e) ブラウザが表示する
- (f) **リロードもしくは再度カートにアクセスすると、先ほどの商品と数量が表示される (NEW)**
- Web APIサーバのURLは次のものとする

```
http://学籍番号.cloud.fml.org/api/cart/v1
```

(脚注) 新しいwww.pyのソースコード中にcart関数の"ひながた"は用意されています。 次ページ以降の説明にそって、書いてください

解説: キャッシュシステム

[構成図]

```
www.py ---(key,value)---> キャッシュシステム
```

- Pythonでは、 `cache = { key: val }` と書けば、辞書(dict変数)cacheにキー(key)と値(val)を設定できます。 これのWeb API版だと思ってください
- ただ、dict変数のように簡単には書けません。 bws.py モジュールにあるコードを使い、 書きこみ(set)と読みこみ(get)をします

[www.pyのコード例]

```
import bws
cache = bws.cacheInit(host, port, password)

bws.cacheSet(cache, "123456789-b292900-item-01", 3)
v = bws.cacheGet(cache, "123456789-b292900-item-01")
print(v) # 3 と表示される
```

- AWSの本物を使うと高価なので、 代わりに（うちの偽クラウド）BWSを使ってください
- **全員で同じサーバを使うので、 キーには長い文字列(session-id+学籍番号+商品名)が必須です**

(脚注1) プログラミングがメインの授業ではないので、 bws.pyはブラックボックスのままでよいです。 興味のある人はコードを読んでください (脚注2) AWS Academyでもキャッシュサービス(AWS ElastiCache)は使えるのですが、 もともと高価な上に、サービスを止められないので、課金がすごいことになります。 そこでインチキクラウドBWS(Bibi Web Serices)の登場です(w)

演習の準備(ダウンロードとPythonモジュールの追加)

1. ファイルを3個、ダウンロードしてください。 URLは次のとおりです **(注: https になりました！)**

```
https://api.fml.org/dist/www.py
https://api.fml.org/dist/bws.py
https://api.fml.org/dist/cart.html
```

   - www.pyは、新しいバージョンに入れ替えます
   - cart.htmlは文字化けしないように英語だけに戻しました
2. Pythonのredisモジュールをインストールしてください。

```
$ sudo apt install -y python3-redis
```

   - Redisというキャッシュシステムへアクセスするためのライブラリです

(脚注) 注意事項: いろいろダウンロードするものがあります。確実にダウンロードを実行してください

演習の準備(ダウンロードとPythonモジュールの追加)の確認

- ファイルの配置が次のようになっていることを確認してください
  - www.pyとbws.pyは/home/admin直下に置いてください
  - cart.htmlはhtdocs(www.pyが見せるコンテンツの置き場所)に置いてください

```
[例] treeコマンド(に似た)出力

/home/admin
|-- bws.py
|-- htdocs
|   +-- cart.html
|   +-- index.html
+-- www.py

2 directories, 4 files
```

ステップ０: CGI FORMの操作法を勉強する

- まずはlsformという関数(www.py v1.84の154行目〜)を読んでください。 これは送られてきたキーと値の一覧を表示するという、 (そのまま何かに使えそうな?)FORMを操作する関数の良い見本になっています
- 関数の仮引数はselfとformです。 self(とかthis)はオブジェクト指向言語のアレ(関数を呼び出したインスタンス自身?)です。 変数formにはHTMLのFORM文で送られてきたデータが一式はいっています
- `form.keys()` はFORMにあるキーの全てが入っている配列を返します
- 特定のキーの値は `form[key].value` で取得できます。 `form[key]` ではありません！ここは要注意

[コードの例]

```
# すべてのキーを取り出し for ループに投入します。key には item-01, item-02, ... が順に入ります
for key in form.keys():
   val = form[key].value # key (例: item-01)に対応する値を val に代入
```

(脚注1) Python公式のマニュアル -> https://docs.python.org/ja/3.11/library/cgi.html

(脚注2) form[key] はオブジェクトです。 FORMで送られてくるものが文字列とは限らないですしね

解説: www.pyのcart機能の実装

www.py v1.84のcart関数(190行目以降)の改造です。 とりあえずノーヒントで頑張ってみてほしいですね。

なお、次ページ以降のスライドはヒントです。 www.py にはコメントとして(1)(1A)などと印があります。

次ページ以降で、それぞれの部分でやるべきことを順に説明していきます

ステップ1: 入力

```
  def cart(self,form):
    # (1) INPUT
    user_id = "b2xxyyyy"
    session = self._get_session_cookie()
    # (1A)
```

- CGIのFORMからブラウザが送ってきたキーと値を取り出します
  - cart.htmlを見ればわかるとおり、キーは item-01 〜 item-03 で、値は数字
- `user_id` は各自の学籍番号に書き換えてください
- `session` 行は、そのままにしてください。session変数にはcookieのセッションIDが入ります
- `# (1A)`行以降で、キャッシュに書きこむキーと値を準備してください
  - キャッシュに書きこむキーは、一意になるように「セッションID-学籍番号-FORMのキー」とします

```
例: 1733750385.8418589-b2902900-item-01
```

www.py と言わず、すべての処理は次の3ステップの積み重ねである。 (a) データを入力し、 (b) 何かの処理をして (c) 結果を出力する

ステップ２: 関数固有の仕事(ここではキャッシュへの書きこみ)

```
# (2) DO
import bws
cache = bws.cacheInit(redisHost, redisPort, redisPass, session)
# (2A)
```

- キャッシュに書きこみます。キャッシュを操作する関数はbwsモジュールに用意されています
- cacheInit()のパラメータを変更し、`# (2A)`行以降に、キャッシュに書きこむコードを書いてください
- `bws.cacheInit(ホスト, ポート番号, パスワード, session)`の部分は、 ポータルサイトにパラメータが書いてあります。それを使ってください。
- 用意されている関数の使い方は次のとおり

```
bws.cacheInit(ホスト, ポート番号, パスワード, セッションID)
bws.cacheGet(cache, キー)
bws.cacheSet(cache, キー, 値)
```

- cacheGet()とcacheSet()の第1引数cacheはcacheInit()が返した値です

www.py と言わず、すべての処理は次の3ステップの積み重ねである。 (a) データを入力し、 (b) 何かの処理をして (c) 結果を出力する

ステップ3: 出力

```
    # (3) OUTPUT
    msg = "<p>shopping cart\n<table border=0 frame=void cellspacing=30>\n"

    # (3A)
    for r_key in sorted(bws.cacheSmembers(cache, session)):
       msg = msg + "<tr><td>{}<td>{}</tr>\n".format(r_key,r_val)

    msg = msg + "</table>\n"
    return msg
```

- 関数から返すHTMLを変数msg (文字列)として組み立ててください。 HTMLの雛形は、ここにすべて書いてあります。 あとは返す値は何か？(適切なキーと値)を考えてください
- `# (3A)`の繰り返し文は、そのまま使ってください
  - `r_key`はredis keyの略でキャッシュサービスで使っているキーを意味しています
  - **`bws.cacheSmembers(cache, session)`は 同一Cookieセッション**IDで操作した全商品のキーを返します
    (この演習で便利なように考えてある)

www.py と言わず、すべての処理は次の3ステップの積み重ねである。 (a) データを入力し、 (b) 何かの処理をして (c) 結果を出力する