

## Programação II

# Mantendo o estado das aplicações: Cookies e Sessions



# Estado das aplicações

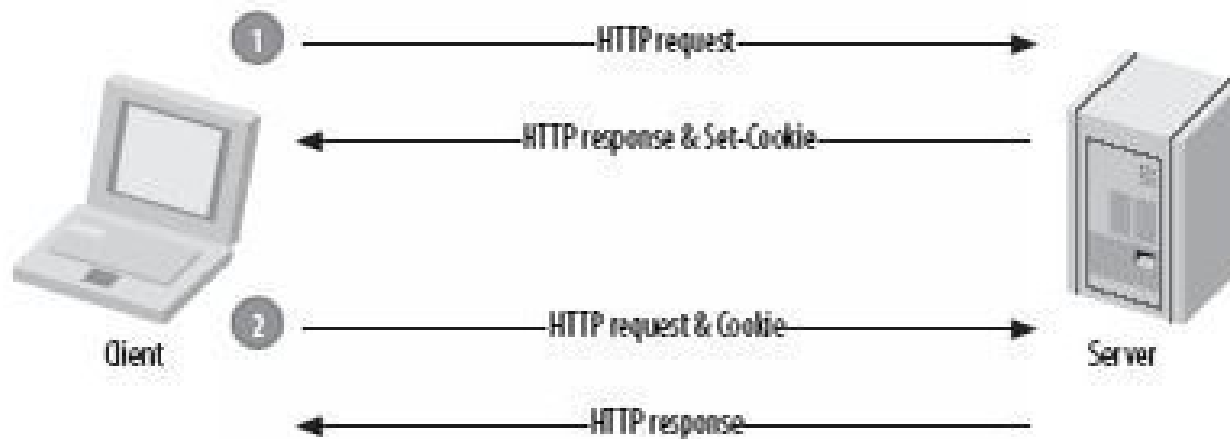
- O HTTP é um **protocolo sem estado** ou *stateless* (não tem memória)
  - as diversas requisições são tratadas de forma independente, não sendo possível identificar se duas requisições partiram do mesmo usuário
- Entretanto, há casos em que precisamos manter a memória de eventos já ocorridos e/ou valores de variáveis através das diversas requisições feitas por um usuário, ou seja, precisamos manter o estado da aplicação
  - No preenchimento de um formulário de múltiplas páginas (na terceira página o sistema ainda precisa “lembrar” dos dados digitados na primeira);
  - Num sistema de webmail, a abertura de cada mensagem gera uma nova requisição. Se a informação de que o usuário já efetuou login não fosse mantida, o mesmo precisaria repetir seu login e senha a cada mensagem aberta;
  - Num carrinho de compras em uma loja virtual, os itens devem ser mantidos quando o usuário deixa a página do carrinho para continuar comprando. Ao adicionar novos itens, os produtos adicionados anteriormente ainda devem estar no carrinho;

## Gerenciando o estado

- Em outras palavras, o servidor precisa saber com qual dos clientes (possivelmente milhares) ele está se comunicando, a fim de entregar a informação correta, considerando as interações prévias deste cliente.
- Temos duas formas de preservar informações de estado:
  - No cliente, através de **cookies**;
  - No servidor, através de **sessões**.

# Cookies

- São variáveis (pares chave-valor) enviadas pelo servidor através do protocolo HTTP para serem **armazenadas no cliente**.
- Quando o browser verifica que existem cookies para o site em questão, eles são anexados a cada nova requisição enviada ao servidor, mantendo assim uma memória entre as várias requisições.



## Cookies

- Por razões de segurança, os cookies podem ser lidos somente a partir do domínio que os gerou.
- Como fazem parte do cabeçalho HTTP, cookies só podem ser gravados antes do envio de qualquer informação para o cliente, o que inclui outros cabeçalhos, tags ou saídas php.
- Geralmente são utilizados para manter preferências de usuário entre as visitas, armazenar detalhes de login, etc.
- Os usuários podem desabilitar os cookies no navegador.

# Cookies

- Existem cookies persistentes e cookies de memória:
  - Os persistentes são aqueles gravados em arquivo, permanecem disponíveis mesmo após o browser ser fechado ou o computador ser reiniciado, possuindo data e hora pré-definidas para a expiração;
  - Os cookies de memória não são armazenados em disco, e permanecem ativos apenas enquanto a janela do browser não for fechada.
- A função `setcookie` do PHP gera o cabeçalho `HTTP_COOKIE`;
- Os cookies ficam disponíveis no vetor superglobal `$_COOKIE`.  
`$_COOKIE['nome_do_cookie'];`

## Gravando cookies

- Todas as chamadas à função setcookie devem ser feitas antes do envio de qualquer header (cabeçalho) ou texto.
- Para gravar cookies no cliente, deve ser utilizada a função setcookie, que possui a seguinte sintaxe:

```
int setcookie(string nome, string valor, int exp, string  
              path, string dominio, int secure);
```

- onde:
  - nome: nome do cookie (obrigatório);
  - valor: valor armazenado no cookie (obrigatório);
  - exp: data de expiração do cookie (opcional), no formato Unix. Se não for definida, o cookie será de memória;
  - path: path do script que gravou o cookie (opcional);
  - dominio: domínio responsável pelo cookie (opcional);
  - secure: se tiver valor 1, indica que o cookie só pode ser transmitido por uma conexão segura (https) (opcional).
- Para excluir um cookie, basta passar o valor false:
  - setcookie(nome\_do\_cookie, false);

# Exemplo de gravação e leitura de cookie persistente

```
<?php
// define um cookie chamado count, que expira depois de 1h (3600s)
if(!isset($_COOKIE["count"])){ // ainda não existe: cria com 1
    $count = 1;
    setcookie("count", $count, time()+3600);
} else { // já existe: incrementa
    $count = $_COOKIE["count"];
    $count++;
    setcookie("count", $count, time()+3600);
}

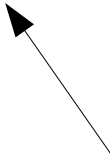
// define um cookie chamado visita, que expira depois de 1 ano,
//contendo a data atual. Na próxima visita, este será o acesso anterior
date_default_timezone_set('America/Sao_Paulo');
$umAno = time() + 60 * 60 * 24 * 365;
setcookie("visita", date("d/m/Y - G:i:s") , $umAno);
?>
```

Continua no próximo slide...



# Exemplo de gravação e leitura de cookie persistente

```
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>Exemplo do uso de cookies</title>
</head>
<body>
  <h2>
    <?php echo isset($_COOKIE["count"]) ? "Esta é sua ".
    ($_COOKIE["count"]+1)."ª visita" : "Esta é sua 1ª visita" ; ?>
  </h2>
  <p>
    <?php echo (isset($_COOKIE["visita"])) ? "Seu último acesso
    foi em: " . $_COOKIE["visita"] : " "; ?>
  </p>
</body>
</html>
```



Exercício: adicione um link “me esqueça”, apontando para uma página que exclui os cookies e redireciona de volta para esta página.

## Dica: tempo no formato Unix

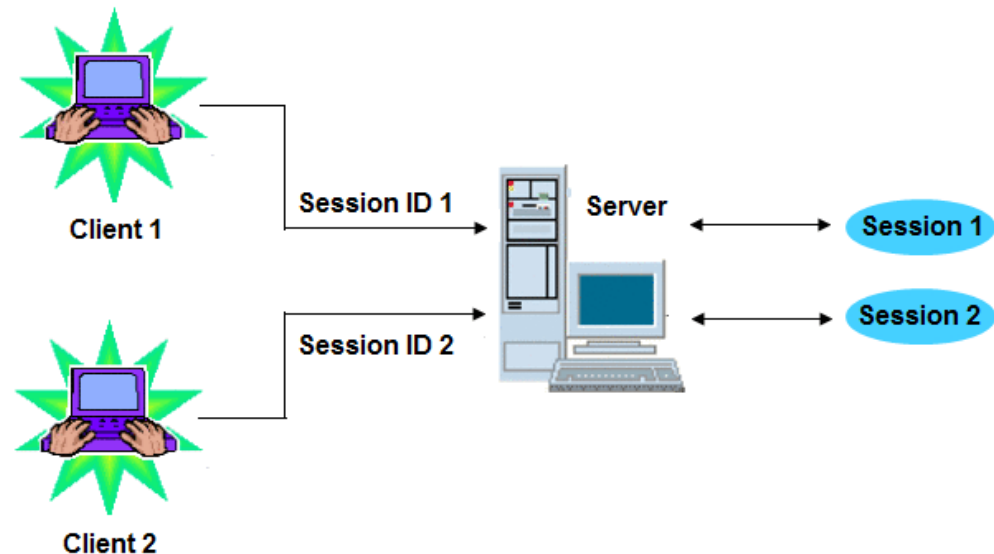
- A data no formato Unix é o número de segundos decorridos a partir de 01/01/1970, também conhecida como **timestamp** (carimbo de tempo);
- A função **time()** do PHP retorna o timestamp da data atual:

```
<?php  
echo time();  
?>
```

- O primeiro cookie do exemplo anterior expirará 1 hora após a execução da página (`time() + 3600`).

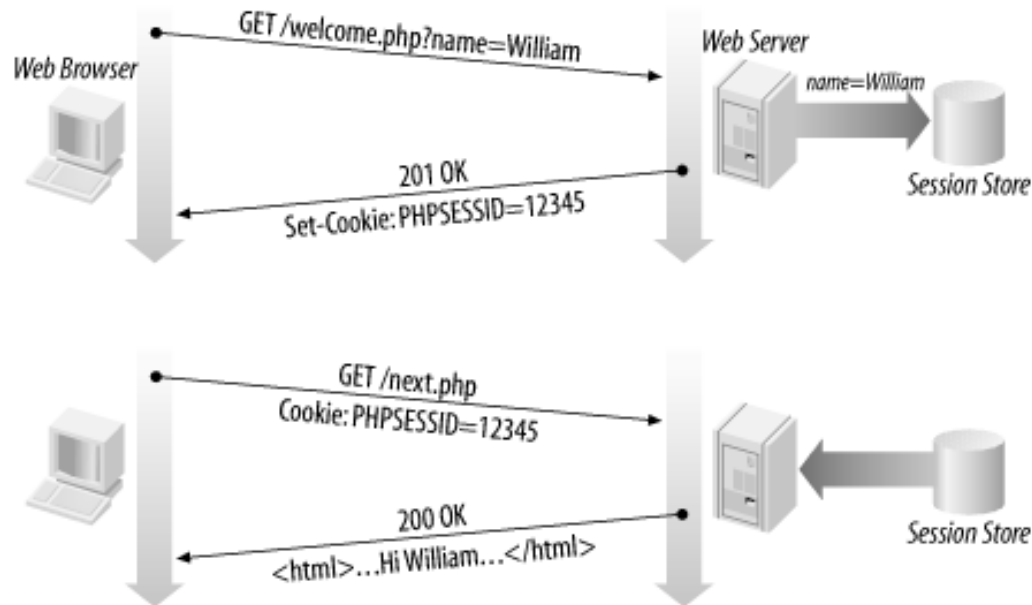
# Sessões

- Uma sessão é um local **no servidor** onde podemos armazenar dados relativos a um usuário específico e suas interações com as diversas páginas de um site. As informações adicionadas à sessão por uma página poderão ser recuperadas em outras páginas que utilizem a mesma sessão.
- A sessão permanece ativa durante um tempo pré-estabelecido ou até que o browser seja fechado.
- Ou seja, as informações das sessões são temporárias. Para mantê-las permanentemente, deve-se utilizar uma base de dados.



# Sessões

- Cada sessão recebe um identificador único (um grande número aleatório), que será a ligação com o usuário.
  - Este número distingue uma entre as várias sessões que podem estar ativas no servidor a cada instante.
  - O id da sessão, chamado **PHPSESSID**, é armazenado num cookie, ou propagado via URL entre as páginas enquanto o usuário navega.



## Sessões

- Dependendo da configuração, as sessões podem ser armazenadas em arquivos de texto ou na memória RAM do servidor
  - As sessões serão gravadas em disco se o arquivo php.ini estiver com a configuração **session.save\_handler = files**
  - Nesse caso, é necessário especificar o caminho onde os arquivos ficarão, através da opção **session.save\_path =**
  - Verifique as configurações acima com **echo phpinfo();**

## `session_start()`

- Comando que inicia uma nova sessão vazia (se ainda não existe) ou vincula a página a uma sessão existente.
- É necessário declarar `session_start()` no início de cada página onde for necessário alguma informação que esteja registrada na sessão (antes de qualquer saída ou tag HTML).
- Após `session_start`, os valores na sessão podem ser acessados através da variável superglobal **`$_SESSION`**.

```
<?php  
session_start();  
//demais comandos  
?>
```

## `session_destroy()`

- Comando utilizado para encerrar uma sessão existente (numa página de logoff, por exemplo).

```
<?php
    session_start();
    session_destroy();
?>
```

- Obs: antes de destruir a sessão, é necessário iniciá-la, caso contrário, teremos um erro.

## Registrando variáveis

- Os valores registrados nas sessões ficarão armazenados no vetor global `$_SESSION['nome']`. As demais páginas poderão acessar os valores, desde que executem `session_start()` no início de seu código.
- Para registrar uma variável, basta atribuir um valor à mesma no vetor `$_SESSION`:

```
// primeiro arquivo - registrando o valor
```

```
<?php
session_start();
$_SESSION['var'] = "Teste";
?>
```

```
// segundo arquivo - resgatando o valor
```

```
<?php
session_start();
echo $_SESSION['var']; // o retorno na tela deve ser "Teste"
?>
```



## Verificando se há uma sessão ativa

- O comando `session_start()` sempre irá iniciar uma nova sessão caso nenhuma esteja ativa. Deste modo, para verificar se de fato a sessão atual contém os valores esperados (e não é uma sessão vazia), utilizamos a função `isset`.

```
<?php
    session_start();
    if (isset($_SESSION['var'])) {
        echo "var registrada";
    } else {
        echo "var ainda não foi registrada";
    }
?>
```

## Exemplo: pagina1.php

```
<?php
//pagina1.php
session_start();
if(isset($_SESSION['views']))
    $_SESSION['views'] = $_SESSION['views'] + 1;
else
    $_SESSION['views'] = 1;

echo "Views=" . $_SESSION['views'];
?>
<p>Atualize a página para incrementar o contador!</p>
<p><a href="teste2.php">Página 2</a></p>
```

## Exemplo: pagina2.php

```
<?php
// pagina2.php
session_start();
echo "O valor gravado pela página 1 atualmente é:
". $_SESSION['views'];
?>
```

## Redirecionamento de páginas

- O PHP possui a função header para enviar cabeçalhos HTTP ao cliente.
- Esta função deve ser chamada antes de qualquer saída ser emitida (como tags, echos, linhas em branco, etc).

```
header("Location: http://www.example.com/");
```

- É uma função muito utilizada para levar o usuário automaticamente a uma nova página de acordo com os dados de entrada
  - Ex: direcionar para uma página de acesso restrito após login, ou de acesso negado, caso tente acessar algo sem passar pelo login.