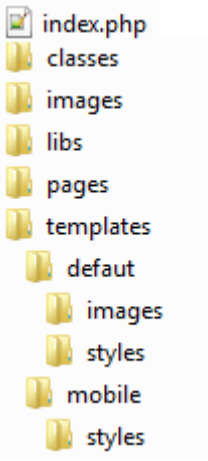
PHP – Moteur de template

Étape 1 : Organisation du framework

Repartir de la correction du TP2 de CSS. Comprendre le code du fichier `index.html` et le découper en 2 fichiers distincts, de façon à externaliser le corps (contenu principal) de la page dans un fichier séparé.
Renommer le fichier `index.html` en `template.php` et le corps de la page en `index.main.php`, puis les placer à l'endroit que vous trouvez le plus approprié dans l'arborescence du projet qui devra ressembler à cela :



Créer le fichier frontal `index.php` (unique point d'entrée à toutes les pages du site) et en utilisant les directives `include` à bon escient, arrangez-vous pour que l'affichage de la page complète fonctionne à l'appel de l'url :
`http://localhost/tp-php/index.php`
Aide : le fichier `index.php` doit inclure le fichier `template.php` qui doit lui-même inclure le fichier `index.main.php`.
Déplacer ensuite les autres ressources (feuille de style, police, images...) au bon endroit et appliquez les corrections nécessaires au code pour obtenir un affichage correct.

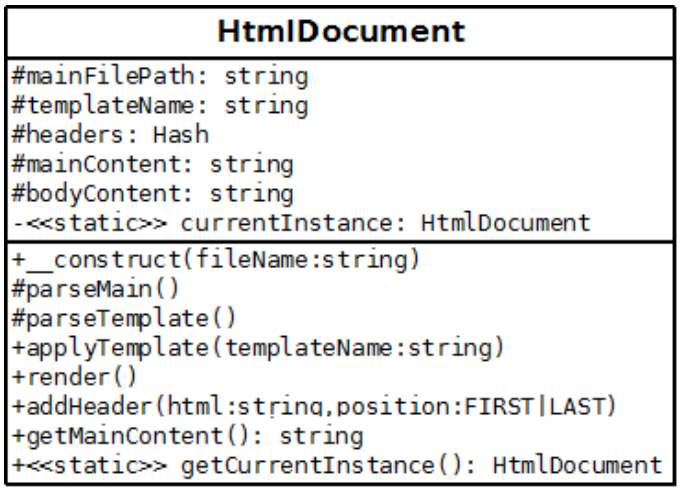
On veut créer une seconde page `contact.main.php` pour proposer un formulaire de contact.
Modifier le fichier `template.php` pour qu'il utilise un paramètre `$_GET['page']` lui indiquant quel contenu de page est à inclure. Les pages « index » et « contact » doivent s'afficher via les URLs respectives :
`http://localhost/tp-php/index.php`
`http://localhost/tp-php/index.php?page=contact`
Modifier le lien vers la page contact (en haut à droite) pour qu'il pointe vers la page créée.

On souhaite définir un titre spécifique pour cette page « contact » (balise title) car actuellemebt, c'est le même titre que pour la page d'accueil qui s'affiche. Il paraît logique de vouloir définir ce titre dans le fichier `contact.main.php` lui-même. Peut-on le faire ? Si oui, expliquez comment ; si non, expliquez pourquoi.

Étape 2 : Classe `HtmlDocument`, le cœur de notre framework

On souhaite regrouper la logique d'habillage de notre contenu au sein d'une classe qui modélisera notre document HTML et le générera.

Créer la structure du fichier `HtmlDocument.class.php` à partir du diagramme de classe ci-dessous (dans un 1er temps, laissez vide le contenu des fonctions/procédures) :



Cette classe est un singleton (un peu particulier). Une instance unique sera créée par le fichier frontal `index.php` et permettra de générer le code HTML correspondant à la page demandée :

```
// Inclusion des classes et librairies utilisées
require 'classes/HtmlDocument.class.php' ;
require 'libs/mobile.lib.php' ;               // Pour is_mobile()

// Valeur par défaut pour la page si non renseignée
$page = isset($_GET['page']) ? $_GET['page'] : 'index' ;
// Appel au constructeur : il lit le corps de la page et le mémorise dans la variable interne $mainContent
$doc = new HtmlDocument($page) ;
// Application du template désiré : on lit le fichier de template (ce dernier utilisera la fonction getMainContent() pour inclure le corps de la page à l'endroit souhaité) et on stoque le résultat dans la variable interne $bodyContent (= ce qu'il y a entre les balises <body> du document HTML)
$doc->applyTemplate( is_mobile() ? 'mobile' : 'defaut' ) ;
// On envoie au navigateur le code HTML complet de la page. C'est à ce stade que l'on génère la balise <html> mais aussi <head> pour inclure les styles CSS entre autre.
$doc->render() ;
```

Bien comprendre les différents attributs et fonctions/procédures de cette classe. Une fois que vous avez compris ce qu'allait contenir chaque attribut et ce qu'allait faire chaque fonction, implémentez-les.

Étape 3 : Utilisation du framework et ajustements

Modifier le fichier `contact.main.php` pour lui définir un titre et lui inclure une feuille de style personnalisée et afficher le bouton de soumission en rouge.

Côté sécurité : que se passe-t-il si on passe la valeur `../../../unFichierQuiExiste` au paramètre GET `page` ? Modifier la classe `HtmlDocument` pour interdire l'inclusion de fichiers situés à l'extérieur de l'arborescence de notre framework (Path Traversal attack).

Afin de palier aux principales attaques connues, on interdira dans le paramètre `$_GET['page']` l'utilisation des 6 caractères `.<>!?:` grâce à la fonction `strpbrk`.