Architecture d'un site web : manipulation des données

Alexandre Niveau

GREYC — Université de Caen

Un petit point

- Rappels sur notre architecture
  - modèle : logique métier. API séparée de l'affichage et du contrôle.
  - vue : gestion de l'affichage HTML en fonction de l'état du modèle.
  - contrôleur : modifie le modèle et construit la vue adaptée en fonction des actions de l'internaute.
  - routeur : interprète les requêtes HTTP comme des actions, appelle le contrôleur, et affiche la vue.
- Appliquée sur l'exemple très simple des poèmes :
  - le routeur regarde le poème demandé dans l'URL
  - il demande au contrôleur de s'en occuper
  - le contrôleur récupère le poème dans le modèle
  - et le passe à la vue

CRUD

- Les sites web interactifs doivent faire beaucoup plus que ça
- Opérations de base sur les objets : CRUD, *create*, *read*, *update*, *delete*
- On va présenter comment appliquer notre architecture à ces problèmes avec l'exemple d'un site où les internautes peuvent proposer des noms à des couleurs

Règle d'or

> Tout ce qui vient du client est **dangereux**, il faut systématiquement s'en méfier.

Affichage

- La règle d'or nous dit qu'il ne faut pas afficher n'importe comment les données qui viennent (potentiellement) du client.
- L'internaute peut mettre du HTML (et donc du JavaScript) dans ses données : les afficher directement, c'est compromettre tous les utilisateurs !

- Pour afficher des données qui viennent des clients dans une page HTML, il faut échapper les caractères spéciaux du HTML en les remplaçant par des entités HTML : `<` par `<`, `&` par `&`, etc.
- La fonction PHP `htmlspecialchars` s'occupe de ça, mais il faut préciser certaines options :

```php
<?php
$texte = "Du texte & de l'<abbr>HTML</abbr>.";
$utf8inv = 'Un caractère "'.hex2bin("C0").'" invalide.';
echo "1. ".htmlspecialchars($texte)."\n";
echo "2. ".htmlspecialchars($utf8inv)."\n";
echo "3. ".htmlspecialchars($texte, ENT_QUOTES|ENT_SUBSTITUTE|
ENT_HTML5, 'UTF-8')."\n";
echo "4. ".htmlspecialchars($utf8inv, ENT_QUOTES|ENT_SUBSTITUTE|
ENT_HTML5, 'UTF-8')."\n";
?>
```

```
1. Du texte &amp; de l&#039;&lt;abbr&gt;HTML&lt;/abbr&gt;.
2. Un caractère &quot;&#65533;&quot; invalide.
3. Du texte &amp; de l&apos;&lt;abbr&gt;HTML&lt;/abbr&gt;.
4. Un caractère &quot;&#65533;&quot; invalide.
```

1. Du texte & de l'<abbr>HTML</abbr>. 2. Un caractère "" invalide. 3. Du texte & de l'<abbr>HTML</abbr>. 4. Un caractère "" invalide.

- `ENT_QUOTES` encode les apostrophes, `ENT_SUBSTITUTE` remplace les séquences UTF-8 invalides par le caractère (U+FFFD) plutôt que de retourner une chaîne vide, et `ENT_HTML5` utilise la table d'encodage d'HTML5.

Échappement à l'entrée

Échappement à l'entrée : échec. (Source [https://instagram.com/p/SVfQruppEE/])

- Attention, la façon d'échapper la sortie dépend de ce qu'on affiche.
- Par ex., pour une sortie en mode texte, on ne veut pas échapper les caractères spéciaux.
- ⇒ il ne faut **pas** échapper les données en entrée, (et les stocker par exemple dans une base de données) : on ne sait pas ce qu'on voudra en faire plus tard.
- Un article sur le sujet [http://lukeplant.me.uk/blog/posts/why-escape-on-input-is-a-bad-idea/]

Structure des répertoires

```
src/
|-- Router.php
|-- ctl/
|   `-- Controller.php
|-- lib/
|   `-- ObjectFileDB.php
|-- model/
|   |-- Color.php
|   |-- ColorStorage.php
|   `-- ColorStorageFile.php
`-- view/
    `-- MainView.php
```

- On retrouve la même structure que précédemment
- La classe `Color` représente une couleur. On peut modifier son nom et sa valeur hexadécimale. On peut récupérer d'autres informations, comme les dates de création/modification et différents formats de couleur.
- L'interface `ColorStorage` gère le stockage des couleurs enregistrées. C'est elle qui est manipulée par les autres composants, sans qu'ils ne connaissent l'implémentation réelle.
- La classe `ColorStorageFile` est une implémentation de `ColorStorage`, qui n'utilise pas de BD mais un simple fichier texte (manipulé à l'aide de la classe `ObjectFileDB` [file_store.php]). Pour le reste de l'application, ça ne change rien (puisque c'est l'interface qui est utilisée).
- Le reste est identique à ce qui a été vu en CM/TP.

Le modèle

- Pour illustrer la séparation des responsabilités, on va considérer que les classes `Color` et `ColorStorageFile` ainsi que les feuilles de style ont été écrites par d'autres. On ne les modifiera pas.

Classe `Color`

```php
<?php

/* Représente une couleur. */
class Color {

    protected $name;
```

```
    protected $hex;
    protected $creationDate;
    protected $modifDate;

    /* Construit une couleur. Si les paramètres de date ne sont pas passés,
     * la couleur est considérée comme étant toute nouvelle.
     * Le nom et le code hexa doivent être valides, au sens
     * de isNameValid et isHexValid, sinon une exception est levée. */
    public function __construct($name, $hex, $creationDate=null, $modifDate=nu
        if (!self::isNameValid($name))
            throw new Exception("Invalid color name");
        $this->name = $name;
        if (!self::isHexValid($hex))
            throw new Exception("Invalid hex");
        $this->hex = $hex;
        $this->creationDate = $creationDate !
== null? $creationDate: new DateTime();
        $this->modifDate = $modifDate !== null? $modifDate: new DateTime();
    }

    public function getName() {
        return $this->name;
    }

    /* Renvoie le code hexadécimal de la couleur,
     * sous forme de chaîne d'exactement 6 chiffres hexa. */
    public function getHex() {
        return $this->hex;
    }

    /* Renvoie le code RGB de la couleur sous forme de tableau [r, g, b]
     * d'entiers entre 0 et 255. */
    public function getRGB() {
        return array(
            base_convert(substr($this->hex, 0, 2), 16, 10),
            base_convert(substr($this->hex, 2, 2), 16, 10),
            base_convert(substr($this->hex, 4, 2), 16, 10),
        );
    }

    /* Renvoie le code HSL (teinte, saturation, luminosité)
     * de la couleur sous forme de tableau d'entiers [h, s, l],
     * avec h entre 0 et 359, et s et l entre 0 et 100.  */
    public function getHSL() {
        /* adapté de http://www.easyrgb.com/index.php?X=MATH&H=18 */
        $rgb = $this->getRGB();
        $r = $rgb[0] / 255;
        $g = $rgb[1] / 255;
        $b = $rgb[2] / 255;
        $min = min($r, $g, $b);
        $max = max($r, $g, $b);
        $delta = $max - $min;

        /* luminosité */
        $l = ($max + $min) / 2;

        if ($delta == 0) {
            /* c'est du gris */
            $h = 0;
            $s = 0;
        } else {
```

```
            /* saturation */
            if ($l < 0.5) {
                $s = $delta / ($max + $min);
            } else {
                $s = $delta / (2 - $max - $min);
            }

            /* teinte */
            $dr = (($max - $r) / 6 + $delta / 2) / $delta;
            $dg = (($max - $g) / 6 + $delta / 2) / $delta;
            $db = (($max - $b) / 6 + $delta / 2) / $delta;
            if ($r == $max) {
                $h = $db - $dg;
            } else if ($g == $max) {
                $h = 1/3 + $dr - $db;
            } else if ($b == $max) {
                $h = 2/3 + $dg - $dr;
            }

            /* normalisation */
            if ($h < 0)
                $h += 1;
            if ($h > 1)
                $h -= 1;
        }
        return array(round($h*360), round($s*100), round($l*100));
    }

    /* Renvoie un objet DateTime correspondant à
     * la création de la couleur. */
    public function getCreationDate() {
        return $this->creationDate;
    }

    /* Renvoie un objet DateTime correspondant à
     * la dernière modification de la couleur. */
    public function getModifDate() {
        return $this->modifDate;
    }

    /* Modifie le nom de la couleur. Le nouveau nom doit
     * être valide au sens de isNameValid, sinon
     * une exception est levée. */
    public function setName($name) {
        if (!self::isNameValid($name))
            throw new Exception("Invalid color name");
        $this->name = $name;
        $this->modifDate = new DateTime();
    }

    /* Modifie le code hexadécimal de la couleur.
     * Le nouveau code doit
     * être valide au sens de isHexValid, sinon
     * une exception est levée. */
    public function setHex($hex) {
        if (!self::isHexValid($hex))
            throw new Exception("Invalid hex");
        $this->hex = $hex;
        $this->modifDate = new DateTime();
    }
```

```
    /* Indique si $name est un nom valide pour une couleur.
     * Il doit faire moins de 30 caractères,
     * et ne pas être vide. */
    public static function isNameValid($name) {
        return mb_strlen($name, 'UTF-8') < 30 && $name !== "";
    }

    /* Indique si $hex est un code hexadécimal valide.
     * On accepte uniquement 6 chiffres hexadécimaux
     * (mais peu importe la casse) */
    public static function isHexValid($hex) {
        return preg_match("/^[0-9a-f]{6}$/i", $hex);
    }

}

?>
```

Classe `ColorStorageFile`

```
<?php

require_once("lib/ObjectFileDB.php");
require_once("model/Color.php");
require_once("model/ColorStorage.php");



/*
 * Gère le stockage de couleurs dans un fichier.
 * Plus simple que l'utilisation d'une base de données,
 * car notre application est très simple.
 */

class ColorStorageFile implements ColorStorage {

    /* le ObjectFileDB dans lequel l'instance est enregistrée */
    private $db;

    /* Construit une nouvelle instance, qui utilise le fichier donné
     * en paramètre. */
    public function __construct($file) {
        $this->db = new ObjectFileDB($file);
    }

    /* Insère une nouvelle couleur dans la base. Renvoie l'identifiant
     * de la nouvelle couleur. */
    public function create(Color $c) {
        return $this->db->insert($c);
    }

    /* Renvoie la couleur d'identifiant $id, ou null
     * si l'identifiant ne correspond à aucune couleur. */
    public function read($id) {
        if ($this->db->exists($id)) {
            return $this->db->fetch($id);
        } else {
            return null;
        }
```

```
    }

    /* Renvoie un tableau associatif id => Color
     * contenant toutes les couleurs de la base. */
    public function readAll() {
        return $this->db->fetchAll();
    }

    /* Met à jour une couleur dans la base. Renvoie
     * true si la modification a été effectuée, false
     * si l'identifiant ne correspond à aucune couleur. */
    public function update($id, Color $c) {
        if ($this->db->exists($id)) {
            $this->db->update($id, $c);
            return true;
        }
        return false;
    }

    /* Supprime une couleur. Renvoie
     * true si la suppression a été effectuée, false
     * si l'identifiant ne correspond à aucune couleur. */
    public function delete($id) {
        if ($this->db->exists($id)) {
            $this->db->delete($id);
            return true;
        }
        return false;
    }

    /* Vide la base. */
    public function deleteAll() {
        $this->db->deleteAll();
    }
}

?>
```

MVCR initial

Routeur

```
<?php

require_once("model/ColorStorage.php");
require_once("view/MainView.php");
require_once("ctl/Controller.php");


class Router {

    public function __construct(ColorStorage $colordb) {
        $this->colordb = $colordb;
    }

    public function main() {
        $view = new MainView($this);
        $ctl = new Controller($view, $this->colordb);
```

```php
        /* Analyse de l'URL */
        $colorId = key_exists('couleur', $_GET)? $_GET['couleur']: null;
        $action = key_exists('action', $_GET)? $_GET['action']: null;
        if ($action === null) {
            /* Pas d'action demandée : par défaut on affiche
               * la page d'accueil, sauf si une couleur est demandée,
               * auquel cas on affiche sa page. */
            $action = ($colorId === null)? "accueil": "voir";
        }

        try {
            switch ($action) {
            case "voir":
                if ($colorId === null) {
                    $view->prepareUnknownActionPage();
                } else {
                    $ctl->colorPage($colorId);
                }
                break;

            case "galerie":
                $ctl->allColorsPage();
                break;

            case "accueil":
                $ctl->homePage();
                break;

            default:
                /* L'internaute a demandé une action non prévue. */
                $view->prepareUnknownActionPage();
                break;
            }
        } catch (Exception $e) {
            /* Si on arrive ici, il s'est passé quelque chose d'imprévu
               * (par exemple un problème de base de données) */
            $view->prepareUnexpectedErrorPage($e);
        }

        /* Enfin, on affiche la page préparée */
        $view->render();
    }

    /* URL de la page d'accueil */
    public function homePage() {
        return ".";
    }

    /* URL de la page de la couleur d'identifiant $id */
    public function colorPage($id) {
        return ".?couleur=$id";
    }

    /* URL de la page avec toutes les couleurs */
    public function allColorsPage() {
        return ".?action=galerie";
    }

}

?>
```

Vue

```php
<?php

require_once("Router.php");
require_once("model/Color.php");

class MainView {

    protected $router;
    protected $style;
    protected $title;
    protected $content;

    public function __construct(Router $router) {
        $this->router = $router;
        $this->style = "";
        $this->title = null;
        $this->content = null;
    }


    /
**********************************************************************************
    /
* Méthodes de génération des pages                                                *
    /
**********************************************************************************

    public function prepareHomePage() {
        $this->title = "Proposez vos couleurs !";
        $this->content = "Bienvenue sur ce site de partage de couleurs.";
    }

    public function prepareColorPage($id, Color $c) {
        $cname = self::htmlesc($c->getName());
        $chex = $c->getHex();
        $crgb = $c->getRGB();
        $chsl = $c->getHSL();
        $cclass = "color$id";
        $cdatec = self::fmtDate($c->getCreationDate());
        $cdatem = self::fmtDate($c->getModifDate());

        $this->style .= ".$cclass { background-color: #$chex; }";
        $this->title = "La couleur $cname";
        $s = "";
        $s .= "<figure>\n<div class=\"sample $cclass\"></div>\n";
        $s .= "<figcaption>Un échantillon de $cname.</figcaption>\n</
figure>\n";
        $s .= "<p>La couleur $cname a pour code hexadécimal $chex.</p>\n";
        $s .= "<p>Elle contient du rouge à ".round($crgb[0]*100/255)."%, du ve
        $s .= round($crgb[1]*100/255)."%, et du bleu à ".round($crgb[2]*100/25
p>";
        $s .= "<p>Sa teinte est de ".$chsl[0]."°, avec une saturation de ".
$chsl[1];
        $s .= "% et une luminosité de ".$chsl[2]."%.</p>";
        $s .= "<p>Elle a été créée ".$cdatec." et modifiée ".$cdatem."</
p>\n";
        $this->content = $s;
    }
```

```
    public function prepareGalleryPage(array $colors) {
        $this->title = "Toutes les couleurs";
        $this->content = "<p>Cliquer sur une couleur pour voir des détails.</
p>\n";
        $this->content .= "<ul class=\"gallery\">\n";
        foreach ($colors as $id=>$c) {
            $this->content .= $this->galleryColor($id, $c);
        }
        $this->content .= "</ul>\n";
    }

    public function prepareUnknownColorPage() {
        $this->title = "Erreur";
        $this->content = "La couleur demandée n'existe pas.";
    }

    public function prepareUnknownActionPage() {
        $this->title = "Erreur";
        $this->content = "La page demandée n'existe pas.";
    }

    /* Génère une page d'erreur inattendue. Peut optionnellement
     * prendre l'exception qui a provoqué l'erreur
     * en paramètre, mais n'en fait rien pour l'instant. */
    public function prepareUnexpectedErrorPage(Exception $e=null) {
        $this->title = "Erreur";
        $this->content = "Une erreur inattendue s'est produite.";
    }

    /
*************************************************************************
    /
* Méthodes utilitaires                                                  *
    /
*************************************************************************

    protected function getMenu() {
        return array(
            "Accueil" => $this->router->homePage(),
            "Couleurs" => $this->router->allColorsPage(),
        );
    }

    protected function galleryColor($id, $c) {
        $cclass = "color".$id;
        $this->style .= '.'.$cclass.' { background-color: #'.$c-
>getHex().'; }';
        $res = '<li><a href="'.$this->router->colorPage($id).'">';
        $res .= '<h3>'.self::htmlesc($c->getName()).'</h3>';
        $res .= '<div class="sample '.$cclass.'"></div>';
        $res .= '</a></li>'."\n";
        return $res;
    }

    protected static function fmtDate(DateTime $date) {
        return "le " . $date->format("Y-m-d") . " à " . $date-
>format("H:i:s");
    }

    /* Une fonction pour échapper les caractères spéciaux de HTML,
```

```
    * car celle de PHP nécessite trop d'options. */
    public static function htmlesc($str) {
        return htmlspecialchars($str,
            /* on échappe guillemets _et_ apostrophes : */
            ENT_QUOTES
            /* les séquences UTF-8 invalides sont
            * remplacées par le caractère 
            * au lieu de renvoyer la chaîne vide…) */
            | ENT_SUBSTITUTE
            /* on utilise les entités HTML5 (en particulier &apos;) */
            | ENT_HTML5,
            'UTF-8');
    }

    /
*****************************************************************************
    /
* Rendu de la page                                                           *
    /
*****************************************************************************

    public function render() {
        if ($this->title === null || $this->content === null) {
            $this->prepareUnexpectedErrorPage();
        }
        /* On affiche la page.
         * Ici on pourrait faire des echo, mais simplement fermer
         * la balise PHP revient au même, et le HTML est plus lisible.
         * En revanche le code PHP est moins lisible : une autre solution
         * est de mettre ce squelette dans un fichier à part et
         * de simplement faire un «include» (c'est ce qui a été fait pour
         * le site des poèmes). */
?>
<!DOCTYPE html>
<html lang="fr">
<head>
    <title><?php echo $this->title; ?></title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="skin/screen.css" />
    <style>
<?php echo $this->style; ?>
    </style>
</head>
<body>
    <nav class="menu">
        <ul>
<?php
/* Construit le menu à partir d'un tableau associatif texte=>lien. */
foreach ($this->getMenu() as $text => $link) {
    echo "<li><a href=\"$link\">$text</a></li>";
}
?>
        </ul>
    </nav>
    <main>
        <h1><?php echo $this->title; ?></h1>
<?php
echo $this->content;
?>
    </main>
</body>
```

```
</html>
<?php /* fin de l'affichage de la page et fin de la méthode render() */

    }

}

?>
```

Contrôleur

```
<?php

/*** Contrôleur du site des couleurs. ***/

/* Inclusion des classes nécessaires */
require_once("model/Color.php");
require_once("model/ColorStorage.php");
require_once("view/MainView.php");


class Controller {

    protected $v;
    protected $colordb;

    public function __construct(MainView $view, ColorStorage $colordb) {
        $this->v = $view;
        $this->colordb = $colordb;
    }

    public function homePage() {
        $this->v->prepareHomePage();
    }

    public function colorPage($id) {
        /* Une couleur est demandée, on la récupère en BD */
        $color = $this->colordb->read($id);
        if ($color === null) {
            /* La couleur n'existe pas en BD */
            $this->v->prepareUnknownColorPage();
        } else {
            /* La couleur existe, on prépare la page */
            $this->v->prepareColorPage($id, $color);
        }
    }

    public function allColorsPage() {
        $colors = $this->colordb->readAll();
        $this->v->prepareGalleryPage($colors);
    }

}

?>
```

Vue et contrôleur pour CRUD basique

- Voilà donc la structure de laquelle on part. Le résultat est visible ici

[couleurs_ini/], et le code peut être récupéré dans cette archive [couleurs_ini.zip].

- Encore une fois, j'ai utilisé des paramètres d'URL dans un souci de simplicité dans l'implémentation, mais il n'est vraiment pas difficile de mettre en place des URL plus propres en utilisant `$_SERVER['PATH_INFO']`
- On va améliorer progressivement le site en ajoutant une gestion basique des points suivants :
  - suppression d'une couleur
  - création d'une couleur
  - modification d'une couleur

Suppression d'une couleur

- C'est le cas le plus simple : on n'a presque rien à faire
- Le contrôleur supprime la couleur en BD, et indique à l'internaute si tout s'est bien passé
- On va quand même améliorer un peu l'ergonomie en demander à l'internaute de confirmer son choix avant d'effectuer la suppression
- Notre page de suppression sera donc une page normale (accédée en GET), avec un bouton pour confirmer la suppression (qui utilisera POST)

Création/modification d'une couleur

- Création/modification d'une couleur = formulaire
- Les champs du formulaire constituent une nouvelle « interface » pour les couleurs, en plus de celle de la classe `Color` (modèle) et de celle de l'objet en BD (qui est cachée dans `ColorStorage`).
- On ne peut pas se contenter de manipuler des instances de `Color`, car les données issues des formulaires sont potentiellement *incomplètes ou fausses*.
- On va donc manipuler des instances d'une nouvelle classe, `ColorBuilder`, qui va encapsuler le tableau associatif récupéré du client
- Comme son nom l'indique, le travail de `ColorBuilder` est de construire une nouvelle instance de `Color` à partir des données qu'on lui a fournies.

  ➤ `ColorBuilder` peut-il faire confiance aux données fournies ?

Validation des données

- On peut utiliser la validation côté client, mais il ne faut pas oublier la règle d'or ! Ici le corollaire est qu'on va systématiquement revalider les données côté serveur.

  ➤ C'est le `ColorBuilder` qui sait construire une couleur, c'est donc également lui qui s'occupera de valider les données
- En cas de problème, il faut reproposer à l'internaute le formulaire pré-rempli, en lui précisant ce qui ne va pas.

➤ `ColorBuilder` aura des méthodes de validation qui vont renvoyer un tableau avec les erreurs, tableau qui sera utilisé par la vue pour avertir l'utilisateur.

Version finale… pour l'instant

- Le résultat auquel on arrive en suivant cette progression est visible ici [couleurs/], et le code peut être récupéré dans cette archive [couleurs.zip].
- Ça marche… mais c'est encore très basique : plusieurs fragilités au niveau de l'ergonomie (réfléchissez-y !)

Limites de notre approche basique

- Notre site fonctionne, mais il n'est pas très robuste. On va voir à présent comment on peut améliorer son confort d'utilisation.
- Considérer les exemples suivants :
  - Tout content d'avoir créé/modifié sa couleur, Jean-Michel la met dans ses marque-page ou envoie le lien à un ami. Que se passe-t-il ?
  - Dominique crée une couleur puis actualise régulièrement la page, pour vérifier que personne ne la modifie. Que se passe-t-il ?
  - Martine bidouille l'URL de la page de suppression définitive en modifiant l'identifiant : que se passe-t-il ? (Elle peut aussi générer un lien et l'envoyer à quelqu'un.)
- Solution : *POST-redirect-GET*

POST-redirect-GET

- Les pages destinées à être accédées en POST ne doivent pas être visibles directement par les internautes.
- Il faut les rediriger immédiatement vers une page normale.
- La logique est que les utilisateurs ont l'habitude de GET, qui est *idempotente* : la même requête donne toujours le même résultat.
- En redirigeant systématiquement après un POST, on donne à chaque méthode son rôle de base tel que défini par HTTP : POST modifie les données, GET affiche les données.
- La redirection doit utiliser le code de statut `303 See Other` ; on peut utiliser une option de la fonction PHP `header` :

```php
header("Location: " . $url, true, 303);
```

Feedback

- Le POST-redirect-GET marche bien pour la création et la modification (sans erreur), car on redirige vers la page de la couleur
- Pour la suppression, on voudrait pouvoir rediriger vers la galerie avec un *feedback* du type « La couleur a bien été supprimée »
- Nécessite de se souvenir de ce qui s'est passé à la requête précédente : il faut utiliser les variables de session

- Principe :
  - Une variable `$_SESSION["feedback"]` contient le *feedback* de la requête précédente.
  - Au début du traitement, le routeur récupère son contenu et la vide. Ce qui a été récupéré est passé à la vue.
  - Il peut ensuite y mettre toutes les informations utiles, qui seront affichées à la prochaine requête.

Spécifications et normes

- HTTP/1.1: Semantics and content [http://tools.ietf.org/html/rfc7231]

Tutoriels

- The great escapism (or: what you need to know to work with text within text) [http://kunststube.net/escapism/]

Lectures complémentaires

- Why escape-on-input is a bad idea [http://lukeplant.me.uk/blog/posts/why-escape-on-input-is-a-bad-idea/]
- What every programmer needs to know about encodings and charsets to work with text [http://kunststube.net/encoding/] (avec des détails sur PHP)