# Remise à niveau

## Utilisation d’un système de versionning : Git

Pour faciliter les échanges de fichiers du formateur, on va utiliser le logiciel git.

Cela permet de « versionner » les fichiers sur un serveur, afin que tous les développeurs travaillent sur les mêmes sources en même temps, avec chacun sa propre version sur son poste.

Le code de ce cours est disponible à l’url : <https://github.com/ice-devel/formation2020-cda>

## HTML

### Présentation HTML/CSS

Le HTML/CSS sont deux langages de description. Ils servent à indiquer aux navigateurs quels éléments afficher dans une page ainsi que leur disposition.

C’est la base du web, les spécifications HTML/CSS sont basés sur un standard : tous les navigateurs les comprennent.

### Utiliser la console du navigateur

Dans chaque navigateur, il y a un outil de debug : on peut y accéder en cliquant droit sur la page, et en sélectionnant « Inspecter l’élément », ou encore en appuyant sur la touche F12.

### Balises/tags de base

* Html/head/body (title/charset)
* Paragraphes
* Images
* Blocs
* Titres
* Liens (url absolue / relative – attribut target)
* Listes à puces / listes numérotées

### Les attributs des balises

* Attributs obligatoires
* Attributs facultatifs

### Position block/inline

Les éléments de type bloc prennent par défaut toute la largeur, et sont disposés les uns en dessous des autres.

Les éléments inline prennent la largeur de leur contenu et sont affichés les uns à côté des autres.

### Exercice de mise en page

Créer un fichier HTML et intégrer le design fourni (exercice.jpg). Respectez bien les balises html, head, body et le doctype.

### Tableau HTML

Les tableaux HTML permettent d’afficher des données sous forme de lignes et de cellules, un peu comme une feuille Excel.

### Formulaire HTML

* Les différents types de champs
* Les attributs

(min, max, minlength, maxlength, accept, placeholder)

* Les validations côté client

Never trusts user input

Validation html5 : se déclenche au moment de la soumission du formulaire

* Labels / Placeholders
* Envoi des données au serveur / attribut name / attribut action

Clé/valeur

* Method GET/POST

Avantages/inconvénients

### Autres balises (html5, em/strong)

Header, Footer, Section, Article, Nav, Aside : sémantique

<audio> <video>

<iframe>

## CSS

Le CSS permet de mettre en forme, de positionner des éléments HTML.

* Les propriétés courantes
* Style inline
* Sélecteur par tag / id (unique) / class
* Priorité entre sélecteurs / règle « !important »

Régle !important > Style Inline > Sélecteur ID > Sélecteur Classe > Sélecteur Tag

Les priorités ne font qu’écraser les propriétés déjà définies.

* Positionnement
  + Static
  + Relative
  + Absolute
  + Fixed
* Positionnement avancé
  + Flexbox : permet de créer des « grilles » afin de positionner des blocs proprement et facilement. Les uns à côté des autres / en dessous des autres / dans l’ordre / dans l’ordre inverse, de répartir la largeur de façon équitable entre les différents éléments, etc.
* Responsive design / Medias queries
* Frameworks HTML/CSS/JS
  + Bootstrap
  + MaterializeCSS
  + Foundation

## SQL / MySQL

* Persistance des données
* CRUD : Create Read Update Delete
* SQL : Structured Query Language
* SGBDR : Système de gestion de bases de données relationnelles
  + MySQL / MariaDB
  + SQL Server
  + Oracle
  + PostgreSQL
* PHPMyAdmin

Site web développé en PHP, offrant une interface graphique pour interagir avec une base de données MySQL/MariaDB.

* Structure base de données
  + Base
    - Table1
      * Champs1 : type
      * Champs2 : type
    - Table2
      * Champs1 : type
      * Champs2 : type
* SELECT
  + Sélectionner tous les enregistrements dans une table

SELECT \* FROM player

SELECT name, email FROM player

* Sélectionner les enregistrements correspondant à une condition

SELECT \* FROM player WHERE birthday >= ‘2000-01-01’

* Sélectionner les enregistrements correspondant à plusieurs conditions

SELECT \* FROM player WHERE birthday >= ‘2015-01-01’ AND points >= 50

SELECT \* FROM player WHERE birthday >= ‘2015-01-01’ AND points >= 50 OR name=’toto’

SELECT \*

FROM player

WHERE birthday >= ‘2015-01-01’

AND (points >= 50 OR name=’toto’)

* Sélectionner les enregistrements dont un champ respecte un « pattern »

%  : 0, 1 ou une infinité de caractères

\_  : un et un seul caractère

// adresse mail termine par mail.fr

SELECT \* FROM player WHERE email LIKE ‘%@mail.fr’

// nom commence par la lettre t

SELECT \* FROM player WHERE nom LIKE ‘t%’

// nom commence par la lettre t suivi d’un seul autre caractère

SELECT \* FROM player WHERE nom LIKE ‘t\_’

* Sélectionner les enregistrements dont la valeur d’un champ se trouve dans une liste prédéfinie

SELECT \* FROM player WHERE points = 45 OR points = 65 OR points = 85

SELECT \* FROM player WHERE points IN (45, 65, 85)

SELECT \* FROM player WHERE name IN (‘toto’, ‘gérard’, ‘abdel’)

* Sélectionner les enregistrements dont la valeur d’un champ se situe entre deux valeurs

SELECT \* FROM player WHERE birthday >= ‘2010-01-01’ AND birthday <= ‘2020-12-31’

SELECT \* FROM player WHERE birthday BETWEEN ‘2010-01-01’ AND ‘2020-12-31’

* ORDER BY : trier les résultats suivant un ou plusieurs champs (clause en fin de requête)

ASC : du plus petit au plus grand

DESC : du plus grand au plus petit

SELECT \* FROM player ORDER BY birthday ASC

SELECT \* FROM player ORDER BY birthday ASC, name DESC

* Exercice :

Ecrire les requêtes SQL pour récupérer les joueurs dans la table player :

* les enregistrements dont l’id est supérieur ou égal à 3

SELECT \* FROM player WHERE id >= 3

* les enregistrements dont le nom est Dupond et l’adresse mail est [dupond@mail.fr](mailto:dupond@mail.fr)

SELECT \* FROM player WHERE name = ‘Dupond’ AND email=’dupond@mail.fr’

* les enregistrements triés par date de naissance

SELECT \* FROM player ORDER BY birthday ASC

* les enregistrements dont la date de naissance est au moins 2005, triés par nom

SELECT \* FROM player WHERE birthday >= ‘2005-01-01’ ORDER BY name

SELECT \* FROM player WHERE YEAR(birthday) >= 2005 ORDER BY name

* les enregistrements dont l’adresse email termine soit par sfr.fr ou hotmail.fr

SELECT \* FROM player WHERE email LIKE ‘%@sfr.fr’ OR email LIKE ‘%hotmail.fr’

* les enregistrements dont le nom commence par « s » ou le nombre de points est compris entre 50 et 100

SELECT \* FROM player WHERE name LIKE ‘s%’ OR points >= 50 AND points <= 100

SELECT \* FROM player WHERE name LIKE ‘s%’ OR points BETWEEN 50 AND 100

* Regrouper des enregistrements : GROUP BY

// Sélectionner le nombre de joueurs par points

SELECT points, COUNT(\*) FROM player GROUP BY points

*// Sélectionner le nombre de points par équipe*

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) [SUM](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_sum)(points), team

FROM player

GROUP BY team

On peut cumuler plusieurs groupements :

*// Sélectionner le nombre de points par équipe par ville*

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) [SUM](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_sum)(points), team, zipcode

FROM player

GROUP BY team, zipcode

* Conditionner un GROUP BY : HAVING

*// Sélectionner le nombre de points par équipe en excluant les équipes qui moins de 60*

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) [SUM](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_sum)(points), team

FROM player

GROUP BY team

HAVING [SUM](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_sum)(points)>= 60

Là où le WHERE à filtrer les enregistrements, le HAVING sert à filtrer les groupements.

*// Sélectionner le nombre de points par équipe en excluant les équipes qui ont au moins 60, et en ne comptant pas le nombre de points des players qui nés après 2019*

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) [SUM](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_sum)(points), team

FROM player

WHERE YEAR(birthday) < 2019

GROUP BY team

HAVING [SUM](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_sum)(points)>= 60

* Requête d’insertion : INSERT INTO

[INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO player (id, name, email, birthday, points)

[VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (NULL, 'Bulbizar', 'bubul@mail.fr', NULL, '13');

Si un champ peut être null, il n’y a pas obligation à le préciser dans une requête INSERT INTO.

[INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO player (id, name, email, points)

[VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (NULL, 'Bulbizar', 'bubul@mail.fr', '13');

On peut ne préciser aucun, dans ce cas, il faut indiquer une valeur pour chaque champ dans VALUES, dans l’ordre des champs de la table.

[INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO player

[VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (NULL, 'Bulbizar', 'bubul@mail.fr', NULL, '13');

On peut réaliser plusieurs insertions dans la même requête :

[INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO player (id, name, email, birthday, points)

[VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (NULL, 'Bulbizar', 'bubul@mail.fr', NULL, '13'), (NULL, 'Pikachu', pika@mail.fr', NULL, '45');

* Requête de mise à jour : UPDATE

[UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) player [SET](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/set.html) points = '29' WHERE id = 6;

[UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) player [SET](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/set.html) points = points + 10 WHERE birthday = '2015-07-01';

La clause WHERE fonctionne exactement comme dans la requête de sélection. Ici elle sert à préciser quels sont les enregistrements à modifier.

Sans clause WHERE, la modification affectera tous les enregistrements de la table :

[UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) player [SET](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/set.html) points = '29';

Pour modifier plusieurs champs en même temps, il suffit de les séparer par des virgules :

[UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) player [SET](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/set.html) name = 'toto', email = 'new@mail.fr' WHERE id = 6;

* Requête de suppression : DELETE FROM

[DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) FROM player WHERE id = 6;

[DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) FROM player WHERE YEAR(birthday) = 2005;

[DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) FROM player;

* Fonctions scalaires

Une fonction scalaire, c’est une fonction qui va appliquer un traitement sur un champ de chaque enregistrement.

* + UPPER / LOWER : majuscule / minuscule

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) UPPER(name) FROM player

* + REVERSE : inverser les caractères
  + CONCAT

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) CONCAT(name, '/', email) FROM player

* + TRUNCATE : troncature

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) TRUNCATE(price, 0) FROM product

* + ROUND : arrondi

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) ROUND(price, 1) FROM product

* + YEAR, MONTH, DAY, HOUR, MINUT, SECOND
* Fonctions d’agrégation

C’est une fonction qui s’applique sur un ensemble de valeur : elles regroupent les lignes.

* COUNT
* SUM
* AVG
* MIN / MAX
* GROUP\_CONCAT : lorsqu’un GROUP BY est présent dans la requête, la fonction permet d’obtenir toutes les valeurs qui ont été groupées, séparées par défaut par une virgule.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) [SUM](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_sum)(points), team, [GROUP\_CONCAT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_group_concat)(name SEPARATOR ',') FROM player GROUP BY team

* Exercices

Ecrire les requêtes qui :

* Sélectionnent les joueurs qui habitent à Paris (75000) et qui ont plus de 75 points, ainsi que les joueurs qui habitent à Lille et qui ont plus de 50 points.

SELECT \* FROM player WHERE zipcode = ‘75000’ AND points > 75 OR zipcode=’59000’ AND points > 50

* Sélectionnent le nombre de points par ville, en excluant les villes dont le nombre de points est inférieur à 30.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) [SUM](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_sum)(points), zipcode

FROM player

GROUP BY zipcode

HAVING [SUM](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_sum)(points) >= 30

* Sélectionnent les joueurs qui habitent à Arras (62000), qui ont une adresse mail chez wanadoo.fr et dont le nombre de points est compris entre 15 et 25

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \*

FROM player

WHERE zipcode = '62000'

[AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_and) email [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-comparison-functions.html#operator_like) '%@wanadoo.fr'

[AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_and) points BETWEEN 15 [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_and) 25

* Sélectionnent les noms et équipes des joueurs qui habitent à Lille en ayant un nombre de points entre 50 et 60 et qui font partie de l’équipe 1, ainsi que les noms et équipes des joueurs de l’équipe 3 dont le nom commence par « t »

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) name, team

FROM player

WHERE zipcode = '59000'

[AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_and) points BETWEEN 50 [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_and) 60

[AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_and) team = 1

[OR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_or) team = 3

[AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_and) name [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-comparison-functions.html#operator_like) 't%'

* Sélectionnent le nombre de joueur dont le mois de naissance est octobre, en comptant uniquement les joueurs qui ont plus de 30 points

SELECT COUNT(id) FROM player WHERE MONTH(birthday) = 10 AND points > 30

## Algorithmie