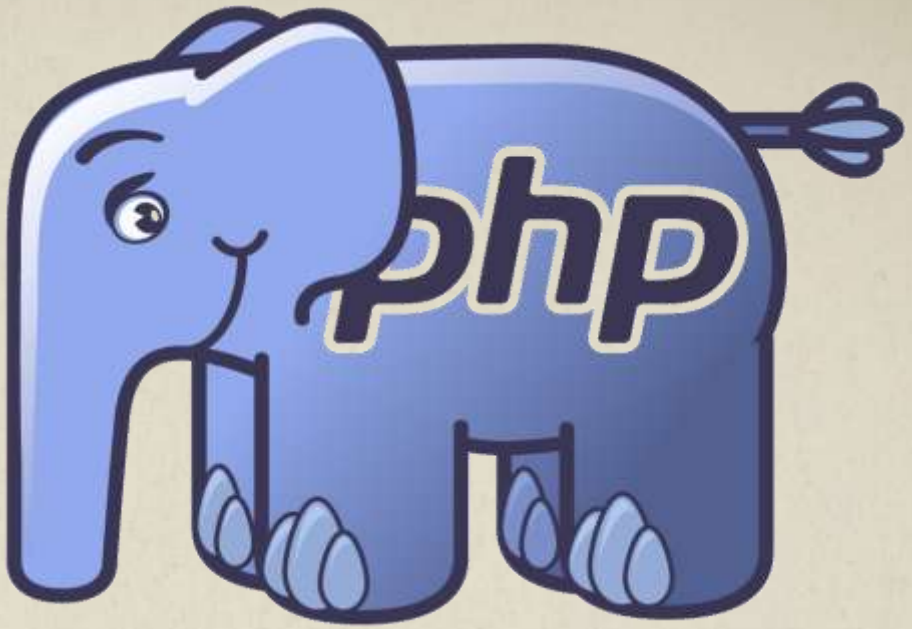
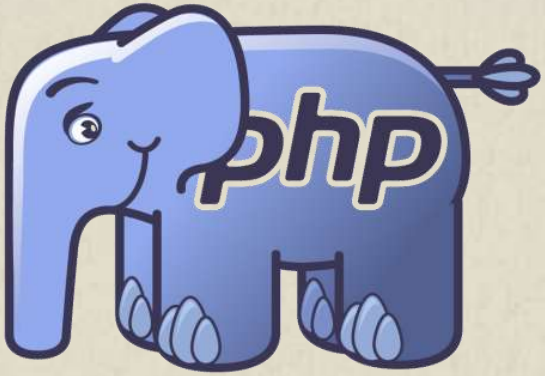


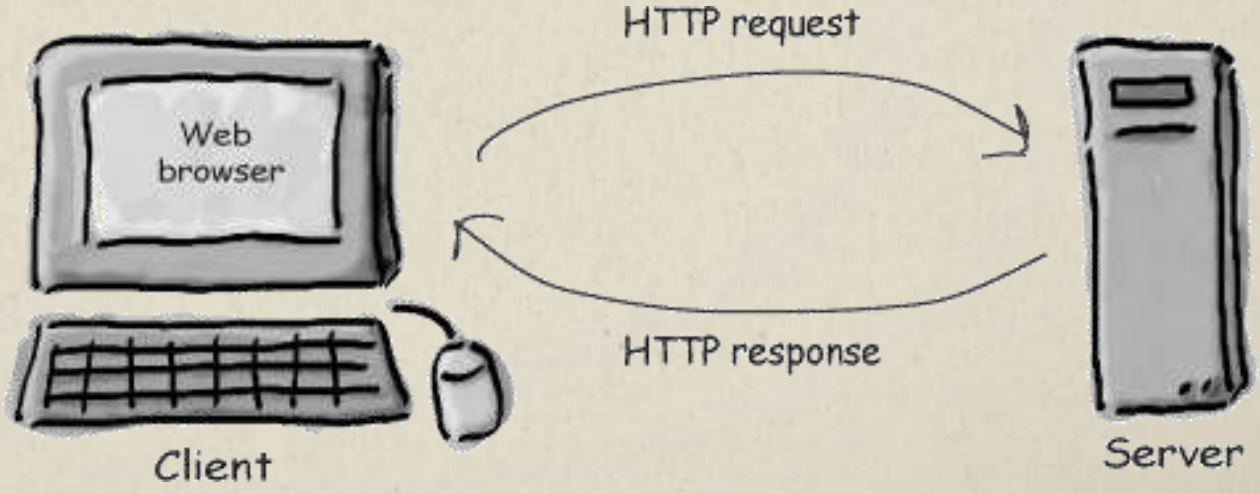
PHP: ВВЕДЕНИЕ В ПРОФЕССИЮ

УРОК 5. КУКИ. СЕССИИ. АВТОРИЗАЦИЯ.



ЕЩЕ РАЗ ПРО HTTP

ПОСМОТРИМ ВНИМАТЕЛЬНО НА HTTP



КОНЦЕПЦИЯ «КЛИЕНТ-СЕРВЕР»

«Клиент-сервер» – это, пожалуй, самое важное, что вам нужно знать об интернете ☺

СОЕДИНЕНИЕ:
Входим в магазин, видим продавца

ЗАПРОС: «Дайте, пожалуйста, 300 грамм Краковской»

СЕРВЕР работает, ищет колбасу

ОТВЕТ: «Сейчас!» (заголовок)
подаёт колбасу (тело ответа)

HTTP: HyperText Transfer Protocol

Из чего состоит запрос?

- Метод, URI, версия протокола:
  `GET / HTTP/1.1`
- Обязательный заголовок Host:
  `Host: lenta.ru`
- Другие заголовки
  `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36`
- Данные

А ответ?

- HTTP/версия, код состояния, пояснение
  `HTTP/1.1 200 OK`
- Другие заголовки:
  `Content-Type: text/html`
- **ДВА** перевода строки
- Тело ответа

Проблема: мы не можем идентифицировать клиента…
По сути разговор всегда начинается заново.

ЧТО ВАЖНО ЗНАТЬ О HTTP?

ЧТО ТАКОЕ COOKIE?

Cookie

- **В-первых** это специальный заголовок ОТВЕТА СЕРВЕРА:
  `Set-Cookie: foo=bar`
- **Во-вторых** это ХРАНИЛИЩЕ в браузере, где он обязан сохранять поступившие ему от сервера пары «ключ-значение»
  Обычно данные cookie хранятся в небольших текстовых файлах.
- В-третьих обязанность браузера высылать хранящиеся данные серверу при СЛЕДУЮЩИХ запросах с помощью заголовка:
  `Cookie: foo=bar`

ЧТО ТАКОЕ COOKIE?

HTTP: HyperText Transfer Protocol

Атрибуты

- **Ключ и значение**
- **Срок действия**
  Если нет – до закрытия браузера!
  Формат 'r' (RFC 2822), GMT
- **Путь**
- **Домен**
  Автоматически доступны и для поддоменов
- **Признак HTTPS**
  (запрет на передачу от клиента по незашифрованному соединению)
- **HTTPOnly**
  Недоступность через JavaScript

```
Set-Cookie: foo=bar; expires=Fri, 31
Dec 2010 23:59:59 GMT; path=/;
domain=.example.net
```

ПОДРОБНОСТИ О COOKIE

HTTP: HyperText Transfer Protocol

Использование cookie в PHP

- Для установки cookie (передачи заголовка клиенту) используйте функцию `setcookie()`
  - `$name`
  - `$value`
  - `$expired`
  - `$path`
  - `$domain`
  - `$secure`
  - `$httponly`

- Для чтения cookie (переданных от клиента!) используйте суперглобальный массив

  `$_COOKIE`

  Внимание: только читать! Запись в него не имеет смысла

COOKIE В PHP

НЕМНОГО О БЕЗОПАСНОСТИ

Безопасность в cookie

Вы должны отчетливо понимать, что:

- Браузер может не сохранить куку
- Браузер может не передать ее при следующем запросе
- Клиент может удалить куку из браузера
- Клиент может изменить любые данные в куке
- Куку могут украсть
- Данные cookie могут быть подделаны MitM

Вывод: в cookie нельзя хранить никакие чувствительные данные!

БЕЗОПАСНО ЛИ?

СЕССИИ В ЯЗЫКЕ PHP

$_SESSION

Сессия

Допустим, мы «пометили» клиента с помощью cookie. А что дальше? Как сохранить какие-то данные безопасно?

Сессии – встроенный в PHP способ хранения данных на сервере между последовательными запросами одного и того же клиента:

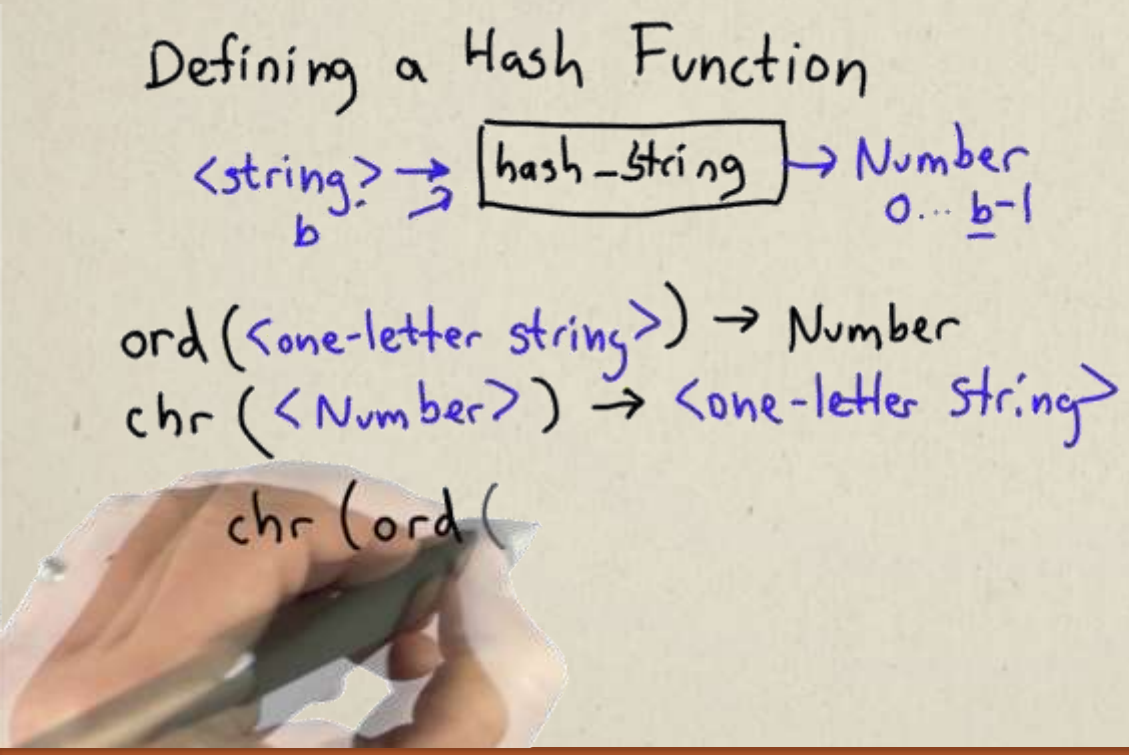
- Уникальный идентификатор сессии
- Выбор способа передачи идентификатора (cookie или get-параметр)
- Выбор хранилища данных сессий
- Удобный интерфейс для доступа к данными: массив **`$_SESSION`**

Полезные функции:

```
session_start();
session_id();
session_regenerate_id();
```

$_SESSION

НЕМНОГО ОТВЛЕЧЕМСЯ И ПОГОВОРИМ О ХЭШ-ФУНКЦИЯХ



Хэш-функции

Это «односторонние функции», позволяющие по тексту произвольной длины получить некий короткий, достаточно случайный «хэш»-идентификатор текста.

- Самый простой (и совершенно дурацкий) пример – функция вычисления длины текста
- Современные алгоритмы:
  - md5 (категорически устарел)
  - sha1 (устарел)
  - sha256, sha512
  - gost-crypto

Для чего нам может пригодиться хэш-функция?

- «Свёртка» длинных значений в короткие
- Получение псевдослучайной строки (используем время, /dev/urandom, соль)
- Одностороннее шифрование

ХЭШ-ФУНКЦИИ, СОЛЬ И САХАР ☺

Встроенные функции хэширования

Как же правильно хэшировать пароли?

Хэширование часто применяется для одностороннего шифрования пароля:

- Храним в базе пользователей не пароли, а хэши паролей
- Когда пользователь вводит пароль в форму входа – снова вычисляем хэш
- Если вычисленный и сохранённый хэши совпали – считаем проверку пароля пройденной

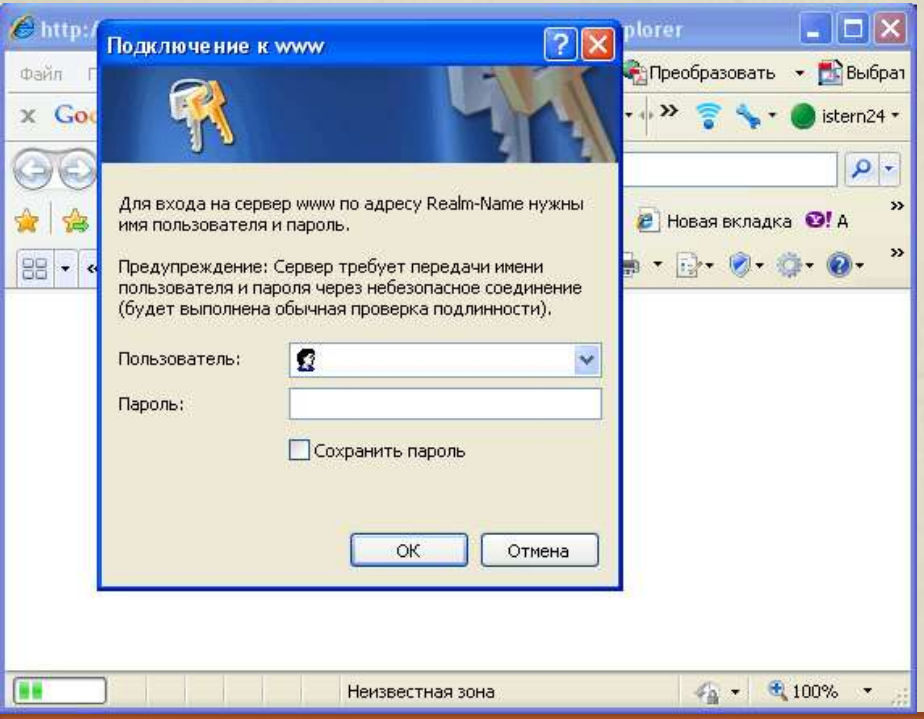
В PHP следует ВСЕГДА использовать встроенные функции

- `password_hash()`
- `password_verify()`

- они обеспечивают максимально безопасное хэширование и проверку хэша!

ХЭШ-ФУНКЦИИ, СОЛЬ И САХАР ☺

ИДЕНТИФИКАЦИЯ
АУТЕНТИФИКАЦИЯ
АВТОРИЗАЦИЯ



Идентификация

- *Кто там?*
- *Это я, Вася! Сосед из 13 квартиры!*

Аутентификация

- *Ты что, попутал? Нету никакого Васи в 13 квартире!*

или

- *Ах Вася… Я тебе щас покажу Васю… Как зовут твою жену, а?*
- *Вера…*
- *Василий, точно ты! Не узнал!*

Авторизация

- *Василий, говорят, мы тебе 1000 рублей были должны? Вот, возвращаю!*

ЕСЛИ НАУЧИТЕСЬ БЕЗ ЗАПИНКИ ВЫГОВАРИВАТЬ – ВЫ МОЛОДЕЦ!

И ЕЩЕ РАЗ О БЕЗОПАСНОСТИ!

- Аксиома некомпетентности (готовое решение лучше велосипеда)

- Логирование всего и вся

- Проактивная защита (Google Captcha например)

- Поведенческая защита (временный пароль на почту или телефон)

ДО ВСТРЕЧИ НА СЛЕДУЮЩЕМ УРОКЕ!

**ВИДЕОЗАПИСЬ, СЛАЙДЫ, ПРЕЗЕНТАЦИЯ
И ДОМАШНЕЕ ЗАДАНИЕ
БУДУТ ВЫЛОЖЕНЫ ДО 10 УТРА СЛЕДУЮЩЕГО ДНЯ**

ProfIT