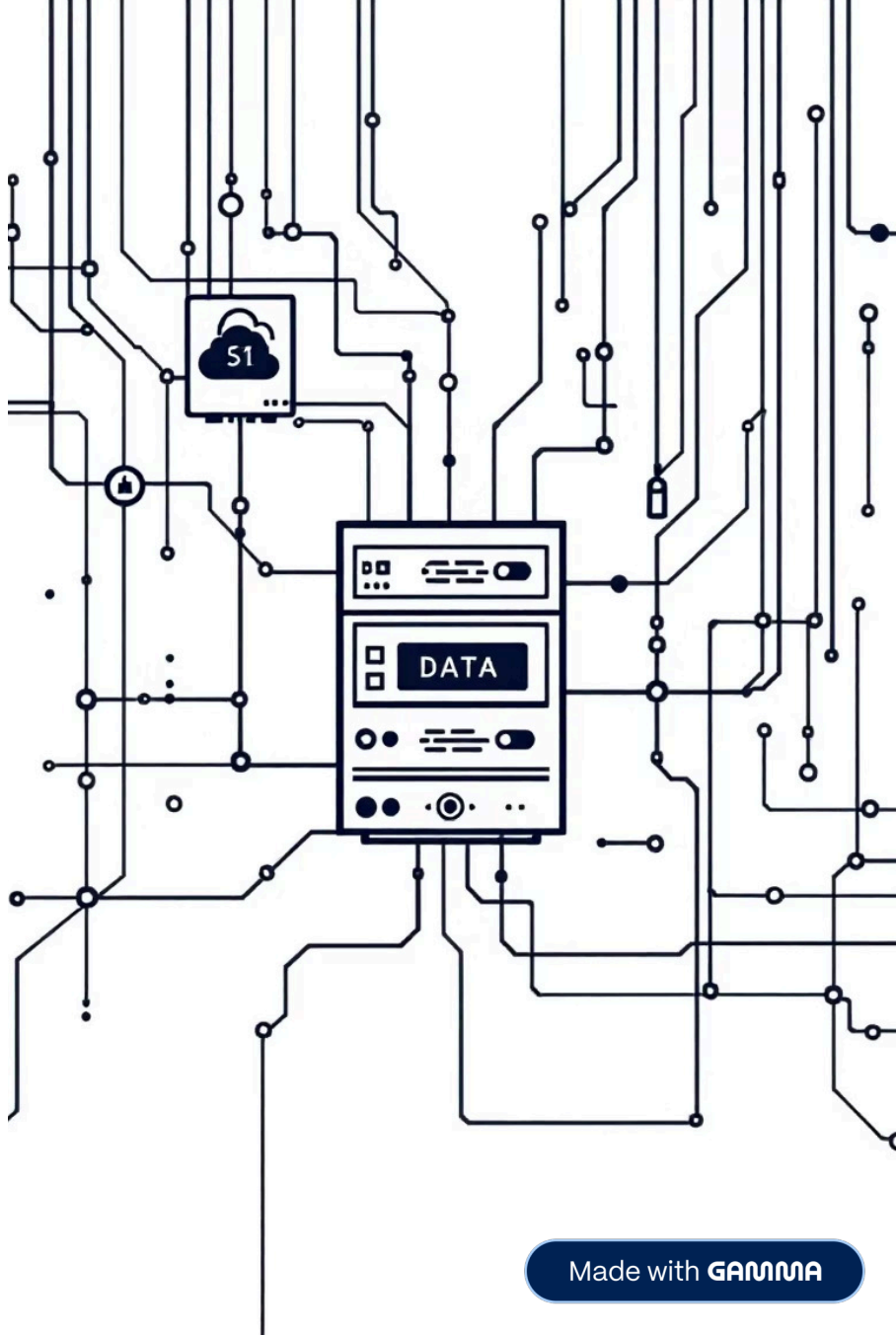
Sessions en PHP

Gestió d'estat al servidor

ASIX2 / SM9 - Implantació d'aplicacions web



Què són les sessions?

El problema d'HTTP

HTTP és un protocol **sense estat** (stateless). Això significa que cada petició és independent i el servidor no recorda res de les peticions anteriors.

Com podem mantenir la informació d'un usuari mentre navega per diverses pàgines del nostre lloc web?

La solució: Sessions

Les sessions són un **mecanisme per emmagatzemar dades de l'usuari al servidor** entre diferents peticions HTTP.

- Ubicació: **Servidor** (no client)
- Identificació: ID únic de sessió
- Nom per defecte: PHPSESSID

Com funcionen les sessions?

01

Usuari accedeix al lloc

El servidor crea una sessió nova amb un ID únic (exemple: abc123)

02

Servidor envia cookie

S'envia una cookie PHPSESSID=abc123 al navegador del client

03

Dades al servidor

Totes les dades es desen **AL SERVIDOR**, no al client

04

Peticions posteriors

En cada petició, el client envia automàticament la cookie PHPSESSID

05

Recuperació de dades

El servidor recupera les dades associades a l'ID abc123


```
Client --solicitud--> Servidor
      <--PHPSESSID-- (crea sessió, desa dades)
Client --PHPSESSID--> Servidor
      <--dades------ (recupera dades)
```

Galetes vs Sessions

És fonamental entendre les diferències entre aquests dos mecanismes per escollir el més adequat segons les nostres necessitats.

Característica	🍪 Galetes	🔐 Sessions
Emmagatzematge	Client (navegador)	Servidor
Mida màxima	~4 KB	Il·limitada*
Seguretat	Menys segur	Més segur
Caducitat	Configurable	Navegador/timeout
Velocitat	Més ràpid	Lleugerament més lent
Dades sensibles	❌ NO recomanat	✅ SÍ adequat

*Dins dels límits de memòria i configuració del servidor

Usos principals de les sessions

Autenticació

Login i logout d'usuaris, mantenint l'estat autenticat mentre naveguen pel lloc

Carros de compra

Emmagatzemar productes seleccionats mentre l'usuari continua comprant

Preferències temporals

Configuracions de l'usuari durant la visita actual al lloc web

Formularis multipàgina

Desar dades entre diferents passos d'un formulari llarg

Control d'accés

Restringir l'accés a pàgines segons el rol o permisos de l'usuari

Protecció CSRF

Prevenir atacs Cross-Site Request Forgery mitjançant tokens de sessió

Gestió bàsica: Inici i emmagatzemament

session_start()

Aquesta funció és **crítica** i s'ha de cridar seguint aquestes regles:

- ✅ A **TOTES** les pàgines que usin sessions
- ✅ **ABANS** de qualsevol sortida HTML
- ✅ **UNA** sola vegada per pàgina

Si envieu HTML abans de session_start(), obtindreu l'error "Headers already sent"

Emmagatzemar dades

Utilitzem l'array superglobal **$_SESSION**:

```
<?php
session_start();

$_SESSION['nom'] = 'Joan';
$_SESSION['edat'] = 25;
$_SESSION['regals'] = ['Llibre', 'Pilota'];
?>
```

Tipus suportats: strings, enters, floats, arrays, objectes

Lectura i modificació de dades

Recuperar dades

```
<?php
session_start();

if (isset($_SESSION['nom'])) {
    echo "Hola, " . $_SESSION['nom'];
} else {
    echo "No heu iniciat sessió";
}
?>
```

⚠️ **Important:** Sempre utilitzeu isset() abans d'accedir a variables de sessió per evitar errors.

Actualitzar dades

```
<?php
session_start();

// Modificar valor
$_SESSION['nom'] = 'Maria';

// Afegir a array
$_SESSION['regals'][] = 'Joc';

// Eliminar element específic
unset($_SESSION['regals'][0]);

// Eliminar variable completa
unset($_SESSION['edat']);
?>
```

💡 Les modificacions estan disponibles immediatament per a totes les pàgines que utilitzin sessions.

Finalització i destrucció de sessions

1

Esborrament de variables

Elimina el contingut però manté la sessió activa

```
$_SESSION = array();
```

2

Destrucció de sessió

Elimina completament la sessió del servidor

```
session_destroy();
```

3

Eliminació de cookie

Esborra la cookie PHPSESSID del navegador

```
setcookie(session_name(), '', time()-3600, '/');
```

Logout complet (recomanat)

```
<?php
session_start();
$_SESSION = array();
if (isset($_COOKIE[session_name()])) {
 setcookie(session_name(), '', time()-3600, '/');
}
session_destroy();
header('Location: index.php');
exit;
?>
```

Configuració i seguretat

⏰ Timeout

Control del temps de caducitat:

```
// php.ini
session.gc_maxlifetime = 3600

// Codi PHP
ini_set('session.gc_maxlifetime', 3600);
```

Valors típics:

- 1800s = 30 min
- 3600s = 1 hora
- 7200s = 2 hores

🔐 Regeneració d'ID

Prevenir fixació de sessió:

```
<?php
session_start();

// Després de login exitós
if ($login_correcte) {
    session_regenerate_id(true);
    $_SESSION['usuari'] = $usuari;
}
?>
```

Sempre regenereu després de canvis d'autenticació.

🛡️ Nom personalitzat

Diferenciar aplicacions:

```
<?php
session_name('MEU_SITE');
session_start();
?>
```

Per defecte: **PHPSESSID**

Útil quan teniu múltiples aplicacions al mateix domini.

Bones pràctiques i errors comuns

✅ Bones pràctiques

1. Crideu session_start() a totes les pàgines
2. Regenereu ID després d'autenticar
3. Establiu un timeout adequat
4. Valideu sempre les dades de sessió
5. Escapeu sortides amb htmlspecialchars()
6. Destruïu la sessió en logout
7. **MAI** emmagatzemeu contrasenyes
8. Utilitzeu HTTPS en producció
9. Protegiu pàgines sensibles
10. Netegeu sessions antigues regularment

⚠️ Errors freqüents

Headers already sent

Enviar HTML abans de session_start()

Oblidar session_start()

No cridar-la en totes les pàgines necessàries

No usar isset()

Accedir a variables sense comprovar si existeixen

No escapar sortides

Vulnerabilitat XSS amb dades no escapades

📚 **Referències útils:** Manual oficial PHP •

jordi.binefa@fje.edu