Cahier des charges V1.0 (bêta)

***webtuts.fr***

# Général

## Sujet :

Webtuts sera un blog présentant des tutoriels solutionnant des problèmes concrets sur la réalisation de son propre site web avec diverses astuces et techniques expliquées de façon détaillés pour répondre aux besoins de tous.

Le développement de ce même blog sera, le plus souvent possible, prit comme exemple à ces tutoriaux.

## Equipe :

* **Quentin Deneuve :** Chef de projet
* **Richard Ettou :** Directeur technique / Ergonome / Développeur
* **Thibault Dulon :** Responsable communication, marketing et référencèrent
* **Jonathan Bicheux :** Rédacteur en chef

## Nom :

* Webtuts.fr

FAQ

Architecture du développement du site :

**Le kernel, l’app**

o    fichier de routage (quoi est ou)

o     fichier d’identifiants SQL etc. (config.php)

o    récupération des librairies

**Les libraries (contient toute les librairies nécessaires pour tout site)**

o    Sql.class.php (BDD quoi)

o    Std.class.php (class par défaut (si la classe demandée n’existe pas) et parente des classes du modèle (toute les fonctions de bases quoi get(), set(), create(), ..))

o    Collection.class.php (utiliser par Sql.class.php pour retourner des tableaux d’objet)

o    Session.class.php

o    etc ….

o    modele/ toutes les classes du modèle dont le app.class.php

**Le dossier “theme” il va afficher le contenu grâce aux variables fournies par le kernel(modele)**

**o**  actions/ un fichier par action sur le thème

o    et tout le reste a votre guise (css/ img/ pages/ partials/ …) Tu récupères les données simplement, genre App::getClass(“user”, 1) pour l’user d’id 1 etc … ou App::getClass(“user”, 1)->get(“name”), pour son nom etc... Si l’attribut est un Objet, l’objet vous est retourné de même pour une collection d’objet.

Donc App::getClass(“article”, 1)->get(“categories”)->get(0)->get(“name”);

o    requête SQL :

Sql::select()

->from(“article”)

    ->where({

           “autor” => “=Tom”

})

->orderBy(“id”, “DESC”)

->limit(0,5)

->fetchArray(); // fetchOne(); pour un seul résultat

    Renvoi un objet Collection avec tous les résultats

## Mutualisation du code source :

Nous utiliserons pour le partage de code source le logiciel GIT et son serveur gratuit Github.com.  
Une fois installé, vous lancez git Gui. Là vous sélectionnez "cloner un dépôt existant".

Dans emplacement source vous copier-coller ça : https://github.com/fitzlucassen/Webtuts.fr.git  
Dans répertoire cible vous mettez le chemin où sera entreposé le projet sur votre PC (le répertoire ne doit pas être déjà crée). Et puis cliquez sur "cloner".

Une fois que le chargement est fini vous pouvez quitter Git GUI.  
Vous lancez ensuite Git Bash et grâce à la commande "cd" vous allez dans le répertoire du projet.  
Ensuite c'est simple. Pour ajouter des fichiers ou commit une nouvelle version de certain fichiers existant :  
1) git add .  
2) git commit -m 'mon commentaire de commit'  
3) git push origin master  
Pour récupérer la dernière version des fichiers qui sont sur le serveur :  
- git pull origin master

## Front end :

* ***Le header***

Le header contient la navigation (“Catégories”, “Articles”, “Actualités”, “Recherche”, “Membre” (Seulement si connecté), “Contact”), un mini formulaire de connexion/déconnexion, un <input> de recherche rapide

* ***Le footer***

Le footer est décomposé en deux footers. Le premier, plus haut que le deuxième, contient une rubrique vers les liens utiles/partenaires (grafikart, alsacreation ...etc.), la deuxième rubrique contient un lien vers les réseaux sociaux avec un court texte du genre “Rejoignez notre actualité sur le web”, la troisième rubrique contient un sitemap pour les liens utiles du site comme la page contact, la page “A propos” ...etc.  
Le deuxième footer contient les copyrights.

* ***Page d’accueil***

**Url : webtuts.fr/**  
  
La page affiche un titre de bienvenue,  un texte de résumé de l’activité ou un carrousel avec quelques images et phrases accrocheuse, un feedback des dernières actualités en sidebar, un feedback des actus twitter et/ou facebook dans la  sidebar, les 5 derniers articles postés en contenu principal.

* ***Page de profil***

**Url : webtuts.fr/membre-{login}**  
  
La page affiche un récapitulatif des informations de la personne ainsi que la possibilité de les modifier.  
Cette page permet de se désinscrire et de se délogger.  
(cf. Fonctionnalité connexion/déconnexion/désinscription)

* ***Page actualités***

**Url : webtuts.fr/actualite.html**  
  
La page affiche les actualités sous forme de time-line. Donc trier par date de la plus récente à la plus ancienne. Système de chargement de news dynamique OU système de pagination dynamique.  
Affichage du titre des 140 premiers caractères, de la date de publication, et éventuellement d’une image en thumbs. On peut aussi afficher un compteur du nombre de commentaire sur cette news.

* ***Page actualité***

**Url : webtuts.fr/actualite-{titre-actualite}**  
  
La page s’affiche avec quasiment le même template que la page article (cf. Page Article). A la différence prêt qu’une actualité n’a pas de tag.

* ***Page Recherche***

**Url : webtuts.fr/recherche.html**  
  
La page propose un formulaire de recherche complet. Textbox pour une recherche textuelle, des checkbox personnalisées pour les catégories dans lesquelles rechercher, des checkbox personnalisées pour les tags recherchés, des checkbox personnalisées pour le type de la recherche (d’un article, d’une actualité, ou les deux). Je propose un “Search As You Type” sur l’input (mais uniquement sur les titres des articles sinon la requête sera trop lourde).

* ***Page Résultat recherche***

**Exemple url : webtuts.fr/recherche-{motif-recherche}/tag~{nom-tag}~{nom-tag2}/categorie~{nom-categorie}**  
  
La page propose selon le même template que la page catégorie, ou la page articles (cf. Page Categorie) une liste des résultats à la différence prêt que les mots présent dans le motif de la recherche seront surlignés ou mis d’une couleur différente sur la liste des résultat afin de visualiser aisément la raison de la présence de tel ou tel résultat dans la liste. Si la requête ne retourne rien le titre de la page change avec un lien re retour vers le formulaire de recherche. Il serait bon aussi de différencier les résultats étant des articles des résultats étant des actualités, via une couleur de fond différente ou autre.  
Je propose qu’on affiche sur le haut de la page sous forme de **<div>** cliquables le nom de chaque filtre appliqué à la recherche afin que l’utilisateur puisse facilement enlevé le filtre qu’il veut s’il juge sa recherche trop précise.

* ***Pages Contact***

**Url : webtuts.fr/contact.html**  
  
La page contact propose un lien vers nos réseaux sociaux : Facebook et Twitter minimum.  
Elle propose aussi un formulaire avec un court texte d’introduction, puis avec les champs “Nom” (pré-remplie par les données de $\_SESSION[‘user’][‘name’]), “Sujet”, “Message”.

* ***Pages Catégories***

**Url : webtuts.fr/categorie.html**  
  
La page catégories affiche une liste de toutes les catégories avec un logo par catégorie et le nombre d’article étant actuellement dans cette catégorie.

* ***Pages Catégorie***

**Url : webtuts.fr/categorie-{nom-de-la-categorie}**  
  
La page catégorie propose une liste de tous les articles correspondant à cette catégorie en mentionnant le titre de l’article, les 140 premiers caractères, la date de publication, l’auteur, le nombre de commentaire de l’article, une image en thumbs, les tags associé à cet article.  
Je propose un fils d’Ariane sur le haut de la page. C’est SEO friendly et ça aide l’utilisateur à se repérer/naviguer.  
  
*Exemple de catégorie : “HTML/CSS”, “JQuery”, “PHP” ...etc.*

* ***Pages Articles***

**Url : webtuts.fr/article.html**  
  
La page articles affiche la liste de tous les articles selon le même template que la page catégorie (cf. Page Categorie). Je propose une pagination dynamique avec une liste de dix articles par page un peu comme le système de commentaire (cf. Partie Système de commentaire).

* ***Pages Article***

**Url : webtuts.fr/categorie-{nom-de-la-categorie}/article-{nom-de-l-article}**  
  
La page article affiche le titre de l’article ainsi que son contenu (texte, image...etc.)  
A la fin de l’article, un système de commentaire est proposé. Pour poster un commentaire, je propose que l’utilisateur soit loggé pour éviter le SPAM et le contenu indésirable. Ainsi on propose une <textarea> s’il est loggé et rien s’il n’est pas loggé.  
Après cela, on affiche tous les commentaires déjà posté sur l’article. (cf. Partie Système de commentaire)  
  
Je propose un fils d’Ariane sur le haut de la page. C’est SEO friendly et ça aide l’utilisateur à se repérer/naviguer.  
De plus, il faut ajouter en haut de la page les tags de l’article (cf. Partie Tag)

* ***Page tag***

**Url : webtuts.fr/tag-{nom-du-tag}**  
  
Cette page affiche le nom du tag ainsi que le listing de tous les articles comportant ce tag de la même façon que sur la page catégorie (cf. Page Categorie).

* ***Tags***

Un tag contiendra un mot clef identifiant un article. Un article pourra avoir plusieurs mots clefs. Pour une exploitation de ces tags plus aisée, il faut les stocker en BDD en séparant par un caractère spécial. La virgule me parait appropriée.  
Lors de son affichage sur la page article, le tag est un lien qui pointe vers la page tag. (cf. Page tag).  
  
*Exemple de tags : “Header”, “Footer”, “Sidebar”, “Navigation”, “Carrousel”*

* ***Gestion des langues***

Pour cela il faudra penser à plusieurs choses :  
Tout d’abord, penser au fait que tout le contenu personnalisable (article et autres pages de contenu) soit en base de données en français ET en anglais. Pour ça soit un attribut “content\_fr” et un attribut “content\_en” soit un attribut à part entière “lang” qui définit la langue de la ligne dans la table.  
Il faut ensuite penser aux fichiers de constantes du genre fr.php et en.php qui contiendront la définition de tous les mots clefs du site du genre “nom”, “prenom”, “commentaire”, “Bienvenue” ...etc. Le bootstrap.php du site s’occupera de charger le bon fichier de constante en fonction de la $\_SESSION[‘lang’].

* ***Système de commentaires***

Je propose un ratio de 10 commentaire par page et que la gestion du multipage des commentaires soit géré en AJAX de sorte à ce que seul le contenu du conteneur de commentaire se rafraîchisse.

* ***Inscription/Connexion/Désinscription***

L’inscription est à contrôler par validation d’adresse e-mail.  
On demandera le nom, le prenom, le pseudo, le mot de passe (plus une confirmation du mot de passe), l’adresse e-mail et une checkbox demandant si l’utilisateur souhaite rester informer de l’actualité du site.  
Si on demande trop d’information ça effraie les utilisateurs.  
Toutes les données de l’utilisateur sont stockées en $\_SESSION[‘user’]  
Le mini formulaire de connexion apparait dans le header. La Déconnexion ainsi que la désinscription sont disponibles sur la page membre (cf. Page profil).  
En cas de déconnexion, on redirige vers la page d’accueil. En cas de désinscription on demande confirmation par email. Une fois l’email confirmé, on effectue le DELETE.

* ***Abonnement flux Rss***

## Backend :

Le back office étant un élément privé du site web, la réecriture d’URL  n’est pas nécessaire.

* ***Gestion des paramètres globaux***

Une page proposant un formulaire de modification des balises **<meta>** “description” et “keywords” ainsi que le contenu de la balise **<title>** de chaque page.

* ***C.R.U.D Catégorie/Article/Tag/Actualité***

Il faudra pour les Catégories, les articles et les actualités, une page formulaire qui permettent d’ajouter en base de données une entité, et une page listant toutes les entités avec un bouton “Supprimer” et un bouton “Modifier”.  
Le listing n’affichera que le titre de l’entité ainsi, peut-être, que l’image de preview.  
Le boutons “supprimer” affichera une pop-in de confirmation. Si oui on effectue le DELETE, si non on retourne à la page de listing.  
Le bouton “modifier” redirige vers une page de formulaire pré-remplie par les informations de l’entité.  
Pour la gestion des dates on peut se servir du plugin jquery “DatePicker” pour les formulaires.

* ***C.R.U.D des membres et gestion des droits***

Pour le C.R.U.D on procèdera de la même manière que pour le C.R.U.D des Catégorie/Articles/Tags/Actualités. (cf. C.R.U.D Catégorie/Article/Tag/Actualité). On remplacera simplement le bouton “Modifier” et son lien par un bouton “Droits” qui redirige vers une page différente où l’on pourra donner les droits que l’on veut à un utilisateur à savoir :  
  
*Exemple de droit : “Accès au back-office”, “Poster des Commentaires”, “Lire les articles”, “Ecrire des articles” (pour la V2).*

* ***Modération des commentaires (suppression/validation)***

Encore une fois, on procèdera de la même manière que pour le C.R.U.D des Catégorie/Articles/Tags/Actualités. (cf. C.R.U.D Catégorie/Article/Tag/Actualité). On remplacera simplement le bouton “Modifier” par “Valider” et on ne pourra pas ajouter de commentaire via le back-office.  
Le bouton supprimer supprimera le commentaire via confirmation, le bouton valider mettra en ligne le commentaire via un “update table” d’un attribut “online”.

* ***Création d'un BB code pour mise en page article avec prévisualisation en direct***
* ***Google analytics***
* ***Gestion des themes***

## Rédaction :

* ***Élaboration d’un article***

On réfléchit tous au sujet de l'article, on donne tous du contenu, mais c'est Jonathan qui l'écrit (Rédacteur en chef). Le nom de l’auteur figurant sur l’article sera la personne initiatrice de l’idée.

* ***Fréquence des articles***

Il faudra assurer au minimum deux articles par mois qui ne seront pas forcément tous très longs mais tous traduit en anglai et illustrer.

* ***Longueurs et contenu***

Le plus succinct possible tout en étant clair et précis. Affichage du code pas à pas avec ses explications, affichage d’images éloquentes, présence d’une page de démo pour chaque article

* ***Une démo finale par article pour illustrer ce dernier***

Il faudra penser dans l’architecture à un dossier “demo” qui contiendra tous les fichiers html/css/php/js des démonstrations des articles.  
  
Technique et matériel :

* **Base De Données :** MySQL
* **Vidéos :** Youtube ou Vimeo
* **Images :** On les hébergera sur notre serveur. Le tout sera de bien classer toutes les images par dossier de catégorie.
* **Mutualisation du code source et gestion de l’historique :** GitHub

## Marketing :

* ***Réseaux sociaux***

Il faudra créer une page Facebook (page fan) et un compte Twitter au minimum.

* ***Référencèrent***

- Il faut que toutes les balises **<title>** est un nom éloquent et résume exactement le contenu de la page cible.  
- Il faut que toutes les balises **<meta>** soient remplies dans les balises **<head>**.  
- Il faut que toutes les balises **<img>** aient un attribut “alt” éloquent qualifiant exactement le contenu de l’image ou de ce qu’elle illustre.  
- Il faut préférer les balises sémantiques telles que **<h1>,** **<h2>**, **<ul><li>** ...etc. Car les robots de Google connaissent leur sémantique alors que les balises **<div>** et **<span>** ne traduisent rien de spécial.  
- Il faut avoir recours à la réécriture d’URL (définie pour chaque page) car une URL lisible, compréhensible et éloquente donne plus d’informations aux robots de Google sur la page voulu. Pour cela il faudra une table en base de données “url\_rewriting” qui contiendra le chemin de chaque page et l’url qui lui correspond.  
- Il faut utiliser les balises html5 **<header><footer><aside><nav>** combinées avec html5shiv.js pour la compatibilité crossbrowser.  
- Il faut que le **DOM** de chaque page commence par la navigation de lien puis une balise **<h1>** de titre. Peu importe si, par la suite, le style fasse que la navigation se retrouve en seconde place et que la balise **<h1>** soit complètement stylisée et à une autre place. En effet, les robots visitent le site sans aucun style (désactivation du css). Les pages les plus indexées sont les pages qui contiennent en premier lieu dans le **DOM** des titres éloquent des paragraphes de résumé éloquent, et beaucoup de liens (là aussi, peu importe si le javascript prend le dessus sur le lien dans notre vrai site du moment que le “href” est renseigné, le robot désactive le javascript).

* ***Newsletter***

Pour chaque nouvel article ou nouveauté majeure du site. On se permettra d’envoyer un e-mail aux personnes ayant souscrit à la newsletter. Pour ça il faut développer un template d’email.

* ***Réseaux sociaux intégrés***

Il faudra intégrer la possibilité de se connecter au blog via Facebook Connect.  
Il faudra aussi ajouter un like button et un tweet button sur la page contact et sur la page d’accueil afin de donner de la visibilité à nos fils d’actualités sociaux.  
Il faudra donner la possibilité aux lecteurs des articles de pouvoir les partager sur tweeter et sur facebook via un facebook sharing.  
Enfin, il faudra intégrer un feedback de la page facebook et de la page tweeter sur la page d’accueil.

* ***Visibilité (Open Graph)***

## Charte graphique :

* **Site plutôt clair**
* **Plutôt aéré, mais pas trop, comme un blog classique**
* **Affichage des articles comme sur un blog classique**
* ***Fonts***

Il en faut plusieurs (au moins une pour les titres, une pour le contenu et une pour la navigation ou les mots clefs ou autres).

* + **Titre :** ‘Jack-font’, Verdana, Helvetica, Georgia
  + **Contenu :**  ‘Ubuntu-C’, Arial, Sans-Serif
  + **Autre :** 'Lucida Grande', Tahoma,
* ***Couleurs du site***

Je propose un site à dominante de blanc avec des soulignements et des bordures bleu. Les titres serait quasi noir et le contenu grisâtre.  
  
Ergonomie :

* ***Version mobile (à voir pour la V2) (Responsive design)***
* ***Dimensions***

Le site aura de base un largeur maximum 1024px de large. Mais il faudra prévoir une version en 640px et une version 320px (responsive design) grâce aux mediaqueries.

* ***Détails ergonomique de développement***

Il faudrait favoriser au maximum les “font-size” en “em” et les tailles en “%”. De plus, l’utilisation des “float” pour les positionnements des blocks est fortement recommandée. Cela permettra un passage en responsive design beaucoup plus aisé. Il faudrait aussi éviter les grandes images redimensionnées (pour rendre la navigation plus agréable et éviter des ralentissements inutiles).