4. PHP

1

Spracovanie formulárov

Obsah

- Spracovanie formulárov
- Kontrola vstupov
- Reťazcové funkcie

Funkcia phpinfo()

- Informácie o PHP a webovom serveri
- **PHP Variables** (tzv. super-globálne premenné, **nepoužívame** pri nich príkaz **global**)
  - **_SERVER** (DOCUMENT_ROOT, PHP_SELF, REQUEST_URI, ...)
  - **_POST**
  - **_GET**
  - **_SESSION**
  - Pristupujeme k nim ako **$_SERVER**, **$_POST**, **$_GET**, **$_SESSION** (záleží na veľkosti písmen)

ukážka: `info.php`

Možnosti spracovania formulára

- Zdrojový formulár je v jednom súbore. Skript, ktorý spracováva formulár je v druhom súbore.
- **Zdrojový formulár a skript, ktorý spracováva formulár, sú v jednom súbore.**
  - **Jedna z výhod**: pri neúplných zadaných údajoch môžeme znovu zobraziť formulár už s vyplnenými údajmi a nemusíme si ich nejako posielať ani nikde ukladať.

Odosielanie formulárov (1)

- Metódy odosielania formulára: **GET** a **POST**
- Po odoslaní máme dostupné **super-globálne** premenné (polia) **$_GET**, **$_POST** (netreba používať príkaz global).
- Do polí $_GET, $_POST sa prenesú prvky formulára (väčšinou len vyplnené). Pristupujeme k nim ako **$_POST["názov_prvku“]**. Názov prvku formulára je atribút **name** nie **id**.
- Názov prvku formulára (**atribút name**) nemôže začínať číslicou, nemôže obsahovať medzeru ani diakritiku.

Odosielanie formulárov (2)

- Formulár sa odošle len vtedy, ak je definovaný element <form>.
- **Ak zadáme atribút action**, uvedieme v ňom názov súboru, ktorý má spracovať odoslaný formulár.
- **Ak nezadáme atribút action**, odoslaný formulár bude spracovávať súbor, v ktorom je formulár.
- Formulár môžeme odoslať len tlačidlom, ktoré je typu submit.
- Pri metóde **GET** sa odoslané údaje prenesú do adresy – má isté obmedzenia (nefunguje upload).
- Metóda **POST** je bezpečnejšia, môžu sa prenášať väčšie objemy údajov.
- Do udalostí onchange, onclick, onsubmit atď. nemôžeme dávať PHP funkcie, ale len JavaScript funkcie.

Objednávka TWD tour – ako by mala fungovať

- Pri prvom načítaní stránky sa zobrazí prázdny formulár.
- Po odoslaní formulára:
  - **Ak nebudú zadané všetky povinné údaje (meno, počet dospelých, destinácia)**, znova sa zobrazí formulár a používateľ musí doplniť chýbajúce údaje. **Vo formulári sa zobrazia predchádzajúce údaje.**
  - Ak budú zadané všetky povinné údaje, objednávka sa spracuje. Formulár sa už nezobrazí.

Kontrola položiek

- Hodnoty položiek, napr. **$_POST["meno"]** a pod.
- **Čo sa naozaj odosiela?**
- Či sa naozaj odošle/existuje v poli $_POST zistíme pomocou funkcie **isset()**,
  napr. if (**isset($_POST["meno"])**)
- Napr. radiobuttony a checkboxy sa neodosielajú, ak nie sú označené.
- Ak pracujeme s neexistujúcou premennou/prvkom poľa, PHP generuje upozornenie (Notice), ktoré sa môže zobraziť na stránke (závisí od nastavenia PHP).

ukážka: `ukazka-01/objednavka.php`

Čo sa deje s aplikáciou?

- Formulár sa zobrazuje vždy ☹
- Pri nekompletnom formulári sa nezobrazujú pôvodne zadané údaje.

- **Zmena č. 1**: Formulár zobrazíme len vtedy, ak nie sú kompletne zadané údaje – **pridáme ELSE vetvu**, do ktorej vložíme celý formulár + upozornenie o vyplnení všetkých údajov.

- **Zmena č. 2**: Zobrazenie odoslaných údajov vo formulári =>

ukážka: `ukazka-02/objednavka.php`

Zobrazenie odoslaných údajov (1)

- Ako zabezpečiť zobrazenie odoslaných údajov ➔ nastavenie tzv. default hodnôt pre HTML elementy
- Pre rôzne typy prvkov iný spôsob (napr. atribúty value, checked, selected...)
- Musíme zistiť, čo sa odoslalo a podľa HTML vygenerovať správny kód
- <input type="checkbox" name="poistenie" id="poistenie" value="ano"
  <?php if (isset($_POST['poistenie']) && ($_POST['poistenie'] == 'ano'))
  echo ' checked'; ?>>

Zobrazenie odoslaných údajov (2)

- <input name="meno" type="text" **value="<?php if (isset($_POST["meno"])) echo $_POST["meno"]; ?>">**

- if (**isset($_POST['pocet_dospelych'])**)
  vypis_options(0, 5, **$_POST['pocet_dospelych']**);
  else
  vypis_options(0, 5, **0**);

ukážka: `ukazka-03/objednavka.php`

Položky s viacerými hodnotami (1)

- Ak položka môže mať viacero hodnôt, treba ju spracovávať ako pole, napr. <select multiple> alebo skupina checkboxov s rovnakým menom.
- PHP môžeme "nanútiť" typ pole ➔ položku nazveme napr. name="**priplatok[]**".

- Dočasne zmeníme príplatky na skupinu checkboxov => rovnaký názov (priplatok[]), ale rôzne hodnoty (value).

Položky s viacerými hodnotami (2)

- Využijeme funkciu **in_array(co, kde)** – vráti true, ak sa hodnota **co** nachádza v poli **kde**.
- <input type="checkbox" name="**priplatok[]**" id="poistenie" **value="poistenie"** <?php if (isset($_POST['priplatok']) && **(in_array("poistenie", $_POST["priplatok"]))**) echo ' checked'; ?>>

- Pri spracovaní (výpise) môžeme využiť funkcie na poliach, napr. **implode()**.

ukážka: `ukazka-04/objednavka.php`

Kontrola vstupov od používateľa (1)

- **Nesmieme dôverovať vstupom od používateľa!**
- Aj položky, ktoré používateľ nezadáva, ale vyberá, musíme kontrolovať.
- Nesmieme sa spoliehať na obmedzenia, ktoré sú vo formulári, napr. max. počet zadávaných znakov.
- Ak bude chcieť útočník napadnúť našu aplikáciu, vytvorí si kópiu nášho formulára, kde namiesto výberových prvkov vloží napr. textové polia (aby mohol zadávať príkazy). [**pozri ukážku 05-hack**, do *meno a priezvisko* vložiť obsah súboru zly_kod-local.txt, resp. zly_kod.txt] – treba skúšať v IE alebo Mozilla Firefox.
- Dôsledky našej nedostatočnej kontroly závisia už len od zabezpečenia servera.

ukážka: `ukazka-05-hack/objednavka.php`

Kontrola vstupov od používateľa (2)

- **strip_tags(string [, povol_elem])** – odstráni z reťazca HTML elementy (okrem povolených v druhom parametri, napr. **strip_tags($meno, '');**).
- **htmlspecialchars(string)** – prevedie všetky HTML elementy na entity (napr. **<**), čo spôsobí, že zadané elementy stratia svoju funkciu a vypíšu sa len ako znaky (napr. **<**).
- **addslashes()** – pred špeciálne znaky (**‘ " \ %** atď.) vloží spätné lomítko, čím znefunkční ich pôvodný význam a stanú sa z nich len bežné znaky.

- Kontrola, kontrola, kontrola

ukážka: `ukazka-05/objednavka.php`

Kódovanie adresy pri metóde GET

- Všetky údaje sa prenášajú cez adresu
- Kódovanie adresy a údajov
- Vždy dvojica **názov=hodnota**
- Oddeľovače medzi položkami/dvojicami: **&**
- Medzery v hodnotách ➔ **+**
- Špeciálne znaky ➔ **%XY** (hexa)

- Ak vytvoríme obyčajný odkaz na súbor v tvare **info.php?id=3**, tak k položke **id** môžeme pristupovať cez **$_GET['id']** aj keď sme neodoslali žiadny formulár.

ukážka: `ukazka-06/objednavka.php`

Reťazcové funkcie (1)

názov funkcie	popis
strlen(ret)	Vráti dĺžku reťazca
substr(ret, zac [, pocet])	Vráti podreťazec dĺžky **pocet** od znaku na pozícii **zac**.
strtolower(ret)	Vráti reťazec prevedený na malé písmená
strtoupper(ret)	Vráti reťazec prevedený na veľké písmená
strrev(ret)	Vráti prevrátený reťazec
trim(ret), ltrim(ret), rtrim(ret)	Odstráni tzv. whitespace znaky (medzery, tabulátory...) z príslušného konca
chr(ascii-kod)	Vráti znak zodpovedajúci ascii-kódu
ord(retazec)	Vráti ASCII kód prvého znaku reťazca
nl2br()	Prevedie konce riadkov (\n) na

Reťazcové funkcie (2)

názov funkcie	popis
strpos(kde, čo [, zac])	Vráti prvú pozíciu **čo** v reťazci **kde** zľava alebo FALSE
strrpos(kde, čo [, zac])	Vráti prvú pozíciu **čo** (len 1 znak) v reťazci **kde** zprava, alebo FALSE
strrchr(kde, čo)	Vráti podreťazec od pozície **čo** v reťazci **kde** zprava do konca reťazca, alebo FALSE. Vyhľadáva len znaky.
strstr(kde, čo)	Vráti podreťazec od pozície **čo** v reťazci **kde** zľava do konca reťazca, alebo FALSE.
str_replace(čo, čím, kde)	Nahradí všetky výskyty **čo** reťazcom **čím** v **kde**

Reťazce – príklady

predpokladáme: $vstup = "PHP v príkladoch";

príkaz	vypíše sa
substr($vstup, 4)	v príkladoch
substr($vstup, 2, 5)	P v p
substr($vstup, -3)	och
substr($vstup, 4, -3)	v príklad
substr($vstup, -7, 4)	klad

ukážka: `retazce.php`

Využitie funkcií na reťazcoch v objednávke

- Vytvoríme funkciu **spravne_meno($m)**, ktorá skontroluje správnosť zadaného mena. Vráti TRUE alebo FALSE.

```
function spravne_meno($m) {
   $m = addslashes(strip_tags(trim($m)));
   return (strlen($m) >= 3) && (strlen($m) <= 30);
}
```

Vo funkcii nemá zmysel robiť isset() na parameter, teda isset($m) nemá zmysel, lebo je vždy TRUE.

ukážka: `ukazka-07/funkcie.php`, `objednavka.php`

Ďakujem za pozornosť ☺

Priestor na vaše otázky