Assosiez à chaque demande du client le design pattern que vous proposez d'utiliser pour répondre à cette demande. Expliquez pourquoi vous avez choisi ce pattern.

Créer des sections à plusieurs colonnes. Une section doit pouvoir être divisée en deux colonnes. Une colonne doit pouvoir être divisée en deux colonnes (Div2Col)

Pattern proposé : Composite

Pourquoi : Le composite permet de mettre en place une structure arborescente de manière à ce que chaque élément de la structure puisse être un composite ou une feuille. Dans notre cas, une section à deux colonnes est un composite et les autres tags sont des feuilles. De plus, une colonne peut être divisée en deux colonnes, ce qui correspond à un composite de composite.

Ajouter de la couleur, du gras et (ou) de l'italique sur n'importe quel tag existant (StyleCSS).

Pattern proposé : Decorator

Pourquoi : Le decorator permet d'ajouter des fonctionnalités à un objet sans modifier sa structure. Le style CSS est une fonctionnalité que l'on peut ajouter à n'importe quel tag existant. Il est possible de composer plusieurs décorateurs pour obtenir un style CSS plus complexe.

Créer très facilement des pages à partir de templates prédéfinis pour facilement alimenter le site. Trois templates de base existent : Page "BigTitlePage" constituée d'un titre principal et d'un paragraphe, la page "PicturePage" qui comprend une image suivi d'un titre en taille 2 et la page "SourcesPage" qui contient cinq liens suivis d'une brève description

Pattern proposé : Prototype

Pourquoi : Le prototype permet de créer des objets à partir d'un modèle. Dans notre cas, les templates sont les modèles et les pages sont les objets créés à partir de ces modèles.

Ajouter aisément de nouveaux templates à ceux existant à l'aide d'un outil permettant de construire rapidement une page personalisée. L'outil propose à l'utilisateur de pouvoir ajouter autant d'éléments qu'il le souhaite à la page avant de la créer et de l'ajouter aux templates (TemplateGenerator)

Pattern proposé : Builder

Pourquoi : Le builder permet de construire un objet étape par étape. Dans notre cas, l'outil

permet de construire une page étape par étape. L'utilisateur peut ajouter autant d'éléments qu'il le souhaite à la page avant de la créer et de l'ajouter aux templates.

Créer des pages au style cohérent sur l'ensemble de la page. L'outil doit donc permettre de selectionner un style appliqué sur l'ensemble des éléments qui seront créés sur la page (CoherentStyler)

Pattern proposé : Abstract factory

Pourquoi : L'abstract factory permet de créer des familles d'objets liés entre eux sans préciser leurs classes concrètes. Dans notre cas, le style cohérent est une famille d'objets liés entre eux (les éléments de la page) et l'outil permet de créer des objets de cette famille sans préciser leurs classes concrètes.

L'ajout ou la suppression de page dans le site web est loggué ; soit dans la console, soit dans un fichier en fonction des paramettres de l'utilisateur (logger)

Pattern proposé : Proxy

Pourquoi : Le proxy permet de contrôler l'accès à un objet. Dans notre cas, le proxy permet de contrôler l'accès à l'objet site web. L'ajout ou la suppression de page dans le site web est loggué.

Pattern alternatif : Observer

Pourquoi : L'observer permet de notifier un objet quand un autre objet change d'état. Dans notre cas, le serveur notifie l'objet logger quand il change d'état (ajout ou suppression de page).

L'ensemble des actions d'édition d'une page doit pouvoir être annulé (Canceler)

Pattern proposé : Memento

Pourquoi : Le memento permet de sauvegarder l'état d'un objet afin de pouvoir le restaurer ultérieurement. Dans notre cas, l'ensemble des actions d'édition d'une page doit pouvoir être annulé.

Le site web doit pouvoir être exporté sous format LaTex ou JSON (Rewriter)

Pattern proposé : Visitor

Pourquoi : Le visitor permet d'ajouter des fonctionnalités à un objet sans modifier sa structure.

Dans notre cas, le site web doit pouvoir être exporté sous format LaTex ou JSON.

Le cout d'hébergement doit pouvoir être calculé en fonction du prix appliqué par divers Hébergeurs. Les hébergeurs appliquent en effet des tarifs différents pour chaque tag (hostCost)

Pattern proposé : Strategy

Pourquoi : Le strategy permet de définir une famille d'algorithmes, encapsuler chacun d'entre eux et les rendre interchangeables. Dans notre cas, le cout d'hébergement doit pouvoir être calculé en fonction du prix appliqué par divers Hébergeurs.

Pattern alternatif : Chain of responsibility

Pourquoi : La chaine de responsabilité permet d'envoyer une requête à travers une chaine d'objets. Dans notre cas, le cout d'hébergement doit pouvoir être calculé en fonction du prix appliqué par divers Hébergeurs. Chaque hébergeur peut calculer le cout d'hébergement ou le transmettre à l'hébergeur suivant.