SCI6306 Bases de données documentaires (Automne 2023)

Christine Dufour, EBSI, UdeM

A2023 9 novembre 2023

Cours 9 : Formulaire pour saisie (1 de 2)

SCI6306

Christine Dufour, EBSI, UdeM

Table des matières

Formulaires Web pour saisir des données dans une BD relationnelle avec PHP (partie 1 de 2)

- + Au programme aujourd'hui
- Bases de données sur le Web : Scénario général
- Bases de données sur le Web : Saisie de données (partie 1 de 2)
    - Étapes pour concevoir une interface permettant d'enregistrer des données dans une BD sur le Web
    - Création d'un formulaire HTML pour faire de la saisie de données dans une BD sur le Web
- Ressources en lien avec le cours

1. + Au programme aujourd'hui

- **Intégration Web d'une base de données MySQL : scénario général**
- **Intégration Web d'une base de données MySQL : enregistrement de données**
    - Partie 1 : Préparation du formulaire Web
        - Travail en laboratoire : Exercice

2. Bases de données sur le Web : Scénario général

L'intégration d'une BD sur le Web permet, par exemple, de donner **accès à distance** à une base de données pour y **saisir** des informations ou y faire des **recherches**, et ce, sans souci quant au système d'exploitation de l'utilisateur ou de l'utilisatrice. Cela permet aussi de **générer des pages Web dynamiques**, c'est-à-dire des pages Web dont le contenu n'est pas a priori fixé et qui sont alimentées à partir d'une base de données (Figure 1).

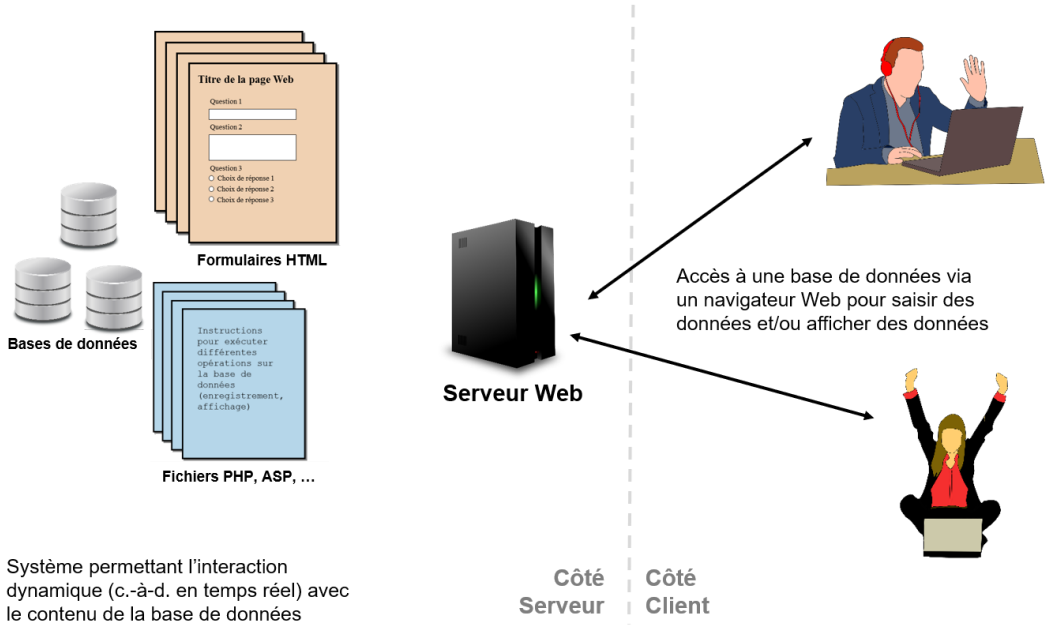


Figure 1. Scénario général d'intégration d'une base de données sur le Web

C'est le cas par exemple de l'environnement Web utilisé à l'EBSI pour la gestion des plans de cours. Les enseignant.e.s ont accès à distance à la base de données des plans de cours pour y mettre à jour leurs plans de cours (Figure 2). La base de données permet aussi de rendre disponibles sur le site de l'EBSI les plans de cours qui sont générés à la demande lorsque l'on clique sur un des sigles de cours (https://cours.ebsi.umontreal.ca/) (Figure 3).

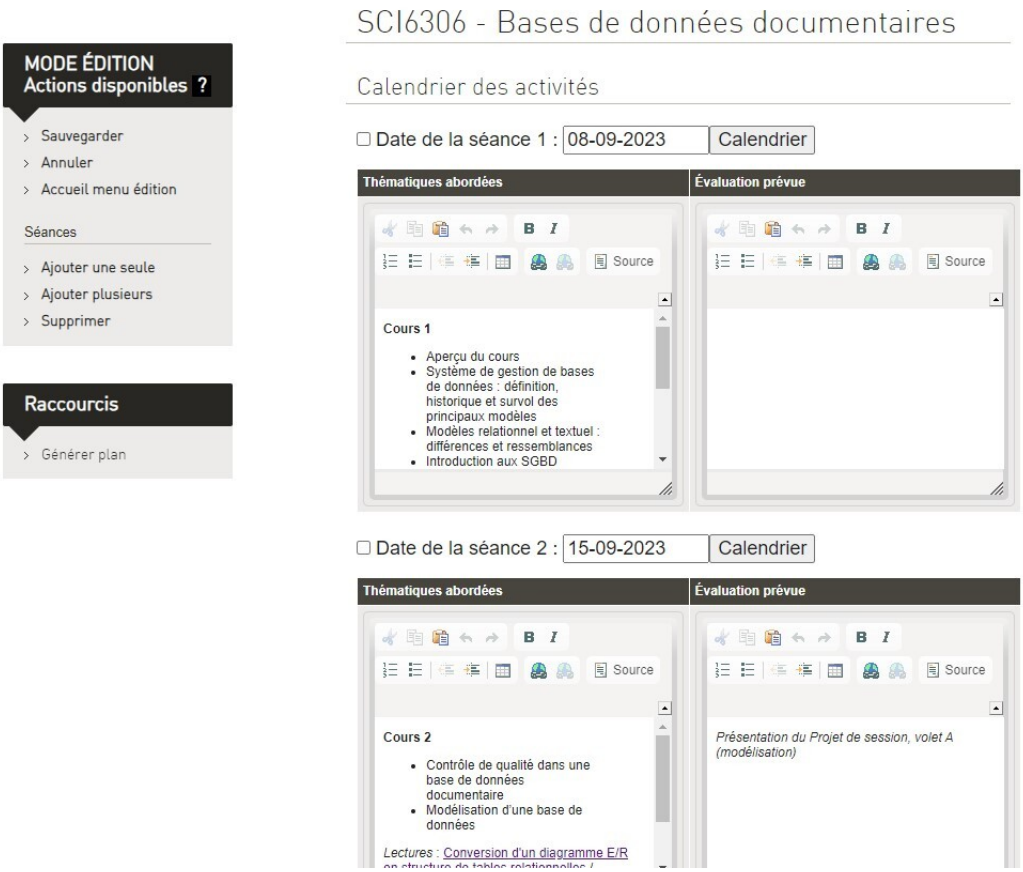


Figure 2. Extrait de l'nterface de saisie du calendrier des activités pour les enseignant.e.s

Calendrier des activités

Date	Activité(s)	Évaluation
2023-09-08	**Cours 1** • Aperçu du cours • Système de gestion de bases de données : définition, historique et survol des principaux modèles • Modèles relationnel et textuel : différences et ressemblances • Introduction aux SGBD relationnelles *Lectures* : Introduction au modèle relationnel par comparaison avec le modèle textuel / Marcoux; Définition et caractéristiques des bases de données / Dufour; Bases de données sur le Web / Dufour *Complément* : Bases de données non relationnelles / Habert	
2023-09-15	**Cours 2** • Contrôle de qualité dans une base de données documentaire • Modélisation d'une base de données *Lectures* : Conversion d'un diagramme E/R en structure de tables relationnelles / Marcoux; Comment distinguer une entité d'un attribut / Marcoux; Clé primaire multi-champ / Marcoux; Bases de données relationnelles / Dufour	*Présentation du Projet de session, volet A (modélisation)*

Figure 3. Extrait de l'affichage du calendrier des activités sur le site de l'EBSI pour les étudiant.e.s dans le plan de cours (https://cours.ebsi.umontreal.ca/planscours/sci6306)

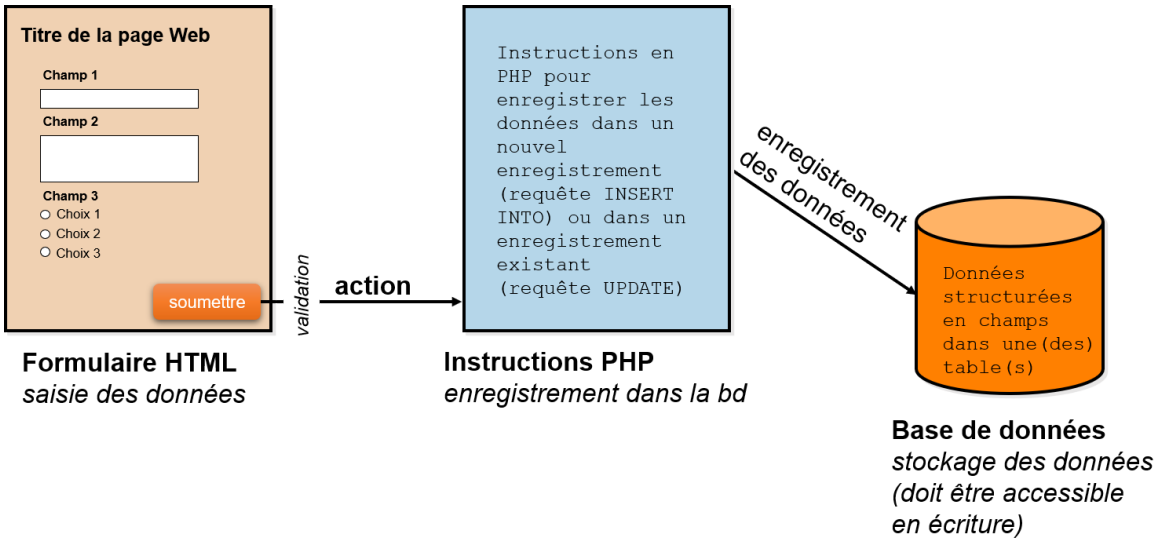
Le scénario général présenté correspond à une **architecture trois tiers**. Il s'agit d'une architecture *client-serveur* où l'on retrouve **trois couches** (d'où son appellation d'architecture trois tiers) :

1. La première couche est la **couche de présentation** qui réside du côté *client*. C'est le navigateur Web qui permet l'interprétation et l'affichage des pages du système Web.
2. La deuxième couche est la **couche fonctionnelle** qui se trouve du côté *serveur*. On y retrouve ce qu'il faut pour exécuter les différentes actions programmées par exemple en PHP ou en ASP, actions qui représentent les fonctionnalités du système Web.
3. La troisième couche est la **couche des données**, aussi du côté *serveur*, et qui correspond à la base de données qui héberge le contenu du système Web.

La personne qui conçoit d'un système Web répondant à cette architecture aura ainsi à garder en tête ces trois couches, comme elle devra toutes les concevoir. Selon les fonctionnalités désirées, elle pourrait avoir à concevoir par exemple des interfaces pour permettre aux utilisateurs et aux utilisatrices de saisir des données dans la base de données ainsi que des interfaces permettant d'afficher le contenu de la base de données.

3. Bases de données sur le Web : Saisie de données (partie 1 de 2)

La saisie de données dans une base de données mise en ligne se fait habituellement par le biais de **formulaires Web**. Ces derniers intègrent différents types de **contrôle** qui permettent à un utilisateur ou à une utilisatrice d'entrer des données pour ensuite les sauvegarder : *boutons radio*, *cases à cocher*, *boîtes de texte libre*, etc. Lorsque l'on conçoit les interfaces nécessaires à la saisie de données, il faut préparer d'une part le **formulaire HTML** et, d'autre part, les **instructions** dans un langage de programmation comme PHP ou ASP qui permettront de **se connecter** à la base de données pour y **verser les données** saisies.



Schématisation de l'enregistrement de données dans une BD sur le Web

Les instructions en PHP sont exécutées habituellement après avoir cliqué sur le bouton prévu dans l'interface pour la sauvegarde. On retrouvera entre autres dans ces instructions la requête SQL permettant d'enregistrer les données qui sera soit de type *Ajout de données (INSERT INTO)*, s'il s'agit d'un nouvel enregistrement, ou de type *Mise à jour de données (UPDATE)*, s'il s'agit d'un enregistrement existant dont on a modifié les données. Ces instructions peuvent être placées soit dans la page HTML du formulaire, soit dans un fichier distinct. C'est cette dernière approche que nous retiendrons pour commencer comme il est plus simple de conceptualiser le formulaire et les instructions PHP de manière séparée lorsqu'on débute.

3.1. Étapes pour concevoir une interface permettant d'enregistrer des données dans une BD sur le Web

On retrouve **trois grandes étapes** lors de la conception d'une interface permettant de sauvegarder des données dans une BD sur le Web :

1. Préparation de la *base de données MySQL*
2. Préparation du *formulaire HTML*
3. Définition du *code PHP pour enregistrer* les réponses dans la base de données

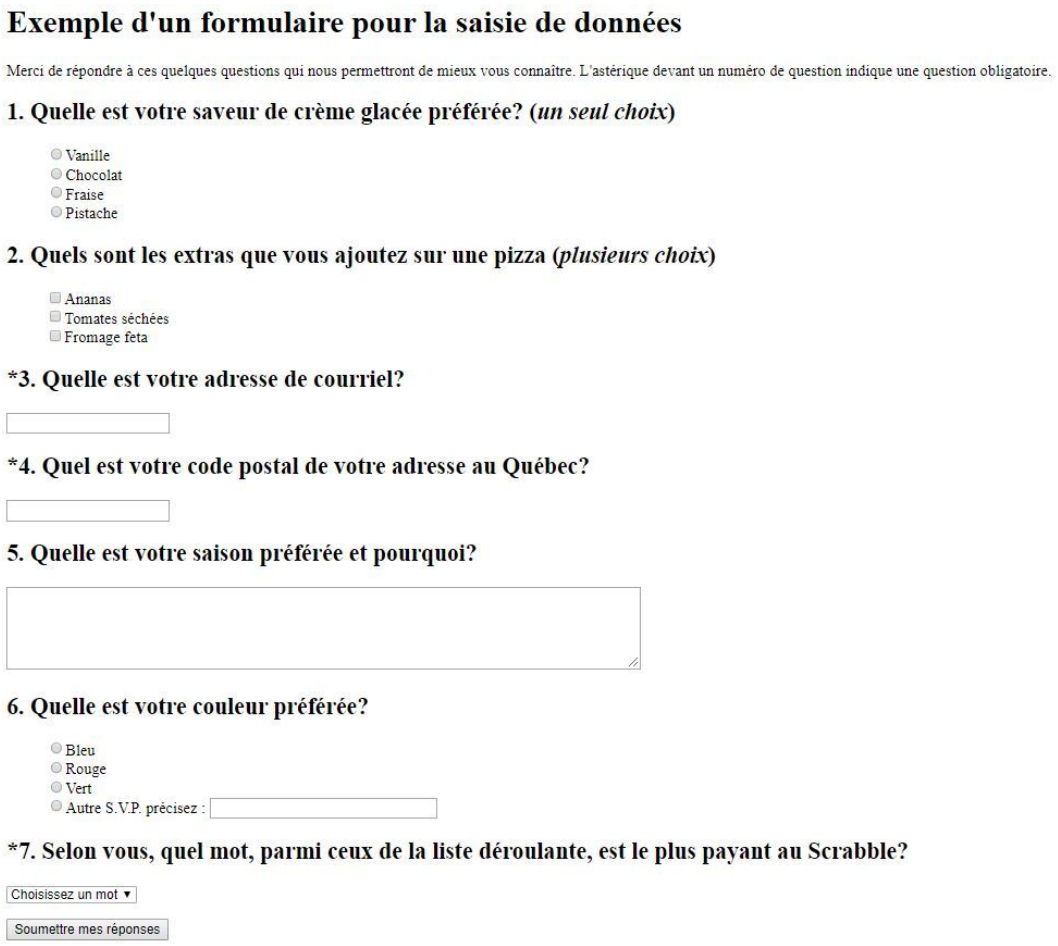
La première étape a déjà été abordée dans les séances de cours précédentes. Pour rappel, cette première étape se divise en deux moments :

1. La **modélisation** de la réalité à décrire pour définir le schéma relationnel correspondant (ce que vous avez fait dans le volet A du projet de session), soit la précision du diagramme Entités-Relations, la définition de la structure des tables, la définition des contraintes additionnelles, la présentation d'exemples de contenus valides.
2. L'**opérationnalisation de la modélisation dans le SGBD relationnel** choisi (ce qui est fait dans le volet B du projet de session), soit la création des tables, des index et des relations et potentiellement l'importation de données s'il s'agit, par exemple, d'une migration à partir d'un ancien système.

Nous nous attarderons dans la séance de cours 9 à la deuxième étape, soit la préparation du formulaire HTML. Finalement, la troisième étape sera abordée à la séance de cours suivante (cours 10).

3.2. Création d'un formulaire HTML pour faire de la saisie de données dans une BD sur le Web

Afin d'illustrer la définition d'un formulaire, nous utiliserons l'exemple fictif du formulaire suivant :



Exemple d'un formulaire pour la saisie de données dans une BD sur le Web

Lorsque l'on clique sur **Soumettre mes réponses**, le formulaire nous redirige vers une page de remerciements qui rappelle nos réponses (mais sans les enregistrer dans une base de données) ainsi qu'elle corrige la toute dernière question :

Merci de votre participation!

Voici ce que vous avez répondu :

- **Crème glacée préférée** : chocolat
- **Extras Pizza** : Tomates séchées
- **Courriel** : me@mysleft.andi
- **Code postal** : H0H 0H0
- **Saison préférée** : La saison 2 de Castle! Bien que la saison 3 de Suits est pas mal aussi...
- **Couleur préférée** : bleu
- **Mot choisi** : xylographiques Eh non, deshypothéquiez est plus payant avec 49 points...

Page affichée après la soumission du formulaire

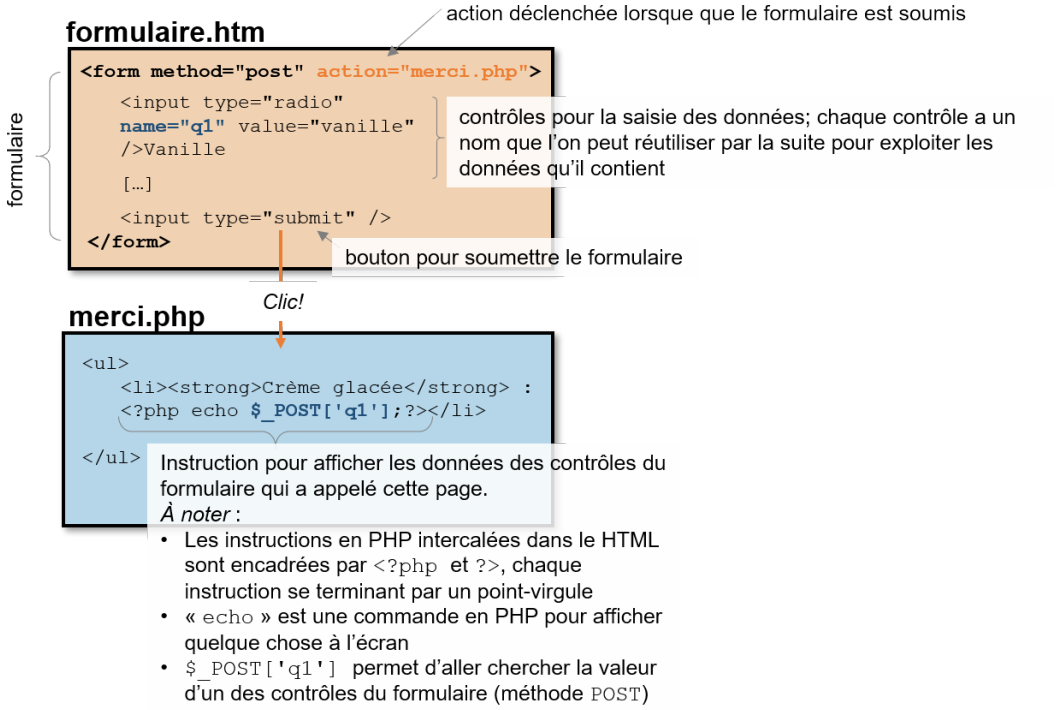
Remarque **:**

Ce formulaire est accessible en lien à l'URL https://cours.ebsi.umontreal.ca/sci6306/demo_c9/formulaire.htm.[1] Il est aussi possible de télécharger les fichiers qui le composent à l'URL https://cours.ebsi.umontreal.ca/sci6306/demo_c9/sci6306_demo_c9.zip.

Principe général

L'idée de base est qu'un **formulaire HTML** contient différents **contrôles** (boîtes de saisie, menus déroulants, etc.) permettant à l'utilisateur ou l'utilisatrice de saisir des données. Dans l'exemple, on retrouve entre autres des *boutons radio* pour le choix de la crème glacée et une *boîte de saisie en texte libre* pour la saison préférée.

Comme illustré ci-dessous, le **bouton pour soumettre** le formulaire déclenche l'**action** indiquée pour le formulaire (attribut *action* de l'élément *form*), comme, par exemple, l'ouverture d'une nouvelle page. Dans l'exemple, ce bouton, intitulé *Soumettre mes réponses*, mène à la page des remerciements (`merci.php`).



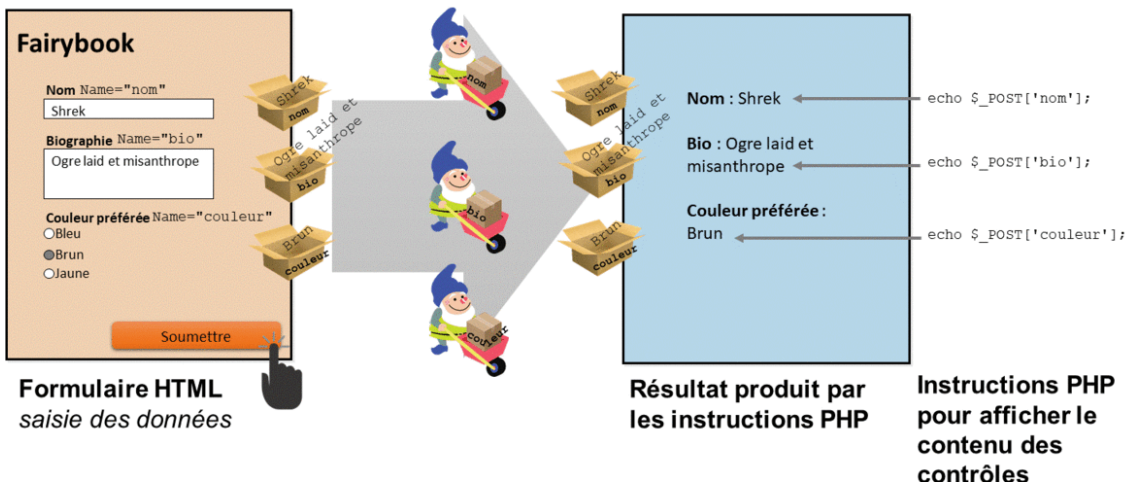
Schématisation de l'articulation du formulaire HTML et du fichier d'instructions PHP

[1] https://cours.ebsi.umontreal.ca/sci6306/demo_c9/formulaire.htm

Il est possible, dans cette *nouvelle page merci.php*, d'avoir accès aux *valeurs contenues dans les contrôles* pour, par exemple, les afficher ou les enregistrer dans une base de données. On retrouve ainsi, dans la page des remerciements de l'exemple :

- des **balises HTML** pour les **chaînes de texte fixes** - par exemple, le titre de la page "Merci de votre participation!",
- ainsi que du **code PHP** pour **insérer dynamiquement le contenu des contrôles** de formulaire de la page précédente.

Le principe du transfert des données du formulaire à la page suivante est simple, comme illustré et expliqué ci-dessous.



Principe du transfert des données des contrôles d'un formulaire

Chaque **contrôle** d'un formulaire porte un **nom** qui lui est propre (valeur de l'attribut *name* du contrôle). On peut visualiser ces contrôles en fait comme des boîtes dans lesquelles on dépose les valeurs saisies. Lorsque l'on clique sur un **bouton de soumission**, ces boîtes sont *transportées* jusqu'à la page pointée par le bouton de soumission (le nom de cette page se retrouve comme valeur de l'attribut *action* de la balise *form* décrite un peu plus loin). Une fois rendue à cette nouvelle page, il est possible de **récupérer** les valeurs de ces boîtes pour les afficher. En PHP, les **boîtes** des contrôles sont représentées par *$_POST['nom de la boîte']*, pour un formulaire privilégiant la méthode *POST* (nous reviendrons sur cette méthode plus tard). C'est la commande PHP *echo* qui permet de faire afficher ce contenu.

Structure générale d'un formulaire HTML

La structure générale d'un formulaire HTML est la suivante :

```
1 <form method="post" action="action">
2     <!-- Éléments du formulaire (questions) -->
3     <p><textarea rows="10" cols="60" name="comment">S.V.P. faites nous part de tout
  commentaire en lien avec un aspect ou l'autre de notre bibliothèque.</textarea></p>
4     [...]
5     <!-- Bouton pour la soumission du formulaire; exécute l'action définie pour le
  formulaire -->
6     <p><input name="soumission_form" type="submit" value="Titre du bouton" /></p>
7 </form>
```

Ainsi :

- L'**ensemble des contrôles** d'un formulaire sera inclus à l'intérieur de la balise *form*.
- On retrouve dans la balise *form* minimalement **deux attributs** :
    1. l'attribut *method* pour indiquer quelle méthode est utilisée pour **transférer les données** (POST dans l'exemple, l'autre valeur possible étant GET),
    2. l'attribut *action* qui permet d'indiquer **ce qu'il faut faire après avoir cliqué** le bouton de soumission du formulaire; on peut par exemple y indiquer le nom d'un fichier à ouvrir. Dans l'exemple, la valeur de l'attribut *action* de la balise *form* a pour valeur *merci.php* : `<form method="post" action="merci.php">`
- Parmi les contrôles du formulaire, on retrouvera un *bouton* qui permettra de **soumettre le formulaire**. Il s'agit d'un élément *input* de type *submit*. Ce que l'on indique comme **valeur de son attribut** *value* apparaîtra sur le bouton. Dans l'exemple, l'attribut *value* du bouton de soumission a pour valeur *Soumettre mes réponses* : `<input name="bouton_soumission" type="submit" value="Soumettre mes réponses" />`

Lorsque l'on conçoit un formulaire on peut utiliser tous les types de questions habituelles dans un formulaire :

Types d'éléments (contrôles) pour un formulaire

Type de questions	Définition	Balises
Fermée - À choix multiples	Plusieurs choix offerts, un seul choix possible	• `input type="radio"` (tous les boutons sont visibles sur la page) • `select` (les choix sont présentés dans un menu déroulant)
Fermée - À réponses multiples	Plusieurs choix offerts, possibilité d'en sélectionner plus d'un	• `input type="checkbox"` (toutes les cases à cocher sont visibles sur la page) • `select` (les choix sont présentés dans un menu déroulant)
Ouverte - Réponse courte	Information courte de différents types (texte, date, nombre)	• Textuel : `input type="text"` • Courriel : `input type="email"` • Date : `input type="date"` • Nombre : `input type="number"` • *Note* : une validation quant au contenu saisie sera faite automatiquement, par exemple, pour le type `number`, la saisie de texte sera refusée
Ouverte - Réponse longue	Information sur une ou plusieurs lignes (alphanumérique)	• `textarea`

L'exemple ci-dessous illustre les quatre types de questions décrites :

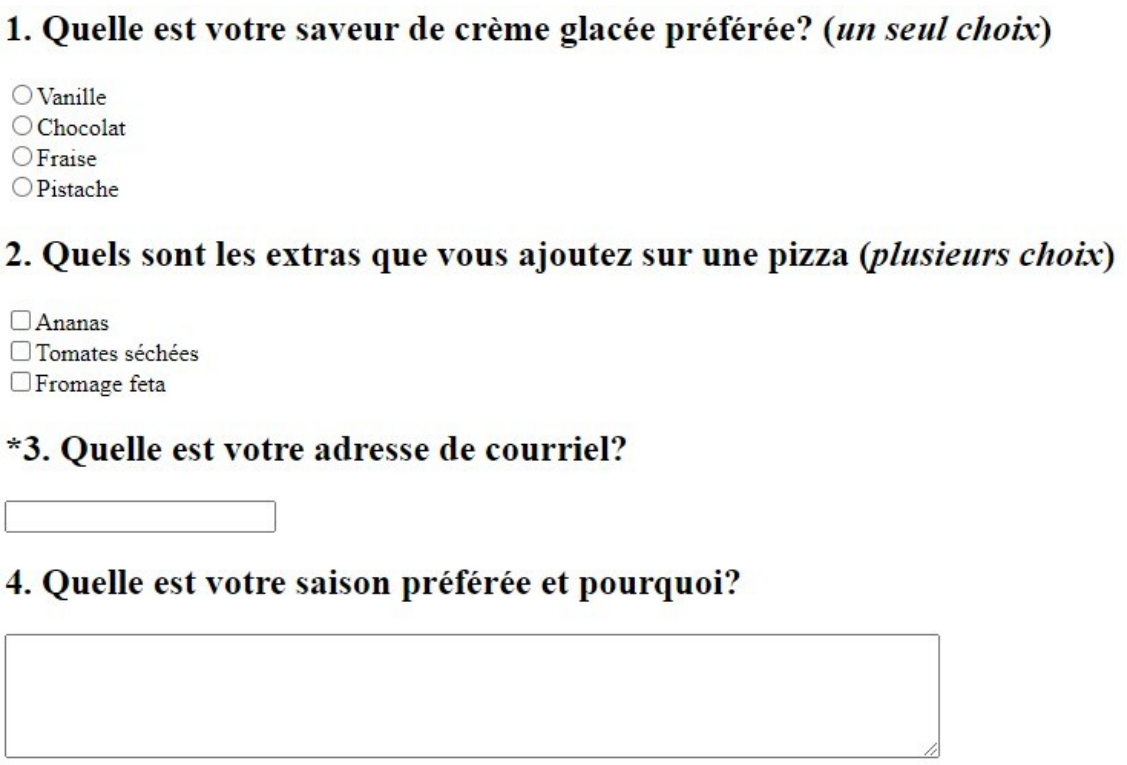
```
<!-- Question fermée - À choix multiples -->
<h2>1. Quelle est votre saveur de crème glacée préférée? (<em>un seul choix</em>)</h2>

<input type="radio" id="q1_choix1" name="q1" value="vanille" /><label for="q1_choix1">Vanille</label><br/>
<input type="radio" id="q1_choix2" name="q1" value="chocolat" /><label for="q1_choix2">Chocolat</label><br/>
<input type="radio" id="q1_choix3" name="q1" value="fraise" /><label for="q1_choix3">Fraise</label><br/>
<input type="radio" id="q1_choix4" name="q1" value="pistache" /><label for="q1_choix4">Pistache</label>

<!-- Question fermée - À réponses multiples -->
<h2>2. Quels sont les extras que vous ajoutez sur une pizza (<em>plusieurs choix</em>)</h2>

<input type="checkbox" id="q2_1" name="q2_1" value="X" /><label for="q2_1">Ananas</label><br/>
<input type="checkbox" id="q2_2" name="q2_2" value="X" /><label for="q2_2">Tomates séchées</label><br/>
<input type="checkbox" id="q2_3" name="q2_3" value="X" /><label for="q2_3">Fromage feta</label>

<!-- Question ouverte - Réponse courte-->
<h2 title="Doit contenir un @">*3. Quelle est votre adresse de courriel?</h2>

<input type="email" maxlength="100" size="20" name="q3" required="required" />

<!-- Question ouverte - Réponse longue -->
<h2>4. Quelle est votre saison préférée et pourquoi?</h2>

<textarea rows="5" cols="80" name="q5"></textarea>
```

Voici quelques éléments à remarquer dans le code HTML ci-dessus :

- **Question 1 (un seul choix)**
  - La balise `input` qui définit un bouton radio est suivie par une chaîne de caractère. Sans cette chaîne de caractère, on ne verrait sur la page Web qu'un bouton radio sans étiquette.
  - Tous les boutons radio portent le même nom (`name="q1"`) afin d'en faire un groupe de bouton qui ne permet d'en choisir qu'un seul.
  - La balise `label` permet de rendre cliquable la chaîne de caractère affichée; sans cette dernière, seul le bouton radio serait cliquable.
  - Ce qui sera mis en mémoire est ce qui se retrouve dans l'attribut `value`.
- **Question 2 (plusieurs choix)**
  - La balise `input` qui définit une case à cocher est suivie par une chaîne de caractère. Sans cette chaîne de caractère, on ne verrait sur la page Web qu'un bouton radio sans étiquette.
  - Les cases à cocher portent des noms différents pour faciliter par la suite l'enregistrement. Si le même nom est utilisé, toutes les valeurs cochées se retrouveraient dans un vecteur qui est un peu plus complexe à traiter lorsque l'on débute.
  - La balise `label` permet de rendre cliquable la chaîne de caractère affichée; sans cette dernière, seul la case à cocher serait cliquable.
  - Ce qui sera mis en mémoire sera un "X" pour indiquer que le choix a été coché.

- **Question 3 (réponse courte)**
  - Le type `email` de la balise input permettra de s'assurer que c'est bien une adresse courriel qui est saisie.
  - L'attribut `maxlength` permet de s'assurer que l'on ne pourra saisir plus de 100 caractères afin de respecter la longueur du champ dans la base de données.
  - L'attribut `size` contrôle la longueur de la boîte affichée. Elle peut être plus courte que le nombre de caractères permis.
  - L'attribut `required` permet d'obliger la saisie lorsqu'un champ est obligatoire. Dans les deux questions précédentes, cet attribut était absent comme les questions étaient facultatives.
  - Dans la balise `h2`, l'attribut `title` permet de faire afficher une infobulle lorsque l'on survole l'énoncé de la question afin de rappeler le format attendu.
- **Question 4 (réponse longue)**
  - La balise `textarea` possède une balise d'ouverture et une balise de fermeture (contrairement à la balise `input` qui n'a qu'une balise concaténant l'ouverture et la fermeture). Ce qui est écrit entre les deux balises sera affiché dans la boîte de saisie. On pourrait s'en servir pour mettre une valeur par défaut par exemple.
  - Les attributs `rows` et `cols` contrôlent la taille de la boîte de saisie.

La copie d'écran ci-dessous illustre l'affichage de ce code dans un navigateur :



Visualisation du code dans un navigateur

✎ *Remarque* :

Un glossaire interactif des différentes balises utilisées pour le cours est accessible à l'URL https://cours.ebsi.umontreal.ca/glossaireweb/index.php?cours=sci6306.

Le choix du type de contrôle à utiliser est bien plus qu'une question de visuel. Il faut choisir les contrôles qui permettent de bien respecter les types de données des champs de la base de données.

Types de contrôles de formulaire versus Types de données d'une BD

Types de contrôle	Types de données
Contrôle de type `radio` & `checkbox`	En fonction des valeurs associées aux choix proposés • Par ex., si choix = 11234 : Champ de type **numérique** • Par ex., si choix = français : Champ de type **caractères** • Par ex., si choix = 12 janvier 2014 : Champ de type **date/heure**
Contrôle de type `text`	Champ de type **caractère** • Par ex., si la réponse saisie est Bibliothécaire
Contrôle de type `number`	Champ de type **numérique** • Par ex., si la réponse saisie est 20
Contrôle de type `date`	Champ de type **date/heure** • Par ex., si la réponse saisie est 1980-01-01
Contrôle de type `email`	Champ de type **caractère** • Par ex., si la réponse saisie est moi@me.com
Contrôle de type `textarea`	Champ de type **caractère**

Validation de la saisie

Plusieurs moyens peuvent être utilisés pour s'assurer de **valider** l'information à la saisie afin d'en augmenter la qualité. Ce peut être par **programmation**, par exemple avec *JavaScript*, mais aussi plus simplement par une **bonne utilisation des différents types de contrôle** et de leurs **attributs**.

On retrouve dans **HTML5** de nouveaux types de contrôle qui permettent de valider le contenu. C'est le cas, par exemple, des balises `input type="number"`, pour s'assurer de la saisie d'un chiffre, et `input type="email"` pour les adresses courriel. On retrouve une liste de ces nouveaux contrôles sur le site du W3Schools à l'URL https://www.w3schools.com/html/html_form_input_types.asp.

Il est aussi possible d'exploiter différents **attributs** des balises des contrôles de formulaire pour effectuer de la validation :

Attributs des contrôles de formulaire permettant la validation

Attribut	Explication	Exemple
required	• Pour les champs *obligatoires* • Peut s'ajouter à tous les types de contrôle	• `<input type="text" required="required" />` • *Exception* : si on veut s'assurer d'avoir au moins un choix parmi une liste de cases à cocher, il faut passer par JavaScript, l'attribut « *required* », pour les contrôles de type « *checkbox* » ne fonctionne que pour une seule case à cocher et non un groupe lorsqu'on leur donne un nom différent.

Attribut	Explication	Exemple
maxlength	• Pour limiter le *nombre de caractères à saisir* (donc ne pas dépasser la taille d'un champ de type caractère)	• Exemple pour un champ VARCHAR de taille 32 : `<input type="text" **maxlength="32"** />`
min / **max**	• Pour un champ numérique, pour *restreindre les valeurs permises*	• Exemple pour un champ TINYINT de taille 2 et unsigned : `<input type="number" **min="0" max="99"** />`
pattern	• Pour *valider un masque* • Utilisation des expressions régulières	• Exemple pour un code postal québécois (forme A0A 0A0) : ◦ **`pattern="[A-Z][0-9][A-Z] [0-9][A-Z][0-9]"`** ▪ [A-Z] veut dire une lettre majuscule entre A et Z ▪ [0-9] veut dire un chiffre entre 0 et 9 • Exemple pour un numéro de téléphone du type (###) ###-#### avec un poste facultatif ◦ **`pattern="\([0-9]{3}\) [0-9]{3}-[0-9]{4}.*"`** ▪ On ajoute la barre oblique inversée devant des caractères réservés comme la parenthèse lorsque l'on veut l'utiliser comme un caractère ordinaire ▪ {3} veut dire 3 fois l'expression qui précède (donc trois chiffres) ▪ Le point représente n'importe quel caractère tandis que l'astérisque veut dire entre 0 et plusieurs fois le caractère qui précède

Il n'est pas toujours possible de valider les données lors de la saisie, soit parce qu'il n'y a pas de « format typé » ou parce que de multiples formes sont acceptées (par exemple, pour des numéros de téléphone internationaux). En ce cas, il peut être utile de **rappeler** certaines **règles d'écriture**, par exemple :

- À même l'**interface**, en les écrivant directement
- Via des **infobulles**, ce qui permet d'éviter de surcharger l'interface
    - L'attribut `title` peut être ajouté à toutes les balises. Son contenu apparaîtra comme une infobulle lorsque l'on survolera la balise avec la souris.

4. Ressources en lien avec le cours

Matériel de cours

- *Notes de cours [cf. sci6306_cours9_notes.pdf]*

Ressources complémentaires

- *Rappel des bases HTML et CSS*
    - Notes de cours sur HTML du cours préalable (SCI6005)[1]
    - Notes de cours sur CSS du cours préalable (SCI6005)[2]
    - Glossaire interactif des éléments HTML vus dans le cours préalable (SCI6005)[3]
    - Glossaire interactif des propriétés CSS vues dans le cours préalable (SCI6005)[4]
- Glossaire interactif des éléments HTML vus dans le cours[5]

[1] https://cours.ebsi.umontreal.ca/sci6005/a2023/co/division_html.html
[2] https://cours.ebsi.umontreal.ca/sci6005/a2023/co/division_css.html
[3] https://cours.ebsi.umontreal.ca/glossaireweb/index.php?cours=sci6005
[4] https://cours.ebsi.umontreal.ca/glossaireweb/index_css.php?cours=sci6005
[5] https://cours.ebsi.umontreal.ca/glossaireweb/index.php?cours=sci6306