**Dossier de Conception des Classes de l'Application Navigateur Web (ICTU SEARCH)**

**1. Introduction**

Ce document décrit les classes principales de l'application Navigateur Web (ICTU SEARCH), détaillant leurs attributs et leurs méthodes. Il inclut également un diagramme de classes pour visualiser les relations et les interactions entre ces composants.

**2. Structures des Classes**

L'application Navigateur Web (ICTU SEARCH) est principalement structurée autour de la classe MainWindow qui intègre les fonctionnalités de navigation web et de gestion de l'historique. Le cœur du rendu web est géré par une classe externe de Qt, QWebEngineView.

**2.1 Classe MainWindow**

La classe MainWindow est la fenêtre principale de l'application et gère toute l'interface utilisateur graphique ainsi que la logique de navigation web et de persistance de l'historique simple.

**Attributs :**

| **Attribut** | **Type** | **Description** |
|---|---|---|
| ui | Ui::MainWindow * | Pointeur vers l'objet UI généré par Qt Designer, représentant l'interface graphique. |
| webView | QWebEngineView * | Pointeur vers le widget de la vue web qui affiche le contenu des pages. |
| backStack | QStack<QString> | Pile stockant les URL des pages précédemment visitées pour la navigation "Retour". |
| forwardStack | QStack<QString> | Pile stockant les URL des pages pour la navigation "Avance" (après un "Retour"). |
| currentString | QString | L'URL actuellement affichée ou en cours de chargement dans la vue web. |

**Méthodes Publiques :**

| **Méthode** | **Signature** | **Description** |
|---|---|---|

| | | |
|---|---|---|
| MainWindow | explicit MainWindow(QWidget *parent = nullptr) | Constructeur. Initialise l'interface utilisateur, la vue web et configure les connexions signaux/slots. |
| ~MainWindow | ~MainWindow() | Destructeur. Nettoie les ressources allouées (notamment l'objet ui). |

**Slots Privés (réactions aux interactions utilisateur) :**

| **Slot** | **Signature** | **Description** |
|---|---|---|
| on_prevButton_clicked | void on_prevButton_clicked() | Déclencheur du bouton "Retour". Gère la navigation vers l l'URL précédente dans l'historique. |
| on_nextButton_clicked | void on_nextButton_clicked() | Déclencheur du bouton "Avance". Gère la navigation vers l'URL suivante dans l'historique. |
| on_setButton_clicked | void on_setButton_clicked() | Déclencheur du bouton "Aller". Charge l'URL saisie dans le champ de texte. |

**Méthodes Privées (fonctions d'aide) :**

| **Méthode** | **Signature** | **Description** |
|---|---|---|
| onLoadUrl | void onLoadUrl(const QString &url) | Charge une URL donnée dans webView et met à jour les piles d'historique. |
| saveStringToFile | void saveStringToFile(const QString &str) | Sauvegarde une chaîne (URL) dans le fichier entered_strings.txt. |
| updateButtonStates | void updateButtonStates() | Met à jour l'état (activé/désactivé) des boutons "Retour" et "Avance" en fonction de l'historique. |

### 2.2 Classe QWebEngineView (Classe de Qt)

Bien que ce ne soit pas une classe à implémenter par le développeur de l'application, QWebEngineView est un composant crucial fourni par le framework Qt. Elle est instanciée et utilisée par MainWindow.

**Attributs (pertinents pour l'interaction) :**

- Gère le rendu du contenu web.
- Gère l'historique de navigation interne au moteur de rendu (distinct des piles backStack et forwardStack de MainWindow).

**Méthodes (pertinentes pour l'interaction) :**

| **Méthode** | **Signature** | **Description** |
|---|---|---|
| setUrl | void setUrl(const QUrl &url) | Charge une URL spécifique dans la vue web. |
| url | QUrl url() const | Retourne l'URL actuellement chargée dans la vue web. |
| back | void back() | Navigue à la page précédente dans l'historique interne du moteur de rendu. |
| forward | void forward() | Navigue à la page suivante dans l'historique interne du moteur de rendu. |
| loadStarted | void loadStarted() | Signal émis lorsque le chargement d'une page commence. |
| loadFinished | void loadFinished(bool ok) | Signal émis lorsque le chargement d'une page est terminé. |
| titleChanged | void titleChanged(const QString &title) | Signal émis lorsque le titre de la page change. |
| urlChanged | void urlChanged(const QUrl &url) | Signal émis lorsque l'URL de la page change. |

### 2.3 Classes de Persistance (QFile, QTextStream, QCoreApplication)

Ces classes de Qt sont également des composants externes utilisés par MainWindow pour gérer la sauvegarde des URL.

- **QFile**: Gère les opérations de fichier (ouverture, écriture, fermeture).
- **QTextStream**: Permet d'écrire des données textuelles dans un QFile.
- **QCoreApplication**: Fournit des informations sur l'application, notamment le chemin de son répertoire (applicationDirPath()), ce qui est utile pour placer le fichier de sauvegarde.

**3. Représentation Synoptique des Interactions entre les Classes (Diagramme de Classes UML)**

Voici un diagramme de classes simplifié pour visualiser les relations entre la classe principale MainWindow et les composants de Qt qu'elle utilise.

```
classDiagram
    class MainWindow {
        - Ui::MainWindow *ui
        - QWebEngineView *webView
        - QStack~QString~ backStack
        - QStack~QString~ forwardStack
        - QString currentString
        + MainWindow(parent)
        + ~MainWindow()
        + on_prevButton_clicked()
        + on_nextButton_clicked()
        + on_setButton_clicked()
        - onLoadUrl(const QString &url)
        - saveStringToFile(const QString &str)
        - updateButtonStates()
    }

    class QWebEngineView {
        + setUrl(const QUrl &url)
```

```
    + url() QUrl
    + back()
    + forward()
    + loadStarted()
    + loadFinished(bool ok)
    + urlChanged(const QUrl &url)
    // ... autres méthodes et signaux
}

class QFile {
    + open(mode) bool
    + close() void
    + errorString() QString
    // ...
}

class QTextStream {
    + QTextStream(device)
    + operator<<(const QString &str) QTextStream&
    // ...
}

class QCoreApplication {
    + static applicationDirPath() QString
    // ...
}

MainWindow "1" *-- "1" QWebEngineView : contient (composition)
```

MainWindow ..> QFile : utilise

MainWindow ..> QTextStream : utilise

MainWindow ..> QCoreApplication : utilise (statique)

MainWindow ..> QUrl : utilise

MainWindow ..> QStack : utilise

QWebEngineView ..> QUrl : utilise

**Explication des Interactions :**

- **MainWindow et QWebEngineView (Composition)**:
    - MainWindow crée et possède une instance de QWebEngineView (webView). C'est une relation de composition car MainWindow est responsable de la durée de vie de webView.
    - MainWindow appelle les méthodes de webView (comme setUrl, back, forward) pour contrôler la navigation.
    - MainWindow se connecte aux signaux de webView (comme urlChanged) pour mettre à jour son propre état et l'historique.
- **MainWindow utilise QFile, QTextStream, QCoreApplication (Dépendance)**:
    - MainWindow dépend de ces classes pour la fonctionnalité de sauvegarde des URL (saveStringToFile). Elle les utilise pour ouvrir, écrire dans des fichiers et déterminer le chemin de l'application.
- **MainWindow utilise QStack et QUrl (Dépendance)**:
    - MainWindow utilise QStack pour gérer ses piles d'historique (backStack, forwardStack).
    - MainWindow manipule des objets QUrl pour représenter et valider les adresses web.

Ce diagramme illustre que MainWindow est la classe centrale qui orchestre toutes les opérations de l'application en utilisant les composants spécifiques du framework Qt pour les tâches de rendu web, de gestion de l'interface et de persistance.