Предаване на заявки към сървъра и обработка на заявките с помощта на PHP

1. Взаимодействието между клиента и сървъра става посредством протокол HTTP

Известно е, че взаимодействието между клиента и сървъра в архитектурата клиент-сървър става посредством специални протоколи за комуникация, на различни нива.

За нас интерес представлява само протокол HTTP (Hyper Text Transfer Protokol), (това е протокол на потребителско ниво) понеже за решаване на нашите програмистки задачи е достатъчен само той. Неговата асиметричност именно обуславя точно определените роли на клиента (http-клиента) и на сървъра (под сървър се има предвид web-сървър, наричан още сървър WWW или http-сървър) в архитектурата клиент-сървър.

Web-сървърът реално в нашия случай е програма, чиято основна роля е да получава и изпълнява заявки от множество клиенти. За целта, той непрекъснато проверява («слуша») определен порт (80-ти по подразбиране) за наличие на заявка. В случай че такава се появи, web-сървърът веднага я изпълнява или я добавя в опашката на чакащите за изпълнение задачи (в случай че е зает да изпълнява друга задача. Известни web-сървъри: сървър Apache на група Apache, Internet Information Server (IIS) на компания Microsoft, SunOne на фирма Sun Microsystems, WebLogic на фирма BEA Systems, IAS (Inprise Application Server) на фирма Borland, WebSphere на фирма IBM, OAS (Oracle Application Server).

От своя страна http-клиентите са Интернет браузъри (например, Internet Explorer, Opera или Mozilla), тоест приложения чрез които потребителите осъществяват достъп до ресурсите, които се менажират от web-сървъра (http-сървъра). Ролята на http клиента (например Internet Explorer или друг браузър) е да формира и изпрати клиентската заявка (в съответствие с протокола HTTP) до web-сървъра, и впоследствие да получи, анализира и представи по подходящ начин на потребителя отговора, върнат от сървъра.

2. Методи, предоставени от протокол HTTP за предаване на заявки към сървъра и формат на тези заявки:

Протоколът HTTP реализира принципа заявка/отговор. Той предоставя набор от методи (9 на брой, но от тях най-важни са **GET или POST**), които указват целите на клиентската заявки към сървъра.

Ако в адресната лента на браузъра потребителят въведе:
http://www.uni-varna.bg/students/index.php, то на практика той (http-клиентът) инициира взаимодействието със сървъра, като му изпраща запитване (заявка), в следния **пълен формат (пълна заявка)**:

- HTTP метод (GET, POST и др.);
- URL адрес ; (/students/index.php)
- HTTP версия; (например HTTP 1.1)
- Съобщение, състоящо се от:
  - **хедъри (заглавия) -** съдържат информация за типа на предаваните данни, информация за клиента, изпращащ запитването (например User-Agent: тип на браузъра, версия на ОС и др.) и
  - **тяло** (не задължителен елемент, например GET – няма тяло, но POST има тяло).

И така, конкретно за въведен от потребителя стринг в адресната лента на браузъра, например http://www.uni-varna.bg/students/index.php, браузърът ще генерира заявка към сървъра от вида:

Get /students/index.php HTTP 1.1
Host: www.uni-varna.bg
User-Agent: Mozilla/5.0 (Windows…
…
Accept-language:
...

3. **Отговорът на сървъра:**

Отговорът на сървъра съдържа:

- Стринг за състояние, в който влиза версията на протокола и код за резултат (успех или грешка);
- Съобщение с информация за сървъра, метаинформация (тоест информация за съдържание на съобщението) и тяло на съобщението.

Отговорът от сървъра на конкретната заявка по въведеното от потребителя:
http://www.uni-varna.bg/students/index.php,
е от вида:
HTTP/ 1.1 200 OK,
Date: Sun, 31 Oct 2011 ...
Server: Apache
…
Content-Type: text/html
<html>…</html>
Тоест, отговорът съдържа: HTTP/ <версия><код на резултата><описание на резултата>
<заглавие 1>
...
<тяло на отговора>

Код на резултата : от 200 до 299 означава успешно изпълнение
Заглавните редове в отговора съобщават на браузъра кой е сървъра, какъв софтуер има инсталиран на него и др.
Най-важният хедър е Content-Type, указващ какъв е типа на съдържанието в тялото на отговора
Тялото на отговора най-често съдържа изисквания ресурс – статичен файл, динамично генериран HTML код, изпълним файл и др.

4. **Обработка на заявки с помощта на PHP.**

За извличане на стойността на променливите, предавани чрез HTTP-заявка се използват суперглобални масиви $_POST (например $_POST['firstName']) и $_GET (например $_GET['firstName']). От PHP 4.1.0 е взето и решение, за обръщение към променливите, предавани чрез HTTP-заявка, да се използва специален супер глобален асоциативен масив – ***$_REQUEST*** (например $_REQUEST['firstName']). Този масив съдържа данни, предавани чрез методите *POST* и *GET*, а също и с помощта на *HTTP* cookies. Това е специален суперглобален асоциативен масив, чиито стойности могат да се получат на всяко място в програмата, използвайки в качеството на ключ името на съответната променлива (елемент на формата).

Метод GET - извод

- **К**огато браузърът изпраща данни чрез html форма до сървъра, използвайки метода GET, попълненото съдържание във формата се добавя към URL-а на програмата, която ще обработва изпратените данни, след знак **?** във вид на двойки **име=стойност**, обединени с помощта на амперсанд **&**:

http://localhost/irena/example2.php?firstName=Violeta&lastName=Bozhikova

- Ако в полето за въвеждане въведем някакъв служебен символ, то той ще се предаде в шестнадесетичен код, например, символ $ ще се замени с %24. По такъв начин се предават и букви на кирилица. Така, ако сме въвели Виолета Божикова, в адресната лента ще видим:
  http://localhost/irena/example2.php?firstName=%D0%92%D0%B8%D0%BE%D0%BB%D0%B5%D1%82%D0%B0&lastName=%D0%91%D0%BE%D0%B6%D0%B8%D0%BA%D0%BE%D0%B2%D0%B0
- За полета в които се въвежда текст и парола (това са input елементи, с type=text и type=password), стойностите ще бъдат такива, каквито потребителя е въвел. Това е съществен недостатък при предаване с GET: всеки може да види и преправи стойностите на параметрите в URL полето на браузера. Ето защо не се препоръчва използване на този метод за предаване на информация, влияеща на безопасността на работа на програмата или *сървъра*.
- Независимо от тези недостатъци, използването на метод GET е достатъчно удобно при настройка на скриптове (тогава може да видим значенията и имената на предаваните променливи), както и за предаване на параметри, не влияещи на безопасността.

Кога е безопасно да предаваме данни с Get?

  - Ако данните ще се предават чрез кликване върху хипервръзка – то Get.
  - В случай, че резултата от предаваните данни ще се визуализира в повече от една страница и е необходима навигация между страниците, то пак може Get, тъй като навигацията може да се организира чрез предаване на параметрите по URL. Типичен пример за това са формулярите за търсене. При тях навигацията между страниците се извършва чрез хипервръзки:
  - <First | Previous | Next | Last>
  - Или чрез навигационните бутони на браузъра

- Освен това, ясно е че с Get заявки клиентът (браузърът) може да изпрати ограничено (зависи от самия браузър, например за Internet Explorer – до 2083 байта) количество данни, които се прикрепят към адреса на ресурса (вижда се от примера), тоест не се препоръчва използване на този метод за предаване на голям обем информация.
- Независимо, че и с двата метода могат да се предават данни, GET е основно метод, който се използва в web програмирането, за формиране на заявки, имащи за цел получаване от сървъра на определен ресурс, зададен чрез URL-а си, в отправената до сървъра заявка, а POST е методът чрез който се предават важни данни (например пароли) или големи количества данни, до сървъра, като указаният URL адрес указва програмата, която ги обработва.

Ресурси, извличани с GET заявки

- Ако URL указва статичен файл, то сървърът просто го извлича и го изпраща на клиента (без да го променя по някакъв начин).

- Ако URL указва изпълнима програма, то на потребителя се връща резултата от работата на програмата, а не нейния текст (ако разбира се, текста не е резултат от работата на програмата).
- Ако URL указва файл, който съдържа програмен код, който трябва да се интерпретира (например http://www.uni-varna.bg/students/index.php) и съответния интерпретатор (в случая - php интерпретатор) е инсталиран като модул на web сървъра, тогава web сървърът извлича съдържанието на файла (в случая - index.php) и го предава на php интерпретатора. Резултатите от интерпретиране на php кода се връщат на web сървъра, който от своя страна формира HTTP отговор и го изпраща на потребителя.

Метод POST - извод

- Когато браузърът изпраща данни чрез html форма до сървъра, използвайки метода POST, съдържанието на html формата се кодира точно така, както и при метод GET, **но вместо попълненото съдържание във формата да се добави като стринг към URL**, то се изпраща като блок от данни, **в тялото на съобщението** (за сравнение - Get заявките нямат тяло, а съобщението съдържа само хедъри). URL-а в атрибут ACTION на формата определя, програмата, която трябва да обработва този блок от данни.
- Метод POST е препоръчван за предаване на сървъра на **отговорна или голяма по обем информация (**пароли, пощенски съобщения, данни за добавяне към БД и др.), тъй като данните от формата се предават на сървъра чрез тялото на съобщението (ползвателя не вижда предаваните към сървъра данни в URL полето на браузъра си). Да се променят данните, предавани чрез метод *POST* може да стане единствено чрез промяна на стойностите, въведени във формата, а не в заглавието и, както при *GET*. Дължината на изпращания файл се предава в променливата на обкръжението CONTENT_LENGTH, а типа на данните – в променливата CONTENT_TYPE.
- Основно преимущество на POST заявки, спрямо GET: голямата безопасност и функционалност в сравнение с GET-заявки. Ето защо метод POST по-често се използва за предаване на важна и|или голяма по обем информация.

Метод POST и GET – служебни данни

- При изпращане на данните към сървъра по който и да е метод (GET/POST) се предават не само самите данни, въведени от потребителя, но и редица променливи на обкръжението, характеризиращи клиента, историята на неговата работа, пътища към файлове и др. Ето някои от тях:
  - ***REMOTE_ADDR*** – IP-адрес на хоста (компютъра), отправящ заявката;
  - ***REMOTE_HOST*** – име на хоста, от който е отправена заявката;
  - ***HTTP_REFERER*** – адрес на страницата, указваща текущия скрипт;
  - ***REQUEST_METHOD*** – използван метод;
  - ***QUERY_STRING*** – информация, намираща се в *URL след знака ?*;
  - ***SCRIPT_NAME*** – виртуален път към програмата, която трябва да обработи заявката;
  - ***HTTP_USER_AGENT*** – информация за браузъра, който се използва като *клиент*
- **Как да разберем стойността на някаква променлива на обкръжението?** Такава информация може да се получи с помощта на функция getenv(). Тя връща стойността на променлива, чието име се предава в качеството на параметър.

Пример: Скриптът извежда IP-адрес на потребителя, изпращащ заявката:

```
<?php
getenv('REQUEST_METHOD');
   // връща използвания метод, в случая GET
echo getenv ('REMOTE_ADDR');
   // извежда IP-адрес на потребителя,
   // изпращащ заявката
?>
```

Резултат: 127.0.0.1

Пример: Скриптът извежда стринга който се предава с GET - getenv('QUERY_STRING');
Така, ако се върнем на example2.php и добавим тази команда ще извлечем стринга, например firstName=Ivan&lastName=Ivanov (в случай, че сме въвели Ivan Ivanov във HTML формата.

```
<?php
print '<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=Windows-1251">';
print "Здравей, " . $_GET['firstName'] . ' ' . $_GET['lastName'].'!';
echo getenv('QUERY_STRING'); // firstName=Ivan&lastName=Ivanov
?>
```

example2.php с getenv('QUERY_STRING')

Пример: Скриптът извежда адреса на страницата, от която се изпращат данните - getenv('HTTP_REFERER');

```
//в случая ще се върне: http://localhost/irena/example2.html
<?php
print '<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=Windows-1251">';
print "Здравей, " . $_GET['firstName'] . ' ' . $_GET['lastName'].'!';
getenv('HTTP_REFERER'); // http://localhost/irena/example2.html
?>
```

example2.php с getenv('HTTP_REFERER');

5. **Синтаксис на една HTML-форма**. За създаването на HTML форма се използва таг form. Вътре в него се намират една или няколко input команди:

Пример:

```
<form>
 Име:
 <input type="text" name="name"/>
<br/>
 Фамилия:
 <input type="text" name="family"/>
<br/>
 E-mail:
 <input type="text" name="mail"/>
 </form>
```

Име:
Фамилия:
E-mail:

- Формулярите се указват с таг <form> за начало и таг </form> за край, а между тези два тага се разполагат всички input компоненти на формуляра, указани най-общо по следния начин:

<input type="тип" name="име" />.

- На мястото на "тип" стои конкретния тип елемент (text, например), а "име" е уникално име на елемента (name, family, mail), което позволява неговото управление при обработката на формуляра.
- С помощта на атрибутите action и тага method, на формуляра се задават име на програмата, която ще обработва данните от формата и метода за достъп (Get, Post), например <form action="1.php" method=POST>.
- Изпращането на данните от формата става след натискане на бутон – елемент input от тип submit, а отказът става чрез натискане на бутон reset, което се описва по следния начин:

<input type=submit value="Submit">
<input type=reset value="Reset">