

Slide URL

<https://vu5.sfc.keio.ac.jp/slide/>

Web情報システム構成法  
第7回 フォームインタラクション

萩野 達也 (hagino@sfc.keio.ac.jp)

# 静的Webページ vs 動的Webページ

---

## ▶ 静的Webページ

- ▶ 内容が変わらないページ
- ▶ 通常の文書は静的
- ▶ HTMLとしてWebサーバ上に置いておく
- ▶ 維持管理の関係で動的に生成していることもある

## ▶ 動的Webページ

- ▶ 内容が変化するページ
- ▶ 利用の状態によって中身が変化する
- ▶ Webアプリケーションでの利用
- ▶ 例：
  - ▶ オンラインショッピングのショッピングカート
  - ▶ 検索エンジンの結果

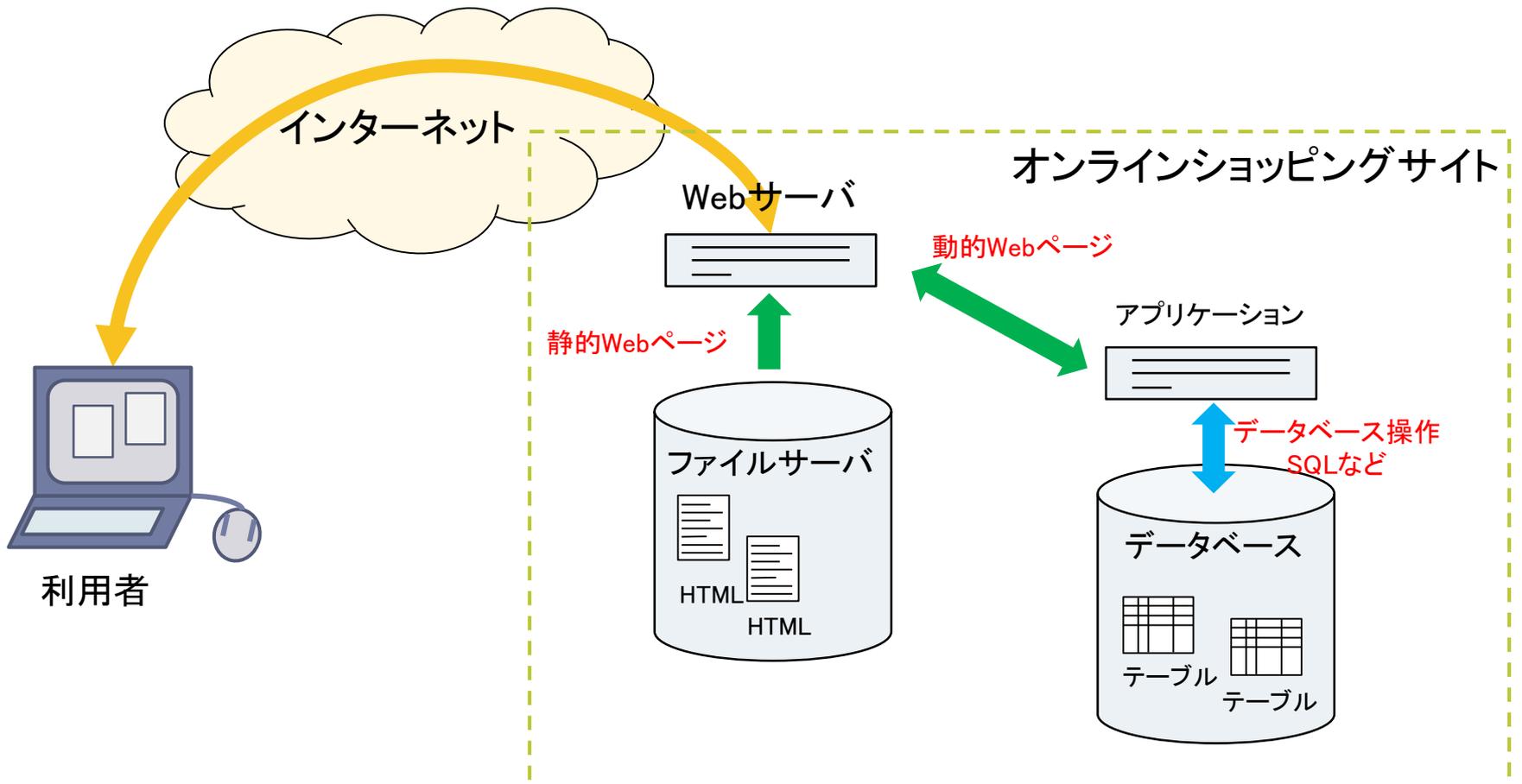
# オンラインショッピングの例

---

- ▶ 静的に用意しても良いページ
  - ▶ お店に関する情報を書いたページ
  - ▶ 買い物の仕方を説明したページ
  - ▶ 商品に変化が少ない場合には、商品の説明も静的に用意しても良い
- ▶ 動的に用意しなくてはならないページ
  - ▶ 在庫が変化する商品に関するページ
  - ▶ ショッピングカートの中身を表示するページ
  - ▶ 決算を行うページ
  - ▶ ユーザ登録するページ
  - ▶ キーワードなどを入れて商品を絞り込むページ

# Webアプリケーション

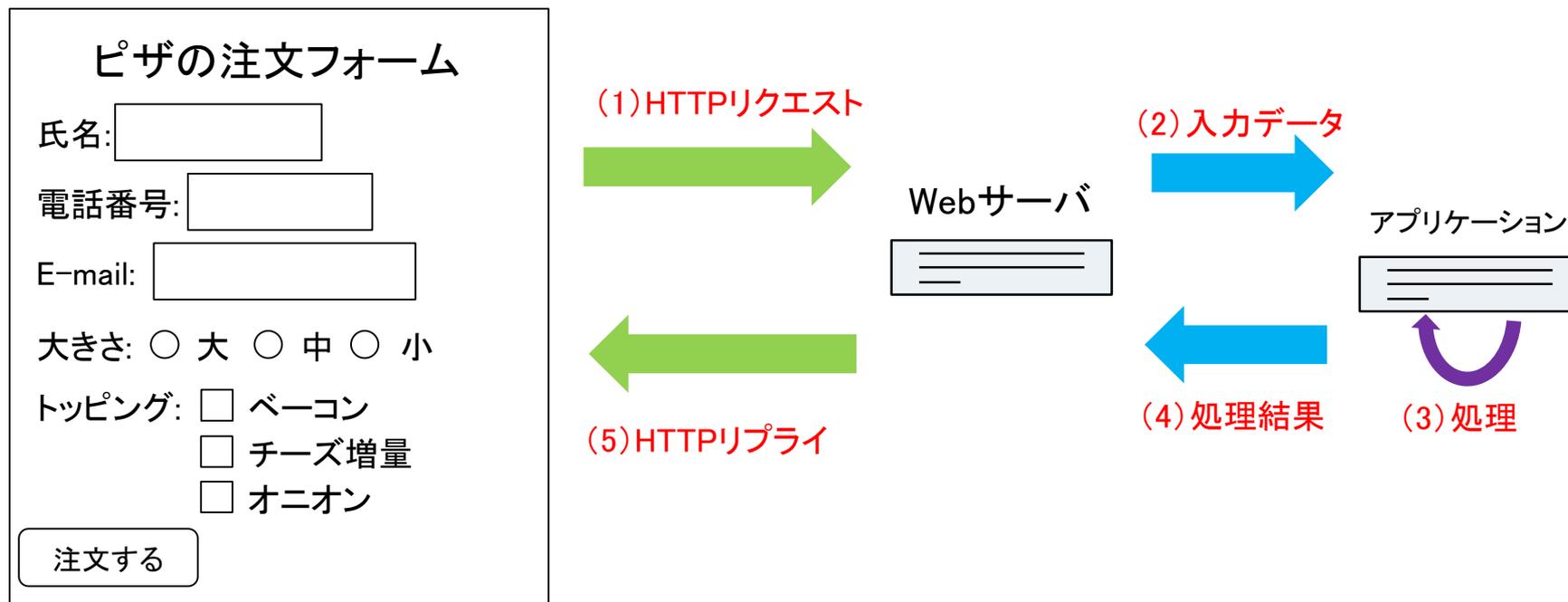
## ▶ オンラインショッピングのためのWebサイトの構成



# Formインタラクション

## ▶ form

- ▶ データを入力するためのフォームを表示する
- ▶ ユーザの入力をWebアプリケーションに渡す
- ▶ HTTPのGETあるいはPOSTによりデータを渡す
  - ▶ GET: URLに入力データをエンコード
  - ▶ POST: HTTPリクエストの本体として入力データを送る



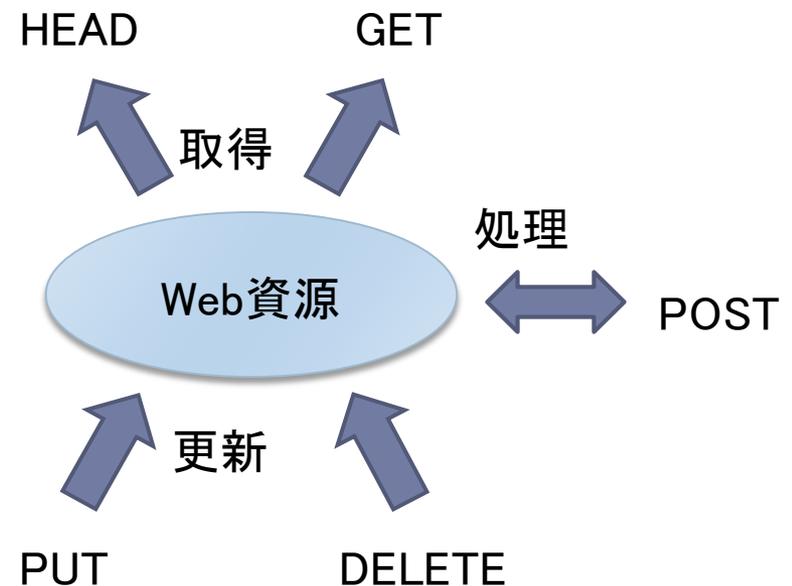
# HTTP (Hypertext Transfer Protocol)

---

## ▶ Web資源を操作するプロトコル

### ▶ 5つの主なメソッドを持つ

- ▶ HEAD
  - ▶ 資源の情報を得る
- ▶ GET
  - ▶ 資源の表現を取得する
- ▶ PUT
  - ▶ 資源を作成あるいは更新する
- ▶ DELETE
  - ▶ 資源を削除する
- ▶ POST
  - ▶ データを処理するために送る



# HTML form要素

---

```
<form method="メソッド" action="URL" enctype="エンコーディング">  
    フォームの中身  
</form>
```

## ▶ メソッド

- ▶ get あるいは post を指定
- ▶ 副作用がないときには get
- ▶ 副作用があるときには post
- ▶ 指定がないときには get と理解される

## ▶ URL

- ▶ 処理を行うアプリケーションのURL
- ▶ 外部プログラムの cgi や、モジュールの php などのURLを指定

## ▶ エンコーディング

- ▶ 送信するデータの形式を指定(指定しない場合はurlencoded)
  - ▶ application/x-www-form-urlencoded
  - ▶ multipart/form-data
  - ▶ text/plain

## ▶ フォームの中身

- ▶ input 要素を主に用いて入力フォームを指定

# form の例

```
<form method="post" action="order.cgi">
  <div>
    <label>氏名: <input type="text" size="15" name="name"></label><br>
    <label>電話番号: <input type="text" size="10" name="tel"></label><br>
    <label>E-mail: <input type="text" size="20" name="mail"></label><br>
    <fieldset>
      <legend>大きさ:</legend>
      <label><input type="radio" name="size" value="large"> 大</label>
      <label><input type="radio" name="size" value="medium"> 中</label>
      <label><input type="radio" name="size" value="small"> 小</label>
    </fieldset><br>
    <fieldset>
      <legend>トッピング:</legend>
      <label><input type="checkbox" name="topping" value="bacon"> ベーコン</label>
      <label><input type="checkbox" name="topping" value="cheese"> チーズ増量</label>
      <label><input type="checkbox" name="topping" value="onion"> オニオン</label>
    </fieldset><br>
    <input type="submit" value="注文する">
    <input type="hidden" name="user" value="12345">
  </div>
</form>
```

テキスト入力

ラジオボタン

チェックボックス

送信ボタン

隠し値

# input type="text"

ラベル:

```
<label>ラベル  
  <input type="text" size="文字数" name="名称">  
</label>
```

- ▶ テキスト入力コントロール
  - ▶ ユーザが文字列を入力できるように箱を表示
  - ▶ `<label>ラベル </label>`
    - ▶ ユーザに何を入力する箱であることを示す
  - ▶ `size="文字数"`
    - ▶ 入力の箱の大きさを文字数で指定
    - ▶ CSSによる幅指定の方が正確
  - ▶ `name="名称"`
    - ▶ formデータとしての値の名前
  - ▶ 複数行のテキスト入力の場合には `textarea` を利用  
`<textarea cols="列数" rows="行数" name="名称">`  
`</textarea>`

# input type="radio"

ラベル  選択1  選択2  選択3

```
<fieldset>
  <legend>ラベル</legend>
  <label><input type="radio" name="名称" value="値1"> 選択1</label>
  <label><input type="radio" name="名称" value="値2"> 選択2</label>
  <label><input type="radio" name="名称" value="値3"> 選択3</label>
</fieldset>
```

## ▶ 一つだけ選択させる

- ▶ `<fieldset>` `</fieldset>`
  - ▶ 同じ選択のボタンをグループする
- ▶ `<legend>`ラベル`</legend>`
  - ▶ ユーザに何を選択させるものなのかを示す
- ▶ `name="名称"`
  - ▶ formデータとしての値の名前
  - ▶ 同じボタンのグループは同じ名称
- ▶ `value="値"`
  - ▶ 選択したときに送られる値
- ▶ `<label>`選択`</label>`
  - ▶ 選択するものの名前

# input type="checkbox"

ラベル  選択1  選択2  選択3

```
<fieldset>
  <legend>ラベル</legend>
  <label><input type="checkbox" name="名称" value="値1"> 選択1</label>
  <label><input type="checkbox" name="名称" value="値2"> 選択2</label>
  <label><input type="checkbox" name="名称" value="値3"> 選択3</label>
</fieldset>
```

## ▶ 複数選択可能

- ▶ `<fieldset>` `</fieldset>`
  - ▶ 同じ選択のボタンをグループする
- ▶ `<legend>`ラベル`</legend>`
  - ▶ ユーザに何を選択させるものなのかを示す
- ▶ `name="名称"`
  - ▶ formデータとしての値の名前
  - ▶ 同じボタンのグループは同じ名称
- ▶ `value="値"`
  - ▶ 選択したときに送られる値
- ▶ `<label>`選択`</label>`
  - ▶ 選択するものの名前

# input type="submit"



```
<input type="submit" value="名前">
```

- ▶ フォームで入力した内容をURLで指定されたアプリケーションに送信する
  - ▶ `value="名前"`
    - ▶ ボタンに表示される名前
- ▶ 入力した内容を取り消す `type="reset"` リセットボタンも存在する
  - ▶ フォームのすべての入力を取り消す
  - ▶ デフォルト値に戻す
  - ▶ 送信を取り消すわけではない.

# select

```
<label for="ID">ラベル </label>
<select name="名称" id="ID">
  <option value="値1">選択1</option>
  <option value="値2">選択2</option>
  <option value="値3">選択3</option>
</select>
```

ラベル	選択1 v
	選択1
	選択2
	選択3

## ▶ メニューで選択

- ▶ `<label for="ID">ラベル </label>`
  - ▶ ユーザに何を選択させるものなのかを示す
  - ▶ 対応するselectのid属性を指定
- ▶ `id="ID"`
  - ▶ labelに対応させるためのid属性
- ▶ `name="名称"`
  - ▶ formデータとしての値の名前
- ▶ `<option value="値">選択</option>`
  - ▶ 選択の名前
  - ▶ 選択した時に送られる値

# フォーム送信データ

---

- ▶ デフォルトでは `application/x-www-form-urlencoded` 形式となる
  - ▶ URLとしても大丈夫な文字列にして送信
  - ▶ GETの場合には, 実際にURLとして保存可能

名称1=値1&名称2=値2&名称3=値3&……

- ▶ `urlencode`
  - ▶ スペースは「+」にする
  - ▶ 数字とアルファベットと「\*-. \_」についてはそのまま
  - ▶ それ以外は文字コードを1バイトごとに16進数2桁として「%」を前につけて表す
- ▶ ファイルを送信するなどの場合には`urlencoded`は利用できない
  - ▶ `multipart/form-data` を利用する

# フォームデータの処理

---

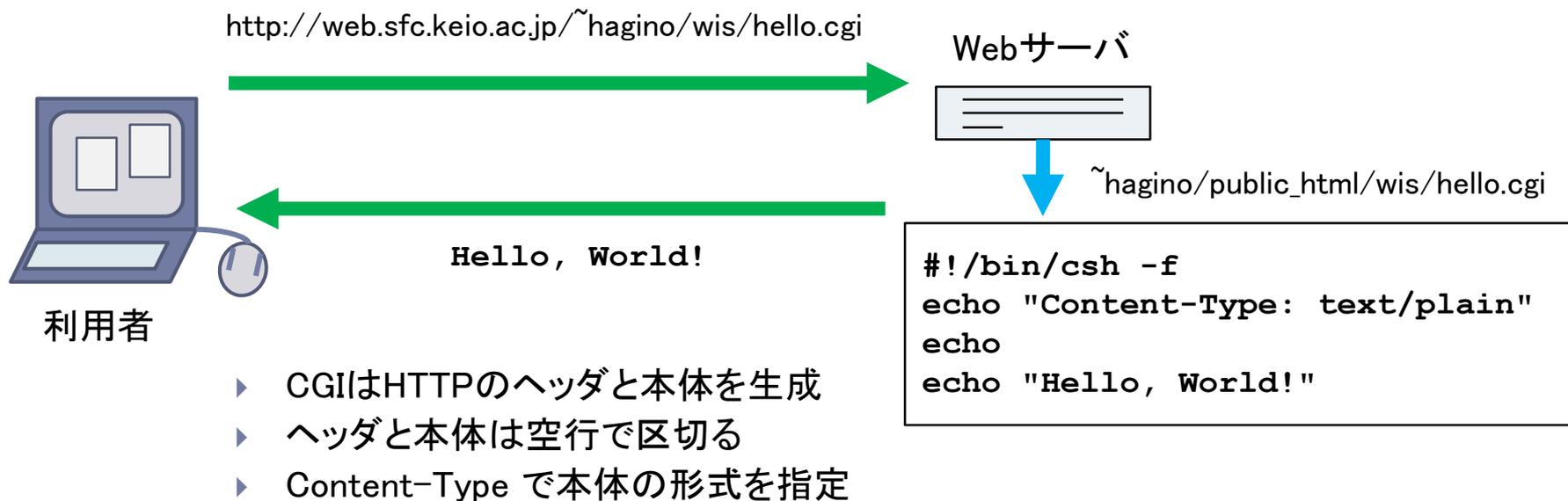
- ▶ Webサーバ内のモジュールで処理
  - ▶ Apacheサーバの場合, perlやphpのモジュールがある
    - ▶ サーバ内から, perlやphpにフォームデータが渡され処理される
  - ▶ JSP (Java Server Page) の場合には, 特別なページとして記述
- ▶ 外部CGIに処理を渡す
  - ▶ CGI (Common Gateway Interface) は外部の実行可能プログラム
  - ▶ フォームのデータが送られてくると CGI を起動する
  - ▶ フォームのデータを CGI に送る
    - ▶ GETの場合には環境変数 QUERY\_STRING に設定される
    - ▶ POSTの場合には CGI の標準入力に渡される

	モジュール	CGI
良い点	軽い	CGIプログラムの権限が指定可能
注意点	プログラムがWebサーバの権限のまま	重い

# CGIの例

## ▶ CGIのプログラミング言語

- ▶ サーバ上で実行可能であればプログラミング言語の制約はない
- ▶ シェルスクリプトでも構わない
- ▶ デフォルトの拡張子は .cgi



# フォームのリプライ

ピザの注文フォーム

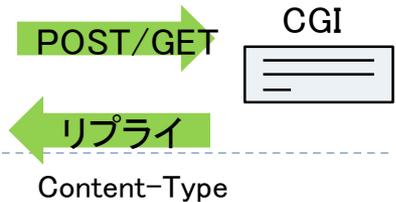
氏名:

電話番号:

E-mail:

大きさ:  大  中  小

トッピング:  ベーコン  
 チーズ増量  
 オニオン



## ▶ フォームのデータ

- ▶ HTTPのPOSTあるいはGETメソッドでCGIに送られる

## ▶ リプライ

- ▶ Content-Type で指定することで、いろいろな形式で返信可能
- ▶ Webのインタラクションとしては、フォームのリプライもHTMLであることが望ましい
- ▶ フォームの処理はプログラムで行われる
- ▶ プログラムでHTMLを生成する必要がある
  - ▶ 静的なHTMLと動的なデータを混ぜる

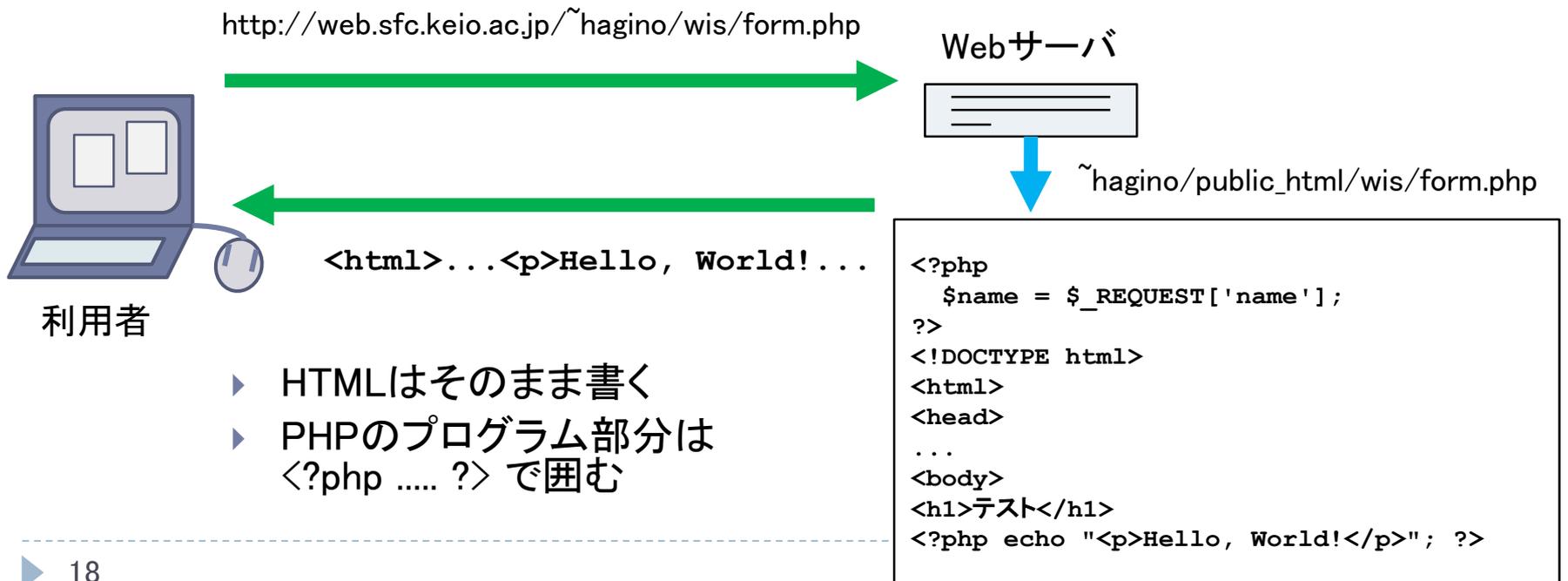
## ▶ JSP, PHP

- ▶ プログラムとHTMLを混在させることが可能

# PHP

## ▶ PHP: Hypertext Preprocessor の略

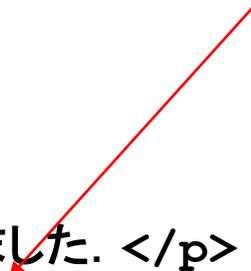
- ▶ Personal Home Page Toolsを起源
- ▶ サーバサイドスクリプト言語の一つ
- ▶ HTMLとプログラムを混在可能
- ▶ 拡張子を php とすると, PHPのプログラムとして実行し, その結果をブラウザに返す
- ▶ フォームのデータは大域変数 \$\_REQUEST に設定される



# ピザの注文を受け取るPHPの例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>注文確認</title>
  </head>
  <body>
    <h1>注文確認</h1>
    <p>ピザの注文ありがとうございました. </p>
    <p>氏名: <?=$_REQUEST['name']?><br>
      電話番号: <?=$_REQUEST['tel']?><br>
      電子メール: <?=$_REQUEST['mail']?><br>
      大きさ: <?=$_REQUEST['size']?>
    </p>
  </body>
</html>
```

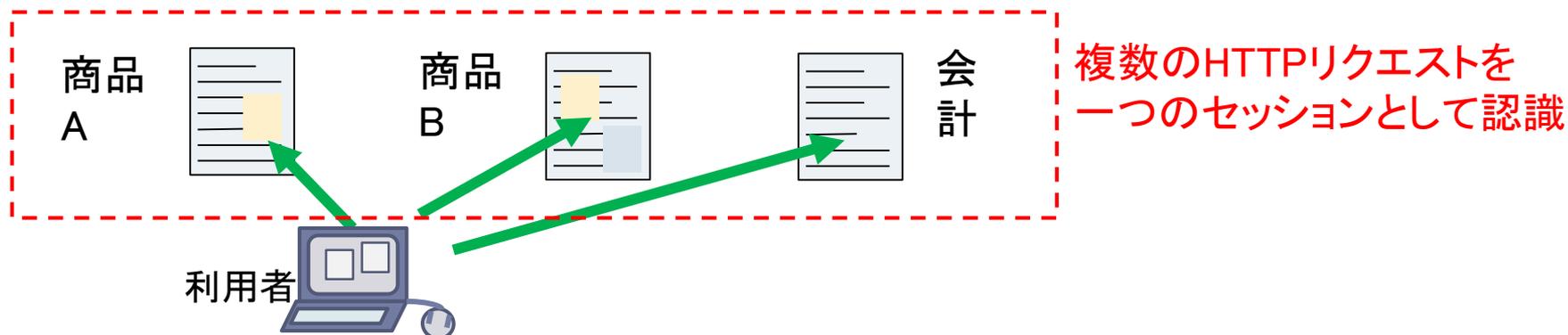
phpの式の値を挿入する



# セッション

## ▶ セッション

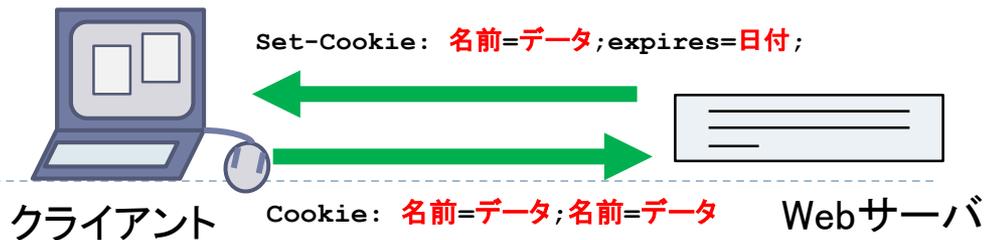
- ▶ それぞれのHTTPリクエストは独立している
- ▶ HTTPサーバはステートレスである
- ▶ 同じユーザからの一連のリクエストであることを認識する必要がある
- ▶ 複数ユーザが同時に利用していることを忘れないように



## ▶ セッションの実装

- ▶ Cookie を用いる
  - ▶ サーバからブラウザに値を記憶させる
  - ▶ 以降のリクエストでは記憶した値を付けて送信する
- ▶ セッションを識別するためのデータをHTMLに埋め込む
  - ▶ `<input type="hidden" name="user" value="1234">`

# Cookie の仕組み



## ▶ Cookieの設定

- ▶ HTTPリプライの Set-Cookie ヘッダを使う

```
Set-Cookie: 名前=データ; expires=日付;
```

- ▶ 与えられた名前でデータをブラウザが記憶
- ▶ 有効期限を設定(過去を設定することで消去することができる)
- ▶ ドメインやパスを指定することも可能

## ▶ Cookieの取得

- ▶ HTTPリクエストで送られる
- ▶ CGIの場合には, 環境変数の HTTP\_COOKIE に「;」区切りで与えられる

```
名前=データ; 名前=データ; ...
```

- ▶ PHPの場合には, `$_COOKIE[名前]` 変数に設定される

# オンラインショッピングでのCookieの利用

---

- ▶ 商品のページも静的ではなく、CGIやPHPなどの動的なもので作成する
- ▶ 訪問した時にCookieが設定されているか調べる
  - ▶ Cookieがない場合(あるいは想定外の場合)
    - ▶ 利用者のセッションのためのユニークなIDを生成しCookieとして設定するようHTTPリプライを送る
- ▶ セッションのユニークIDを使って、買い物かごなどを管理する
  - ▶ 利用者登録している場合には、利用者のデータと結びつけても良い
  - ▶ Cookieに利用者名やパスワードを設定してはいけない(セキュリティ的に問題となる)

# 課題：フォームの作成

- ▶ 自分の好きな海外の街を紹介において、見た人からいろいろな質問や問い合わせを受け取るフォームを設置しなさい。
  - ▶ フォームはトップページにあっても、質問受付専用のページでもかまいません。
  - ▶ フォームのデータを受け取るCGIあるいはPHPを作成しなさい。
- ▶ 提出
  - ▶ 問い合わせフォームのURLを提出してください

問い合わせ

氏名:

E-mail:

問い合わせ内容:



問い合わせの確認

下記の問い合わせを受け付けました。

氏名: 慶應太郎

E-mail: keio@keio.jp

問い合わせ内容:  
おすすめのレストランを教えてください。

# まとめ

---

## ▶ フォームインタラクション

- ▶ form
- ▶ input
- ▶ HTTP: POST, GET

## ▶ フォーム処理

- ▶ CGI
- ▶ PHP

## ▶ セッション