DAR - Cours 5
Généralités

Romain Demangeon

APR, LIP6, UPMC

15/10/2018

Projet DAR 2017

Ce qui doit être fait:

- Structure de la partie client.
- Prototype d'application.
- Ecrire du code client simple.

A faire cette semaine:

- Enrichir le projet (fonctionnalités).
- Prototype de Rapport.

Rapport et Soutenance

Rapport

- Facile: Manuel d'utilisation.
- Facile: Présentation de l'application.
  - description, uses cases, architecture, technologies.
- Difficile: Raconter des choses intéressantes:
  - Choix techniques.
  - Valeur ajoutée.

Soutenances

- 10min de Présentation: choisir un ou deux points intéressants.
- 5min de Démo: s'assurer que l'appli fonctionne.

Examens

Principe

Un sujet d'application est donné; 2 heures pour réaliser un plan détaillé de l'application.

- Objectif 1: réfléchir sur le design d'applications.
- Objectif 2: mobiliser les connaissances de cours.
- Plan guidé par des questions.
- Tous documents papier autorisés.
- Similaires aux exercices de TDs:
  - Use cases à développer,
  - Découpage en servlets à décider, ...
  - Morceaux de code (structure) à écrire.

Framework d'Applications Web

- *framework* (canevas, cadriciel): ensemble de composants logiciels structurels permettant la création d'un logiciel.
  - généralement générique, faiblement spécialisé,
  - le cadre de travail impose certains motifs.
- WAF (*Web Application Framework*): sert à développer les applications web, les sites dynamiques, services web

- des dizaines de cadriciels différents:
  - Langages différents,
  - Objectifs différents,
  - Modèles structurels différents,
  - Modèles de développement (licence).

Pas de consensus sur *le meilleur cadriciel*.

Fonctionnalités courantes d'un Cadriciel Web

- Système de gabarits (*templates*) pour pages Web,
- Gestion du cache,
- Contrôle d'accès: authentification, autorisation,
- Gestion de la persistance, du modèle relationnel,
- Echafaudage (*scaffolding*),
- Gestion des formulaires,
- AJAX,
- Services (SOAP) et Ressources (REST),
- Assignement d'URL,
- Internationalisation et régionalisation.

- Plusieurs approches différentes:
  - Programmatique, Modèle, Hybrides, MVC,
  - Séparation developpeur/designers

Approche Programmatique

- Approche où la source d'une page web est majoritairement composé de code dans un langage de haut-niveau ou de script.
- logique (Java, JS, ...) > structure (HTML)

- le format de la page (HTML) est produit par du code,

```
PrintWriter out = response.getWriter();
out.println("<title>Example</title><body>");
```

```
titre = getElementByName('letitre');
titre.innerHTML = "Example";
```

- Puissant: permet d'intégrer beaucoup de logique à la génération des pages (interactivité).
- Limites:
  - c'est de la programmation,
  - difficulté de visualiser la structure,
- Exemple: Servlets, CGI, JS.

Approche Gabarit (*Template*)

- Approche où la source d'une page web est majoritairement composé de structure.
- structure (HTML) > logique (scripts)

- le format de la page (HTML) contient des scripts,
- utilisateur-sympathique pour les développeurs webs,
- Limites:
  - limité dans la logique,
  - difficulté pour comprendre l'application,
- Exemple: SSI, Apache Velocity.

Server Side Includes (SSI)

Un langage de scripts côté serveur, interprété par un serveur HTTP.
Supporté directement par Apache et IIS sous forme de fichiers `.shtml`.

- *Includes*, utilisation principale: inclure plusieurs fichiers pour construire une page HTML.
- permettent de gagner de l'espace de stockage en écrivant dans un unique fichier les informations partagées sur l'ensemble des pages d'un site.
- exemple: en-têtes et pieds de page:

```
<!--#include file="entete.html" -->
<p>Le répertoire contient les fichiers suivants :</p>
<pre><!--#exec cmd="ls"--></pre>
<!--#include file="pied.html" -->
```

- simple mais pas très puissant,
- technologie historique (remplacé par PHP et Active Server Pages).

Apache Velocity

Moteur de gabarits libre, basé sur Java qui propose un langage de gabarits permettant de manipuler des objets référencés Java.

- utilisé pour faire de l'HTML, mais aussi pour de la génération de code source, des mails automatiques, de la conversion de document (Transformation XML).

Code source Velocity

```
## Velocity Bonjour Monde
<html>
  <body>
    #set( $toto = "Annie" )
    ## followed by
    Bonjour $toto !
  </body>
</html>
```

Code produit HTML

```
<html>
    <body>
     Bonjour Annie !
    </body>
</html>
```

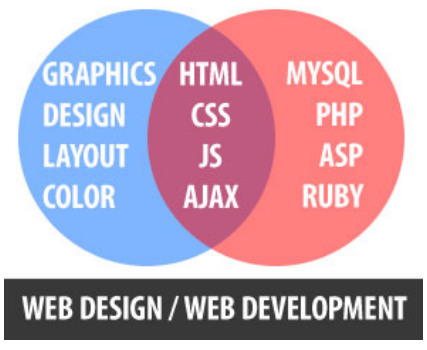
- logique standard:

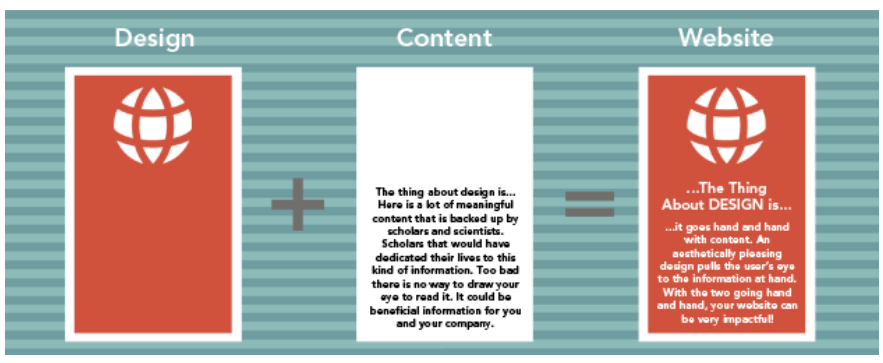
```
<ul>
#foreach( $chapitre in $livre.sommaire )
    <li>$chapitre</li>
#end
</ul>
```

Approches Hybrides

- Approche où les éléments de scripts et les éléments de structure co-existent au sein du même code source avec la même importance.
- logique = structure.

- les développeurs et les *designers* modifient le même fichier source.
- exemples: PHP, ASP.NET, JSP (sans code Java supplémentaire).





PHP: Hypertext Preprocessor (Personal Home Page)

Langage de script côté serveur, utilisé principalement pour les pages webs dynamiques, mais aussi utilisé comme langage générique.

- Langage interprété.
- Utilisé par 244 millions 40% des sites.
- Langage libre.
- Beaucoup de cadriciels web sont basés sur PHP.
- Fait partie du paquet LAMP.

Code PHP - Exemple

Récupérer des informations de la requête:

```
<?php
    $nom = $_POST['nom'];
    if ($nom === 'Cordy')
        echo 'Bonjour Annie !';
    else
        echo 'Bonjour !';
?>
```

Générer du code HTML:

```
<html>
  <body>
    <h1>
      <?php if ($title != "") {
          print $title;
        } else {
          ?>Titre par défaut<?php } ?>
      </h1>
  </body>
</html>
```

Générer du code HTML (hyperlien):

```
<?php
    echo '<a href="' . htmlspecialchars("/nextpage.php?stage=23&data=" .
        urlencode($data)) . '">'."\n";
?>
```

Code PHP - Manipulation DOM

```php
<?php
    $doctype = DOMImplementation::createDocumentType('html', "-//W3C//DTD HTML 4.01//EN","http://www.w3.org/TR/html4/strict.dtd
    $dom = DOMImplementation::createDocument(null, "html", $doctype);
    $html = $dom->documentElement;
    $html->head = $dom->createElement("head");
    $html->appendChild($html->head);
    $html->head->title = $dom->createElement("title");
    $html->head->title->nodeValue = "Exemple de HTML";
    $html->head->appendChild($html->head->title);
    $html->head->charset = $dom->createElement("meta");
    $html->head->charset->setAttribute("http-equiv", "Content-Type");
    $html->head->charset->setAttribute("content", "text/html; charset=utf-8");
    $html->head->appendChild($html->head->charset);
    $html->body = $dom->createElement("body");
    $html->appendChild($html->body);
    $html->body->p = $dom->createElement("p");
    $html->body->p->nodeValue = "Ceci est un paragraphe.";
    $html->body->appendChild($html->body->p);
    $html->body->p->br = $dom->createElement("br");
    $html->body->p->appendChild($html->body->p->br);
    $html->body->p->a = $dom->createElement("a");
    $html->body->p->a->nodeValue = "Ceci est un lien.";
    $html->body->p->a->setAttribute("href", "cible.html");
    $html->body->p->appendChild($html->body->p->a);
    print($dom->saveHTML());
?>
```

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Exemple de HTML</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body><p>Ceci est un paragraphe.<br/><a href="cible.html">Ceci est un lien.</a></p></body>
</html>
```

ASP.NET

Cadriciel web de Microsoft, faisant partie du cadriciel .NET.

- Langage compilé en DLLs.
- Similaire à PHP, du code (Visual Basic, C#) peut être introduit dans des pages HTML.
- ASP.NET AJAX: cadriciel serveur + bibliothèque Javascript client pour AJAX
- ASP.NET MVC Framework: cadriciel Modèle-Vue-Controleur sur ASP.NET.
- ASP.NET Web API: API HTTP pour les services webs.
- ASP.NET SignalR: cadriciel de communication client-serveur en temps réel.

Exemple d'ASP.NET

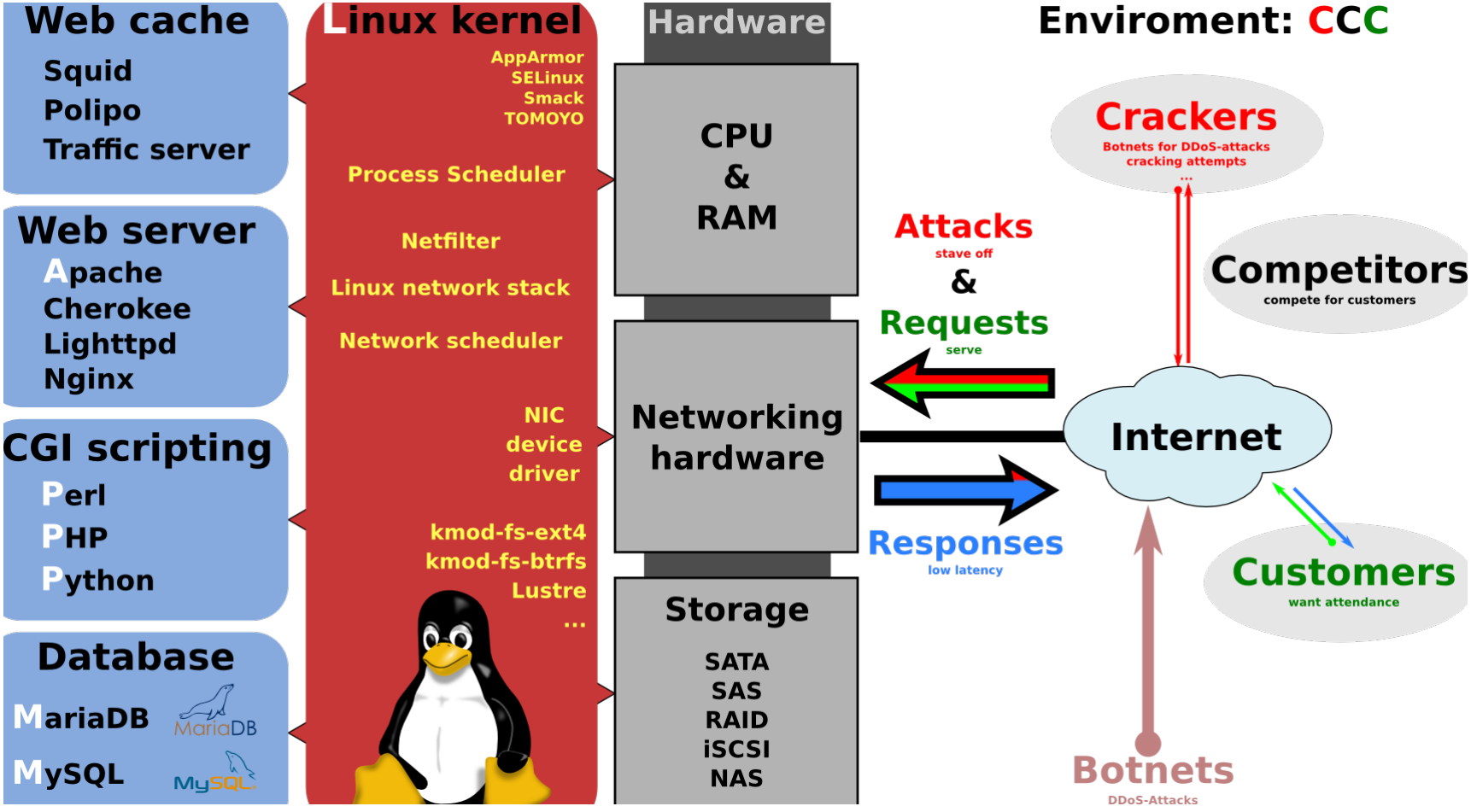
En C#

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "---//W3C//DTD XHTML 1.0 //EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    lbl1.Text = DateTime.Now.ToLongTimeString();
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Sample page</title>
</head>
<body>
  <form id="form1" runat="server">
  </form>
</body>
</html>
```

En VB

```
Imports System
Namespace Website
  Public Partial Class SampleCodeBehind
          Inherits System.Web.UI.Page
          Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
             Response.Write("Hello, world")
          End Sub
  End Class
End Namespace
```

Le Paquet LAMP

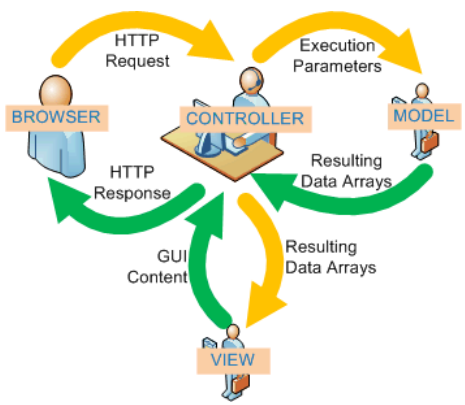


Approches MVC

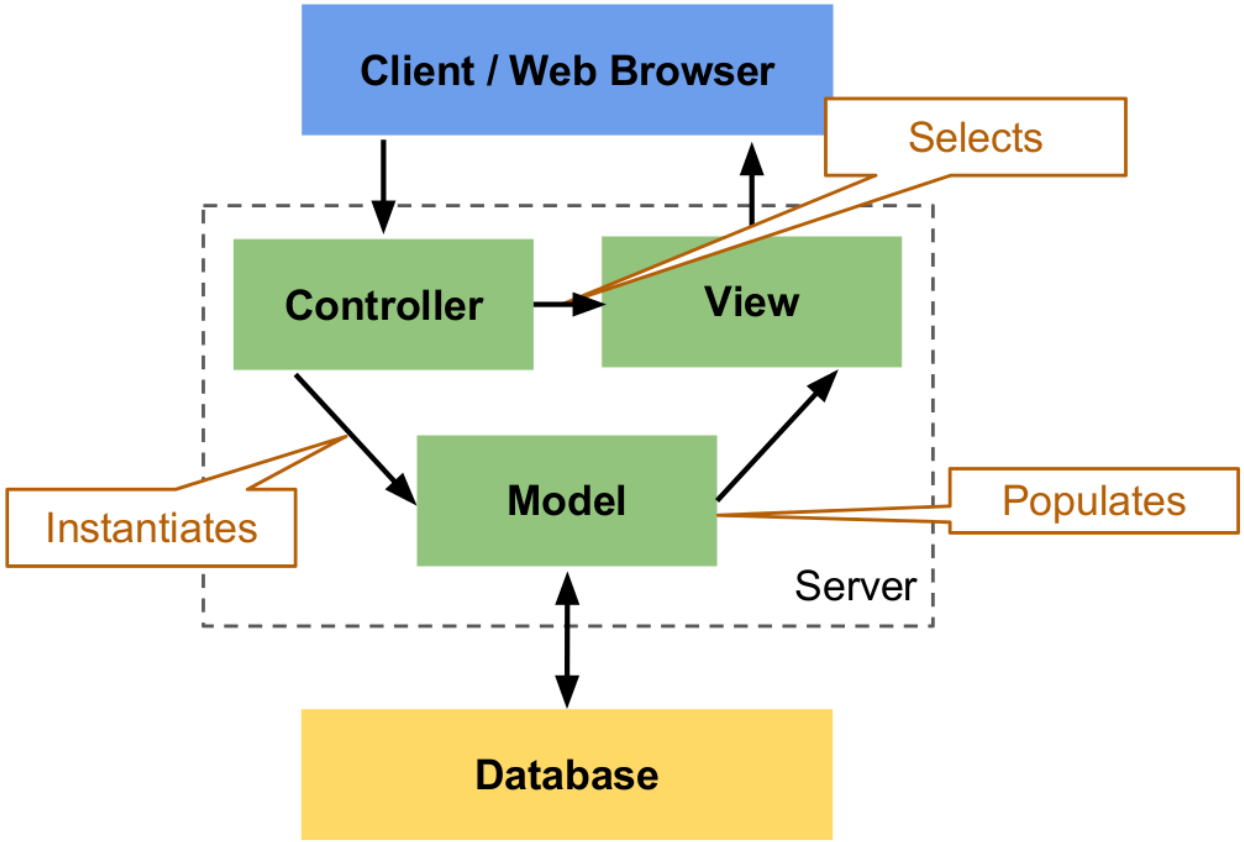
Patron de programmation séparant les différents composants

- Modèle: modèle de données, logique, fonctions,
- Vue: sortie, représentation de l'information,
- Contrôleur: envoi de commandes, édition.

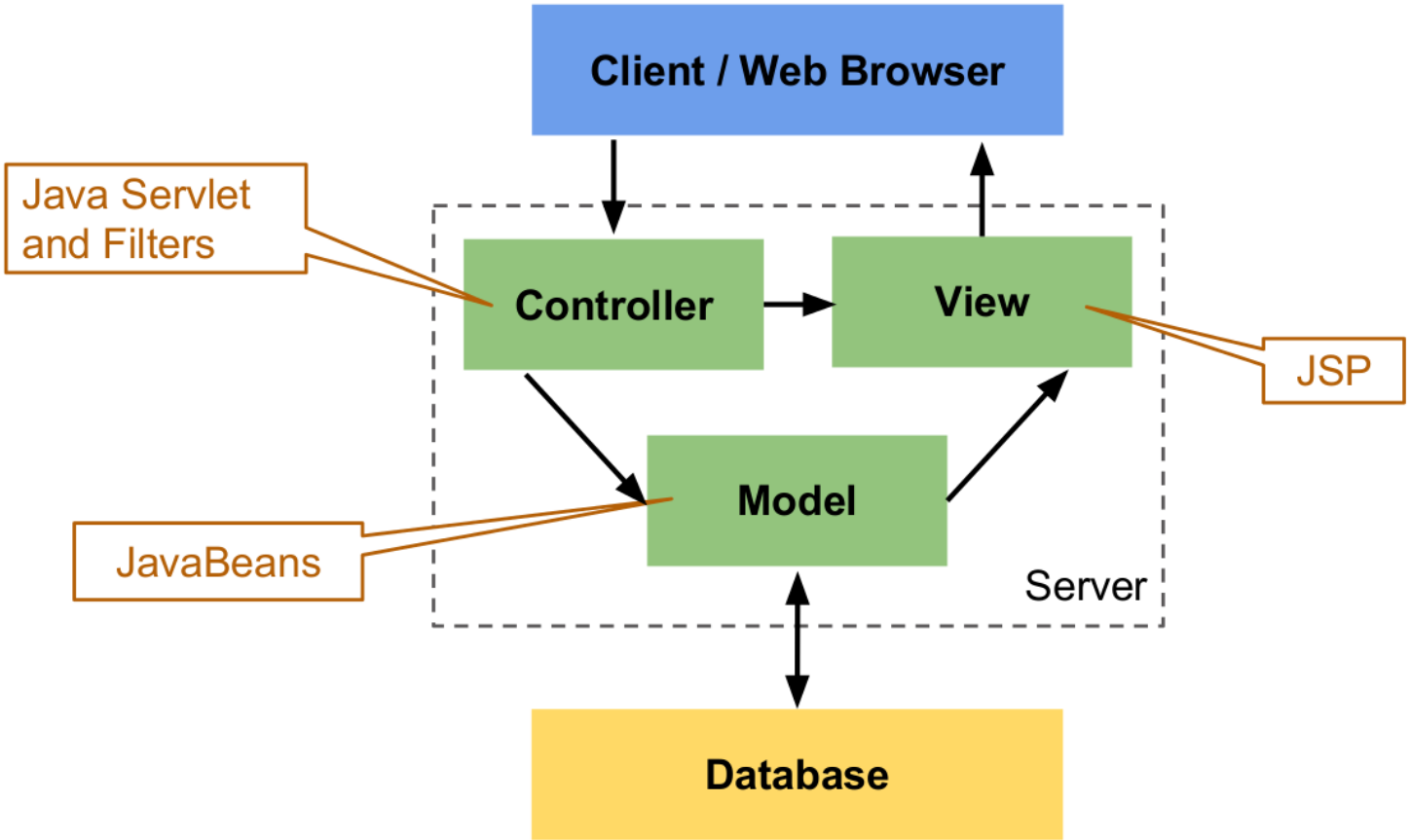
- Utilisé par de nombreux cadriciels (non-spécifique au web).
- Dans les applications web:
  - les requêtes HTTP sont envoyées au contrôleur,
  - le contrôleur accède aux donnés et instantie le modèle,
  - le contrôleur sélectionne, construit et passe le contrôle à la vue (une page dynamique accédant à des données du modèle)
  - la page est calculée puis envoyée comme réponse HTTP.



Modèle-Vue-Contrôleur

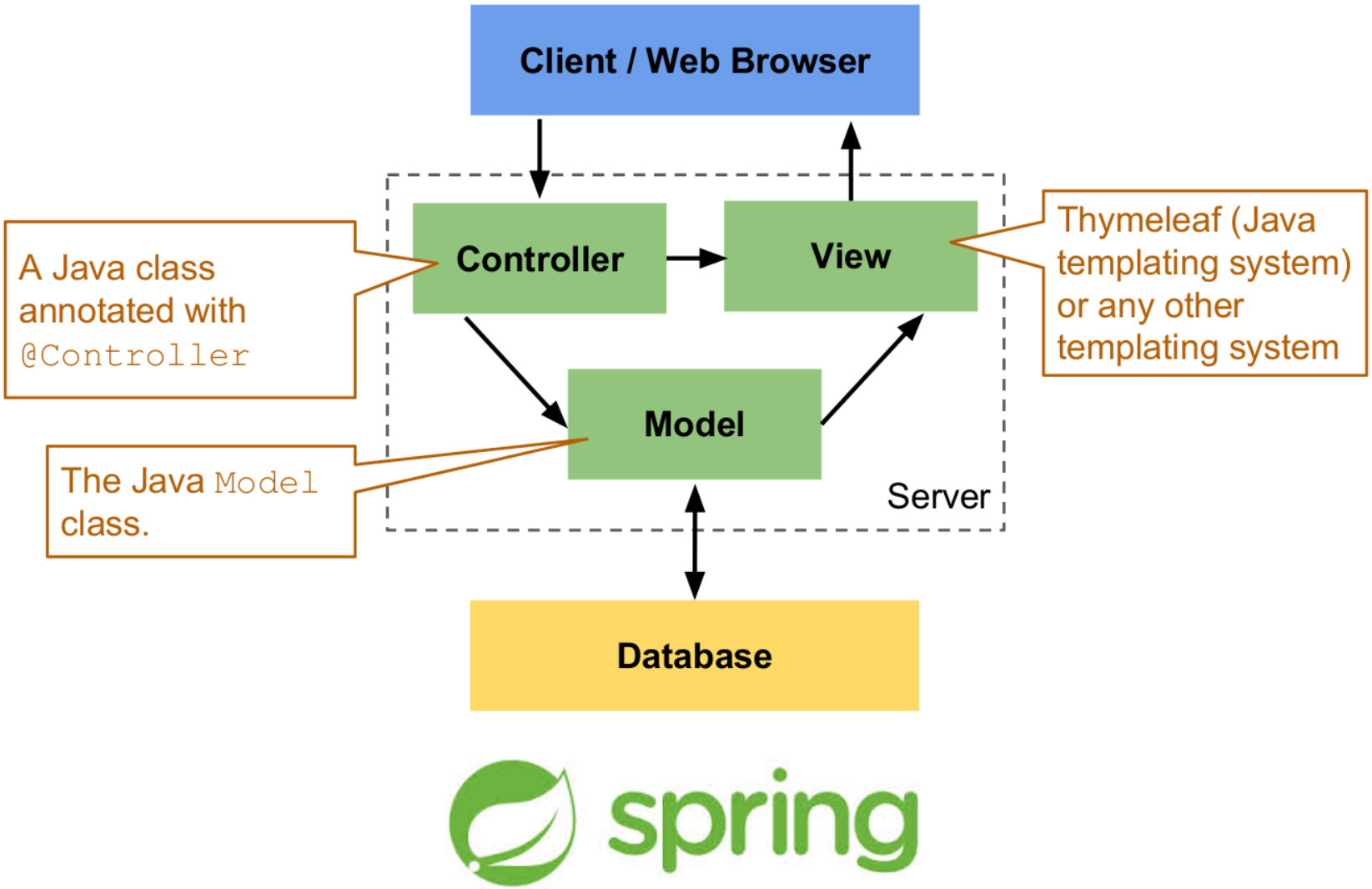


Apache Struts

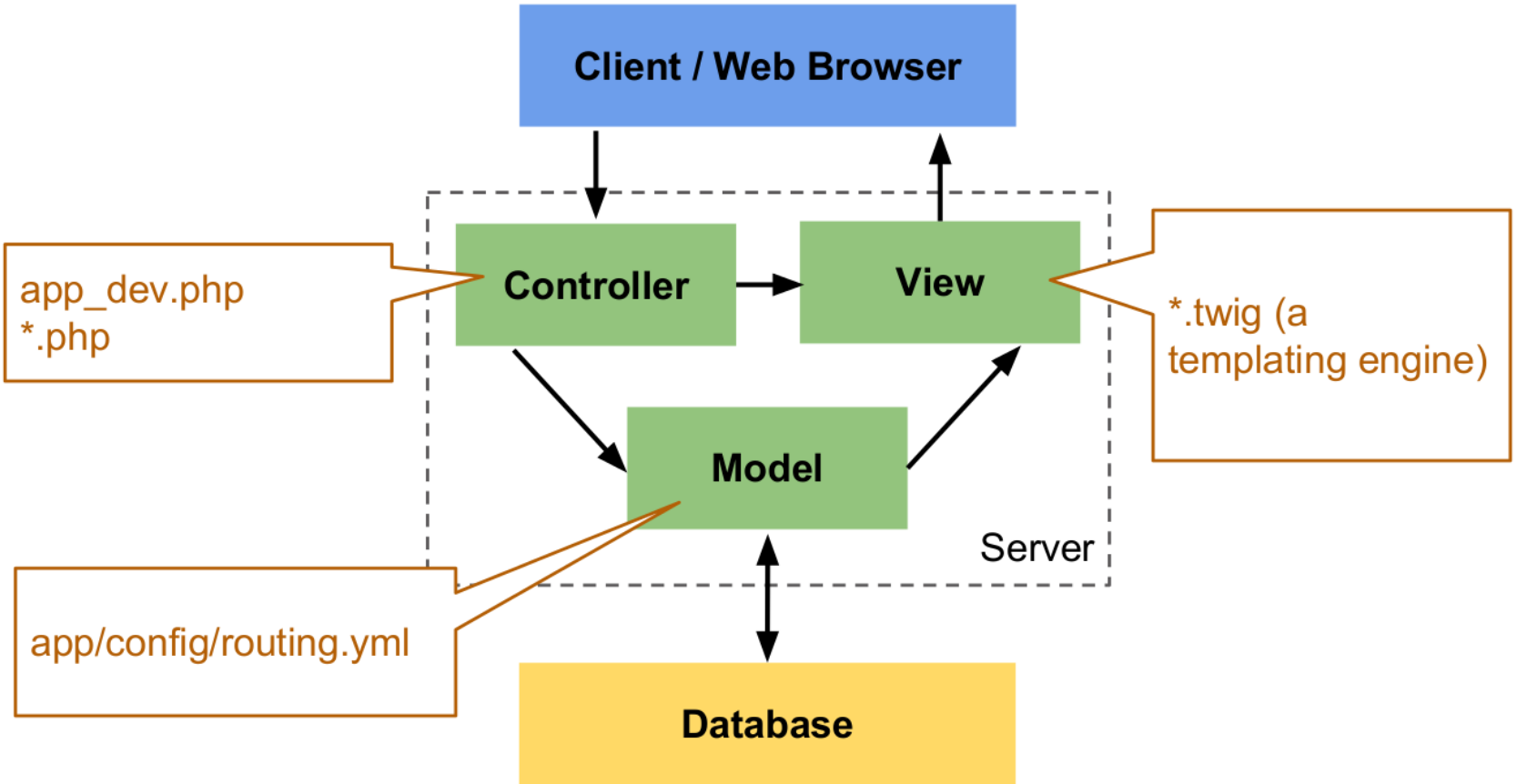


Struts

Spring

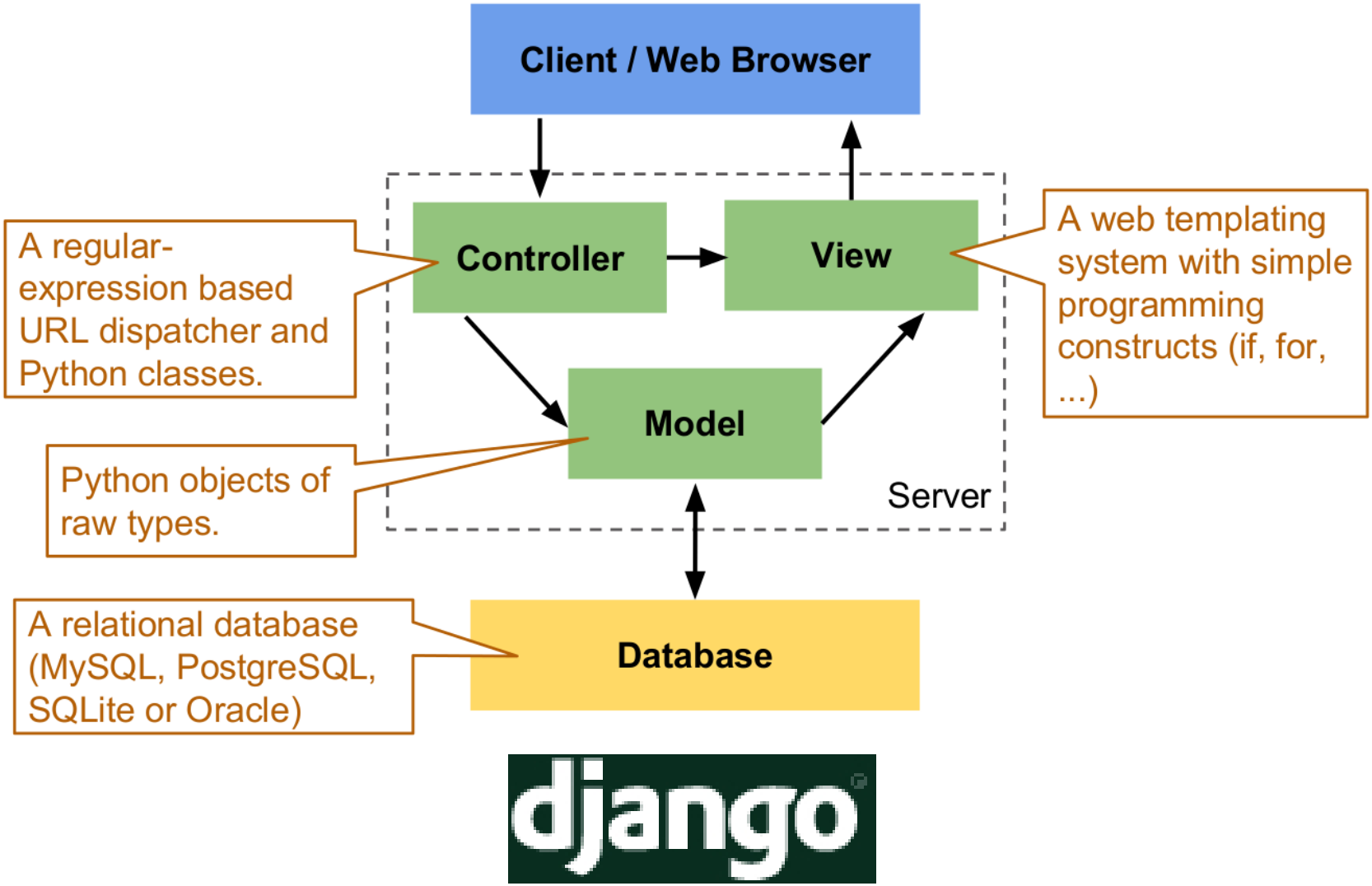


Symfony

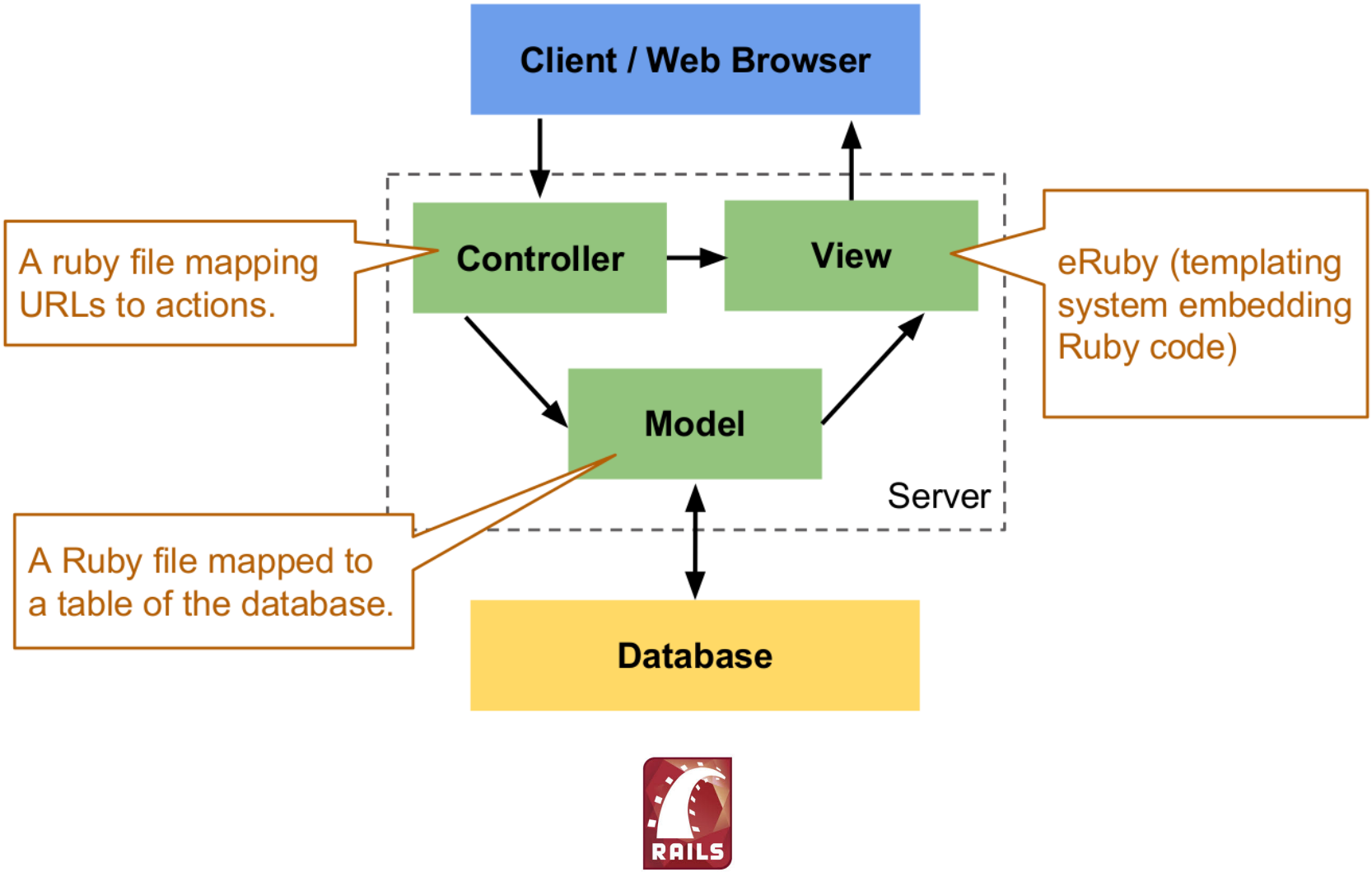


Symfony

Django



Ruby on Rails



Push vs Pull

- un cadriciel MVC travaille en poussant (*push-based*) quand une action de l'utilisateur déclenche le calcul et les données sont envoyées à la vue pour représentation.
- un cadriciel MVC travaille en tirant (*pull-based*) quand la vue a l'initiative et récupère des données de plusieurs contrôleurs.

- Exemples de cadriciels *pull-based*:
  - JavaServer Faces: l'état est sauvé entre les requêtes dans des Facelets,
  - Wicket: arbres de composants qui réagissent aux requêtes HTTP
  - Lift: code Scala dans une JVM.

Exemple de code *pull-based* (Wicket)

Template XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:wicket="http://wicket.apache.org/dtds.data/wicket-xhtml1.3-strict.dtd"
   xml:lang="en" lang="en">
 <body>
    <span wicket:id="message" id="message">Message goes here</span>
 </body>
</html>
```

Composant java

```
package org.wikipedia.wicket;
import org.apache.wicket.markup.html.WebPage;
import org.apache.wicket.markup.html.basic.Label;
public class HelloWorld extends WebPage {
    /**
     * Constructor
     */
    public HelloWorld() {
        add(new Label("message", "Hello World!"));
    }
}
```

Exemple de code *pull-based* (Wicket)

Application Java (transforme la requête en appel au composant)

```
package org.wikipedia.wicket;
import org.apache.wicket.protocol.http.WebApplication;

public class HelloWorldApplication extends WebApplication {
    /* Constructor.*/
    public HelloWorldApplication() {
    }
     /* @see org.apache.wicket.Application#getHomePage()*/
    public Class getHomePage() {
        return HelloWorld.class;}}
```

Descripteur de déploiement: initialise Wicket et lie l'application

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                             http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         id="WebApp_ID" version="2.5">
    <display-name>Wicket Example</display-name>
    <filter>
        <filter-name>HelloWorldApplication</filter-name>
        <filter-class>org.apache.wicket.protocol.http.WicketFilter</filter-class>
        <init-param>
            <param-name>applicationClassName</param-name>
            <param-value>org.wikipedia.wicket.HelloWorldApplication</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>HelloWorldApplication</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

Angular JS

Cadriciel web maintenu par Google, pour le développement d'applications web monopages en utilisant l'approche MVC côté client.

- application définie dans une unique page HTML contenant des attributs spéciaux
- le cadriciel lie les entrées/sorties de la page à un modèle utilisant des variables JS standard.
- découple la logique de la manipulation DOM.
- déplace la charge vers le client.

Cadriciels unilangage: GWT

- Historiquement, langages différents pour le client (JS) et le serveur (Java par exemple).
  - pas de réutilisation de code entre les deux,
  - test et déboguage difficile.
- Certains cadriciels utilisent un unique langage.

- Google Web Toolkit, applications AJAX en Java:
  - Compilateur GWT Java-to-Javascript
  - Mode Développement de GWT (pour lancer des applications sans passer par JS)
  - Bibliotheque d'émulation JRE: implantation JS des bibliothèques JRE standard.
  - Bibliotheques de classes *Web UI*: interfaces pour créer des *widgets*.
- permet d'utiliser RPC (SOAP), la sérialisation,
- Utilisé par Google (Groups, Blogger, Adwords).

Métacadriciel unilangage: Node.js

Plateforme logicielle en JS orientée vers les applications réseau.

- du JavaScript pour le serveur,
- interpréteur efficace de JS pour tourner sur un serveur HTTP, ou faire tourner un serveur HTTP,
- de nombreux cadriciels basés sur Node.js: Express, Geddy.

```
var http = require('http');

var server = http.createServer(function(request, response){
    response.writeHead(200, {'Content-Type': 'text/plain'});
    response.end('Hello World\n');
});
server.listen(3000);

console.log('Adresse du serveur: http://localhost:3000/');
```

Langage stratifié: HOP et Ocsigen

Utiliser le même langage pour client et serveur dans le même code source.

- l'utilisateur doit spécifier explicitement quoi exécuter sur le client et sur le serveur.
  - déléguer cette tâche au compilateur est difficile.
- Hop, un langage web basé sur Lisp financé par inria, développé par Manuel Serrano

```
(define-service (server-date)
  (current-date))

(<HTML>
  (<BUTTON>
    :onclick ~(with-hop ($server-date)
                        (lambda (h) (alert h)))
      "Server time"))
```

- Ocsigen: cadriciel web basé sur OCaml:
  - Eliom: un cadriciel pour programmer des sites web et des applications,
  - js_of_ocaml: compilateur,
  - Ocsigen Server: un serveur HTTP,
  - lwt: une bibliothèque de threads.

Ocsigen (Exemple)

```
let def = new_url ~path:["essai"]  ~params:(_current_url _noparam) ()

let post = new_post_url ~fallback:def ~post_params:(_string "group")

let create_form group =
    [p [select ~a:[a_name group]
          (option (pcdata "choi1")) [option (pcdata "choi2")];
              submit_input "Envoyer"
     ]]

let genere_form current_url = post_form post current_url create_form

let _ = register_url def
              (fun cu ->
                       (html
                           (head (title (pcdata "")) [])
                           (body [genere_form cu])))

let fonction group cu = (html
                               (head (title (pcdata "")) [])
                               (body [h1 [pcdata group]; genere_form cu]))

let _ = register_post_url post fonction
```

(de V. Bala)

Langage Stratifié: Opa

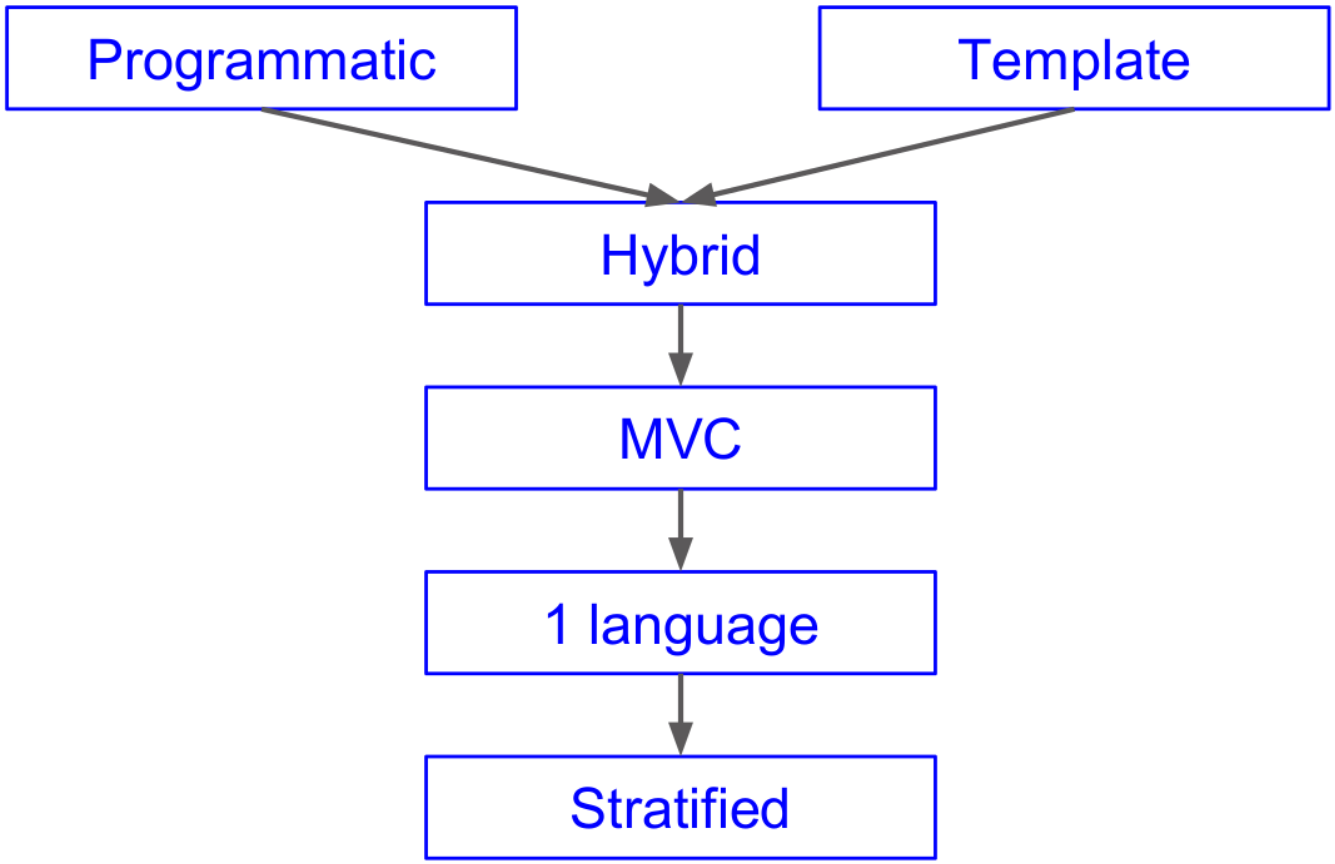
Langage fonctionnel avec types statiques et inférence de types.

- un unique programme est écrit en Opa:
  - il est compilé en JS pour le client et Node.js pour le serveur,
  - le compilateur sépare lui-même le code.
- ne dépend pas des serveurs standard (Apache), implante son propre serveur d'application.
- communication par passage de messages (comme Erlang).

Exemple d'Opa

```
type message =
  { author: string // Le nom de l'auteur
  ; text: string }   // Le texte du message

@publish room = Network.cloud("room"): Network.network(message)

user_update(x: message) =
  line = <div class="line">
           <div class="user">{x.author}:</div>
           <div class="message">{x.text}</div>
         </div>
  do Dom.transform([#conversation +<- line ])
  Dom.scroll_to_bottom(#conversation)

broadcast(author) =
   do Network.broadcast({~author text=Dom.get_value(#entry)}, room)
   Dom.clear_value(#entry)

start() =
   author = Random.string(8)
   <div id=#header><div id=#logo></div></div>
   <div id=#conversation onready={_ -> Network.add_callback(user_update, room)}></div>
   <div id=#footer>
        <input id=#entry  onnewline={_ -> broadcast(author)}/>
        <div class="button" onclick={_ -> broadcast(author)}>Post</div>
   </div>

server = Server.one_page_bundle("Chat",
       [@static_resource_directory("resources")],
       ["resources/css.css"], start)
```

Evolution des approches



Accessibilité

WAI-ARIA (*Web Access Initiative - Accessible Rich Internet Applications*) est une spécification technique du W3C en cours de rédaction concernant l'accessibilité.

- spécifie comment améliorer l'accès au contenu dynamique avec AJAX, HTML, JavaScript.
- ajoute de sémantique et de métadonnées afin de rendre les contenus plus accessibles.
- utilisation de rôles pour spécifier le contenu (par ex., statique ou dynamique ?)

```
<div role="menu" aria-haspopup="true" tabindex="-1">
  File
</div>
```

- Recommandations (extrait):
  - Utiliser les balises standards quand c'est possible.
  - Définir des rôles pour les pages et les éléments.
  - Supporter le clavier seulement.
  - Synchroniser l'IU et l'interface accessible.
  - Délimiter les régions actives d'un document.

Cookies

Un cookie (témoin) est une suite d'informations envoyée d'un serveur HTTP à un client que ce dernier retourne à chaque requête successive.

- on peut aussi traiter les cookies côté client avec des scripts JS.
- Utilisations:
  - Sessions: maintenir un état entre plusieurs requêtes successives (voire plusieurs visites) d'un même utilisateur (paniers de vente en ligne).
  - Personnalisation: stocker des informations à propos d'un utilisateur pour adapter le contenu proposé.
  - Pistage: connaître le comportement d'un utilisateur sur plusieurs visites.
- Historiquement, dans UNIX, un magic cookie est un paquet de donné reçu et retourné inchangé par un programme. Intégré (secrètement) dans Netscape en 1994.
- les navigateurs doivent supporter au moins 300 cookies de 4096 octets simultanément.

Cookies (II)

- Structure d'un cookie:
  - un nom: utile si plusieurs cookies,
  - une valeur: générée par le serveur,
  - une date d'expiration: si expiré, un cookie n'est pas renvoyé,
  - un domaine et un chemin relatifs au cookie,
  - la nécessité d'une connection HTTPS ou non,
  - est-ce que le cookie est accessible autrement que par HTTP (JavaScript)
- Types de cookies:
  - Cookie de session: sans expiration, il disparait à la fermeture du navigateur.
  - Cookie persistent: avec expiration explicite, il est stocké en mémoire et jeté à la date prévue.
  - Cookie sûr: envoyé uniquement à travers le protocole HTTPS.
  - Cookie HttpOnly: inaccessible à Javascript.
  - Cookie tierce partie: cookie d'un autre domaine que celui de la page web
    - Pistage et pixel espion: en utilisant la meme image sur plusieurs site différents, une entreprise peut espionner le comportement d'un utilisateur.

Exemples de Cookies

1. Requête HTTP Client → Serveur:

```
GET /index.html HTTP/1.1
```

2. Réponse HTTP Serveur → Client:

```
HTTP/1.0 200 OK
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Wed,09 Jun 2021 10:18:14 GMT
```

3. Requête HTTP suivante, Client → Serveur:

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value; name2=value2
```

Exemple

En-tête Réponse HTTP:

```
Set-cookie:ubid-acbfr=272-5882146-5830346; path=/; domain=.amazon.fr;
 expires=Mon, 31-Dec-2035 23:00:01 GMT
```

Propriétés requises

- Authentification: prouver l'identité des utilisateurs.
- Contrôle d'accès: restreindre certaines ressources à certains utilisateurs.
- Intégrité des données: pouvoir prouver que des données n'ont pas été modifiées.
- Confidentialité: s'assurer que l'information est disponible uniquement à des utilisateurs autorisés.

Implantées à trois niveaux:

- le protocole HTTPS,
- des outils donnés par le cadriciel,
- la logique propre à l'application.

HTTP Secure (HTTPS)

Version sécurisée du protocole HTTP (port TCP 443).

- Authentification du serveur: les serveurs HTTPS ont des certificats vérifiés (par des autorités, *Verisign* par exemple). Les navigateurs ont des certificats publics de ces autorités.
- Chiffrement SSL de tout le message HTTP (mais pas des en-têtes TCP/IP)
- Authentification des utilisateurs: un certificat créé pour chaque utilisateur.

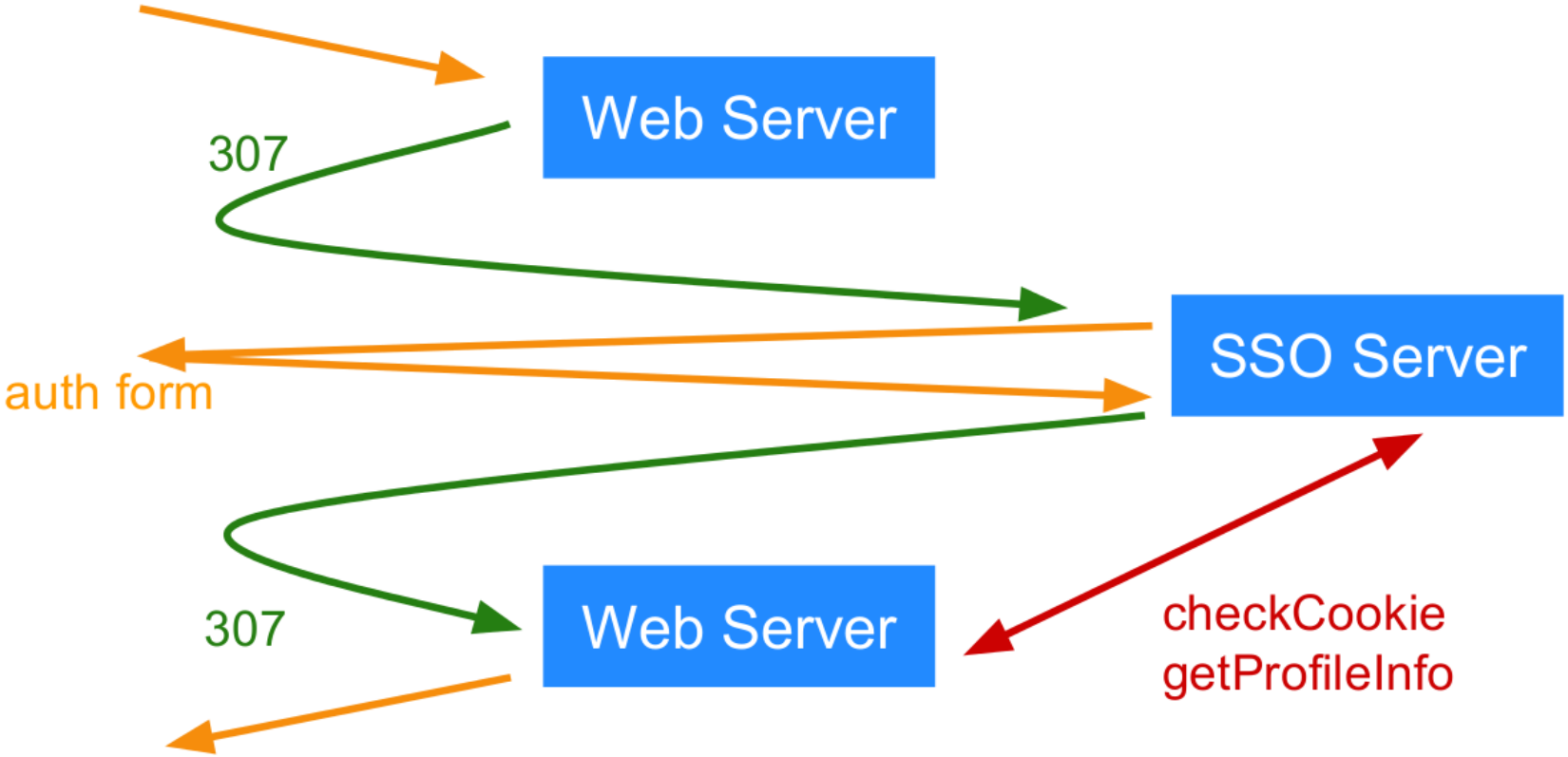
- Attaques HTTPS:
  - Homme-du-milieu (2009): changer des liens `https` en `http`.
  - Obtention frauduleuse de certificats (2011): récupérés sur une ancienne autorité (Diginotar), utilisés pour créer de faux sites Google.
  - Heartbleed (2014): faille dans OpenSSL.

Single Sign-On (SSO)

L'authentification unique est une méthode de contrôle d'acces qui permet aux utilisateurs de se connecter à plusieurs sites/applications en s'authentifiant une seule fois.
Centraliser sur un serveur unique des coordonnées relatives à plusieurs applications .

- Gère plusieurs protocoles d'authentification en même temps.
- Implémenté avec LDAP (*Lightweight Directory Access Protocol*)
  - protocole pour stocker des répertoires accessibles en ligne.
- Combat la fatigue du mot de passe (*password fatigue*).
  - surabondance d'authentifications.
  - idéalement, mots de passe différents,
  - surcharge morale sur les utilisateurs et les services techniques.
- *Reduced Sign-On*: plusieurs niveaux de sécurité.

SSO - Schéma



- 307: redirection vers et depuis le serveur SSO.

Ne jamais faire confiance au client

Un serveur web ne doit jamais faire confiance à aucune information venant du client.

- il ne peut pas supposer que les requêtes ont été générées par le code client JS.
- toujours vérifier les types des arguments,
- l'implémentation de logique sur le client (IU dynamique) implique la réimplémentation de cette logique sur le serveur,
- ne jamais envoyer des informations confidentielles cachées.

Exemple: vente en ligne,

- envoi du catalogue au client,
- calcul de la valeur du panier côté client,
- le serveur doit recalculer la valeur du panier.

Retour sur la SOP (*Same Origin Policy*)

- Rappel: les scripts exécutés sur des pages venant du même site (domaine + hôte + port) peuvent accéder à leur DOM sans restriction.
  - les scripts exécutés sur des sites différents ne peuvent pas.
- en conséquence, un script ne peut faire des requêtes AJAX que sur le même site que sa page.
- la SOP ne s'applique pas aux éléments ⟨`img`⟩, ⟨`script`⟩ ou ⟨`object`⟩.
- la SOP peut être relaxée par:
  - `document.domain` peut-être changé pour un superdomaine.
  - *Cross-Origin Resource Sharing*

    ```
    Origin: http://www.saucisse.com
    Access-Control-Allow-Origin: http://www.saucisse.com
    ```
  - *Cross-document messaging*

Cross-Document Messaging

- Fait partie de HTML5.
- Communication en texte simple.

Dans le document Alice, un autre document Bob est chargé dans une `iframe`:

```
var o = document.getElementsByTagName('iframe')[0];
o.contentWindow.postMessage('Bonjour Bob', 'http://saucisse.com/');
```

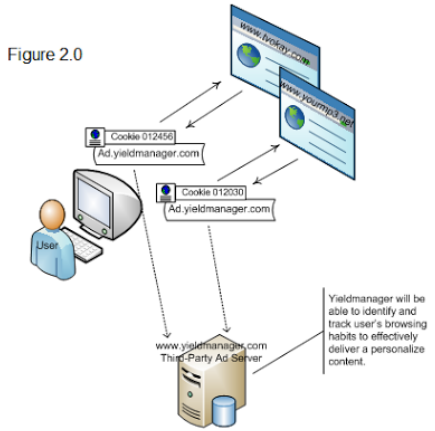
Dans le document Bob, les messages sont récupérés par un gestionnaire:

```
function receiver(event) {
  if (event.origin == 'http://saucisse.net') {
    if (event.data == 'Bonjour Bob') {
      event.source.postMessage('Bonjour Alice, ca va?', event.origin);
    }
    else {
      alert(event.data);
    }
  }
}

window.addEventListener('message', receiver, false);
```

Retour sur les Cookies de Tierce Partie

Comme la SOP ne s'applique pas aux champs ⟨`img`⟩, ⟨`object`⟩, ⟨`script`⟩, on peut utiliser les requêtes GET vers les cibles de ces balises pour planter des cookies tierce partie.

- Safari bloque les cookies TP par défaut.
- Firefox prévoit de le faire.
- Tous les navigateurs permettent de le faire.
  - A prendre en compte en développant des applications web.



Retour sur les Cross-Site Scripting

Un XSS permet à un attaquant d'injecter du code client Javascript dans une page web vue sur le navigateur d'un autre utilisateur.

- cela permet de passer outre la SOP: le script est exécuté dans le contexte de sécurité de l'application.
- si Google renvoyait en brut la recherche saisie
  - Alice envoie un mail à Bob avec un lien de résultats Google:
    `http://www.google.com/?q=<script src="http://alice.fr/script.js"/>`
  - Bob clique sur le lien, le script d'Alice est exécuté dans le contexte de Bob et récupère (ou modifie) ses données Google.
- si Facebook autorisait les messages contenant des tags HTML:
  - Alice écrit un tag sur le mur de Bob:
    `<script src="http://alice.com/script"/>`
  - Quand Charlie consulte le mur de Bob, son navigateur exécute le script.
  - Le script envoie à Alice les likes de Charlie.

Plus sur XSS

Saveurs de XSS:

- XSS non-persistent: la balise HTML malveillante vient directement du client et n'est pas stockée sur le serveur.
  - par exemple, la page HTML générée par le serveur contient une URL,
  - l'attaquant doit préparer son URL et faire cliquer la victime.
- XSS persistent: la balise HTML malveillante est stockée sur le serveur (comme du contenu généré par un utilisateur).
  - l'attaquant crée le contenu et doit faire visiter la victime.

Eviter les XSS:

- Protéger/Echapper toutes les chaines de caractères générées qui ne devraient pas contenir de l'HTML.
  - Whitelister les balises HTML acceptables.
  - en JSP:

```
<c:out value="${param.foo}" />
<input type="text" name="foo"
value="${fn:escapeXml(param.foo)}" />
```

  - en Servlet:

```
StringEscapeUtils.escapeHtml()
```

Cross-Site Request Forgery (XSRF)

- En utilisant des balises ⟨`img`⟩ ou ⟨`script`⟩:
  - une page hébergée sur un hôte X déclenche un GET vers un hôte Y.
  - la plupart des navigateurs vont inclure les cookies de Y dans la requête.
- X peut inclure des balises malveillantes dans sa page qui envoient des requêtes non désirées à Y.
- Par exemple si Gmail autorisait les formulaires simples pour envoyer des emails en utilisant des cookies pour l'authentification:

```
<form method="GET" action="/sendmail">
  <input name="to"/>
  <input name="body"/>
</form>
```

- alors Alice pourrait mettre sur sa page:

```
<img src="http://mail.google.com/sendmail?to=charlie@foo.fr&body=Salut!">
```

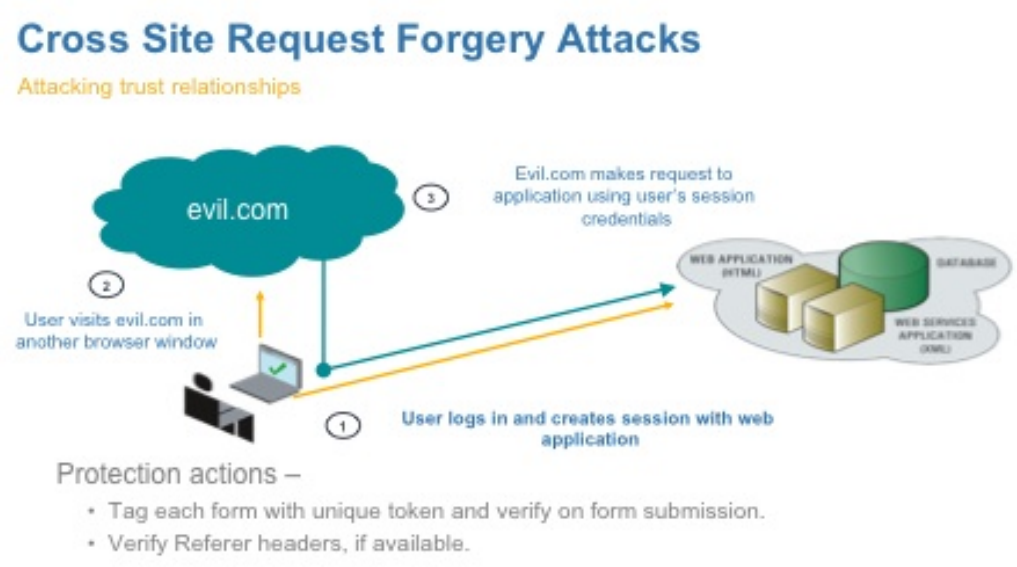
Comment éviter le XSRF

- utiliser un champ ID secret

```
<form method="GET" action="/sendmail">
  <input name="to"/>
  <input name="body"/>
  <input type="hidden" name="secret" value="<valeur aleatoire>"/>
</form>
```

- Transparent en Java

```
CsrfPreventionFilter
HttpServletResponse#encodeURL()
```



Injection SQL

- Pas spécifique au web, mais fréquent.
- Arrive quand on construit des requêtes SQL avec des données utilisateurs
- Exemple: le patron

```
statement = "SELECT * FROM users WHERE
name ='" + userName + "';"
```

- est instancié par:

```
statement = "SELECT * FROM users WHERE
name ='foo' OR '1' == '1';"
```

- s'en prémunir:
  - ne pas générer de requêtes SQL, utiliser des librairies de plus haut niveau.
  - protéger toutes les données générées par l'utilisateur avant de les insérer dans une requête,
  - jouer avec les permissions de base de données.

Bilan Sécurité

1. En générant du code (HTML, JS), toujours protéger les chaînes de caractères dans les variables.
2. Toujours vérifier les arguments envoyés par les requêtes clients,
3. Ne pas charger des scripts d'un utilisateur tiers non reconnu,
4. Ne pas réinventer la roue (utiliser les bibliothèques standards),
5. Ne pas stocker de mots de passe en clair.

Répartition de charge

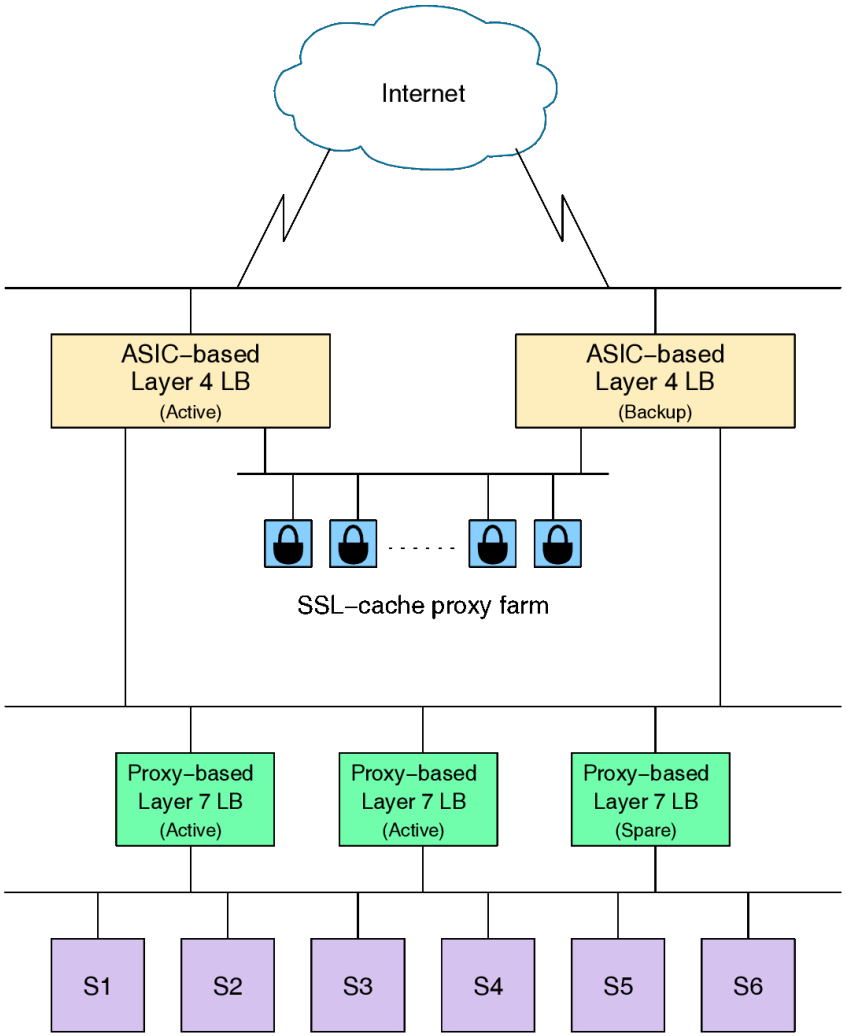
Proposer un même service depuis plusieurs serveurs pour une meilleure disponibilité et une bande passante plus élevée.

- *Round-Robin DNS* (tourniquet DNS),
  - plusieurs adresses IP pour le même nom d'hôte:

```
$ host -t a google.com
google.com. has address 64.233.167.99
google.com. has address 72.14.207.99
google.com. has address 64.233.187.99
```

  - utile pour la répartition géographique.
  - adresse IP stockée en cache.
- Equilibrage de Niveau 3/4(TCP/IP),
- Equilibrage de Niveau 7 (HTTP).
- toutes les requêtes de même origine doivent être traitées par le même serveur réel.:
  - IP de l'expéditeur,
  - cookies (JSESSIONID, PHPSESSIONID, ...).
- répartition facile pour le contenu statique, difficile pour le contenu dynamique.
- Equilibrage transparent sur les sites des hébegeurs (AWS, GAE, MS Azure).

Architecture d'équlibrage



Access :

- Can be redundant using BGP.
- Multiple sites possible with DNS.

Layer 3–4 load balancing :

- Infrastructure Availability and scalability.

Scalable front–end :

- Web acceleration and security : SSL, cache, compression, ...

Layer 7 load balancing :

- Application availability and scalability.
- Persistence, Monitoring, troubleshooting, logging, and content switching.

Applications :

- Running shared or dedicated servers.
- Heavy apps only require more servers.

Optimisation de cache

- Cache navigateur
  - utiliser agressivement le cache pour toutes les ressources statiques.
  - utiliser l'empreinte (*fingerprinting*) d'URL pour utiliser le cache de manière dynamique.
    - l'empreinte change quand la ressource change.
    - toujours servir une ressource depuis le même nom d'hôte.
  - Cache proxy
    - ne pas inclure de chaîne de requête ? pour l'acces aux ressources statiques.
    - ne pas faire de cache proxy pour les ressources qui utilisent des cookies.

Optimisation de Temps de Parcours (*Round-Trip Delay Time*)

- utiliser la réécriture d'URL seulement pour les URL saisies par l'utilisateur.
- grouper les scripts JS en un seul fichier.
- grouper les CSS en un seul fichier.
- grouper les images avec des *sprites* CSS.
- paralléliser les téléchargements sur les différents hôtes.

Optimisation de la Surcharge de Requête

- minimiser la taille de la requête
  - garder des URLs et des chaînes de requêtes courtes.
  - stocker les infromations relatives aux cookies du côté serveur.
  - retirer les cookies inutilisées.
- servir le contenu statique depuis un domaine sans cookie.

Optimisation de la taille du contenu envoyé

- activer la compression,
- minifier le JavaScript,
- reporter (*defer*) le chargement du code JS,
- optimiser les images,

Optimisation du rendu navigateur

- utiliser des sélecteurs CSS efficaces:
  - faire des règles aussi spécifiques que possible,
  - retirer les redondances,
  - utiliser des sélecteurs de classes.
- spécifier les dimensions des images, les jeux de caractères.

Test

- Analyses Statiques:
  - validation HTML et CSS,
  - typage de JS/compilation,
  - typage des programmes serveur,
  - approche intégrée: *Ocsigen*.
- Test unitaires pour chaque composant (JUnit, HTTPUnit).
- Test intégré (avec vue).
- Test de sécurité, test de performance.
- Selenium: cadriciel de test libre pour les applications web.
  - écriture de script de test dans un langage dédié,
  - doit être lancé avec plusieurs navigateurs.

Exemple de test Selenium

```
require_once 'PHPUnit/Extensions/SeleniumTestCase.php';

class CategoryModifTest extends PHPUnit_Extensions_SeleniumTestCase
{
  protected function setUp()
  {
    $this->setBrowser("*firefox");
    $this->setBrowserUrl("http://...");
  }

  public function testCategoryModif()
  {
    $this->open("http://...");
    $this->type("modlgn_username", "admin");
    $this->type("modlgn_passwd", "password");
    $this->click("link=Connexion");
    $this->waitForPageToLoad("30000");
    $this->open("http:.../administrator/index.php?...");
    $this->waitForPageToLoad("30000");
    $name = $this->getTable("//div[@id='element-box']/div[2]/form/table.2.2");
    $this->click("link=".$name);
    $this->waitForPageToLoad("30000");
    $this->type("name", "Ordinateurs portables modifié");
    $this->click("//td[@id='toolbar-save']/a/span");
    $this->waitForPageToLoad("30000");
    try {
        $this->assertTrue($this->isTextPresent("Ordinateurs portables modifié"));
    } catch (PHPUnit_Framework_AssertionFailedError $e) {
        array_push($this->verificationErrors, $e->toString());
    }
    $this->click("link=Ordinateurs portables modifié");
    $this->waitForPageToLoad("30000");
    $this->type("name", "Ordinateurs portables");
    $this->click("//td[@id='toolbar-save']/a/span");
    $this->waitForPageToLoad("30000");
    $this->click("link=Déconnexion");
    $this->waitForPageToLoad("30000");
  }
}
```

Applications Mobiles

Les Applications Mobiles sont typiquement des applications client/serveur.

- la partie client est développée avec des kits de développement propriétaires: Android SDK (Java), iOS SDK (Objective-C).
- les applications mobiles utilisent le même genre de protocoles de communication client/serveur.
  - SOAP, XML-RPC, JSON-RPC, ...
- beaucoup d'applications mobiles partagent le côté serveur avec une application web.
- développer mombile à partir du web:
  - utiliser le navigateur mobile,
  - Apache Cordova: applications hybrides mobiles en HTML 5/CSS 3,
  - Firefox OS: SE mobile basé sur les standards du Web.