**Cours du 12 décembre**

**INTRODUCTION HTML - CSS .**

Professeur : Benoit CHAUVET

Markup -> balises ! HTML = langage de balises.  
Elles s’ouvrent, et se ferment.

Touche F12 permet le débug !

CTRL + F5 : actualiser en forçant le navigateur à vraiment prendre la dernière version (si soucis)

&nbsp ; est un caractère d’espace html.

Pour afficher <br/> on écrit : &lt;br/&gt;

target="\_new" dans un href permet ouverture autre onglet.

<!-- le alt= est lu uniquement si le src ne l'est pas -->  
<img src=""  
alt="Il normalement une image mais ne s'affiche pas"/>

Span (en inline) et Div (block) s’apparentent à des labels / pannels.

La notation héxadécimale s’organise comme le rgb : deux caractères pour rouge, deux pour green, deux pour blue.

Conseil prof : utilisation de w3School pour trouver les couleurs

Balises utiles :  
<header>  
<nav>  
<section>  
<article>  
<aside>

On peut choisir un id pour un élément, il doit être unique pour toute la page !

Idem, on peut utiliser les class sensibles à la casse,  
Faire gaffe, se mettre d’accord avec la team en début de projet.

Pour faire référence à une ID dans du CSS, on utilise #  
id = ‘toto’ on va faire #toto

Utiliser des générateurs de box shadow tel que [celui-ci](http://www.debray-jerome.fr/outils/Generateur-de-box-shadow-css3.html)

Jeu sur sélecteurs (donné par prof) utile pour apprendre CSS : [voila](https://flukeout.github.io/)

[ZenGarden](http://www.mezzoblue.com/zengarden/alldesigns/) pour voir l’impact de CSS sur le visuel.

[Alsacréation](https://www.alsacreations.com/) pour des astuces ciblées et nombreuses.

[JSbin](https://jsbin.com/?html,css,output) pour taper du code avec choix des langages sortants. Très bien pour illustrer directement les propriétés qu’on teste.

Margin : 10px 20px fonctionne comme « haut et bas : 10px, droite et gauche : 20px »

Display :inline-block 🡪 pas bien (pas gérable vraiment) mais relativement efficace

Display :block 🡪 par défaut

Flexbox : [cours OpenClassroom](https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3/la-mise-en-page-avec-flexbox)

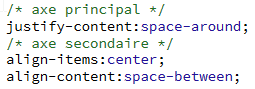
display:flex;  
Sur le border, permet alignement.

Très bon cours sur les flexbox : [CSS-Tricks](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)

Flex-wrap :nowrap ; 🡪 pour préserver  
Aussi : wrap-reverse

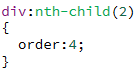
justify-content:space-between; justify-content:center;  
Pour gérer l’espace entre les div.

align-items:stretch; align-items:center;  
Centré l’item sur la ligne ou quoi.



Subtilité : si on est en mode « row » c’est l’axe principal, et la colonne l’axe secondaire. Ca s’inverse si on est en mode « column » !

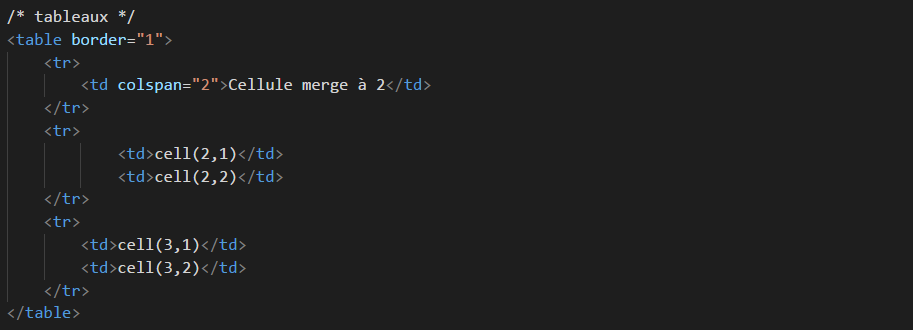
Changer l’ordre :

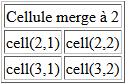


Quant aux flex-grow 🡪 permet l’étirement.  
On peut faire un truc fixe, et le reste qui s’étire lorsqu’on tend la page !  
Y penser pour les menus.  
Divise l’espace en plus, entre les div du cadre, selon la proportion de leurs flex-grow.

Les média query : important pour le responsive design !  
Transitions & animations à regarder.

Tableaux (html) :





[Excel – Mise en forme conditionnelle](http://formations-excel.blogspot.fr/2016/06/utiliser-formule-regle.html)

Ether et PPT PayPie : [clic](https://etherdelta.com/#PPT-ETH)

Site BTC etc : [clic](https://www.worldcoinindex.com/coin/funfair)

Ikea.aspx 🡪 le serveur qui reçoit ça forward au moteur d’exécution ASP.NET.

Ce dernier s’occupe de :  
.Création de la request / response  
.Analyse du aspx  
.instanciation : new Ikea()  
.initialisation (aspx) 🡪 il crée label1.text = « votre nom », l’abonnement à l’événement et l’événement, etc  
.load 🡪 page\_load  
.Rendering : écriture de la réponse.

Attention aux POST BACK : premier accès à la page est un GET, puis tout est contenu dans un form

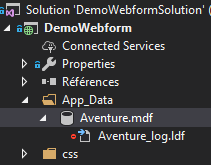
Attention aussi au VIEWSTATE pour des choses qui changent / ne changent pas

.initialisation  
.chargement de la viewstate  
.récupération des données POST  
.load 🡪 page\_load  
.traitement événements (btnOk\_Click)  
.Rendering (écriture de la réponse)  
 .Génération Viewstate

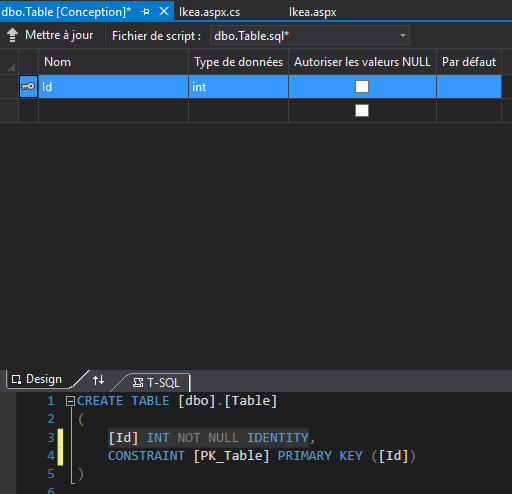
**ORM - OBJECT RELATIONAL MAPPING .**

Hibernate  
1. Créer la bdd 🡪 SQL Server  
2. Créer un EntityDataModel (DAOs + entités)  
3. Jouer avec

On a un dossier App\_Data  
Dedans, un nouvel élément : Aventure.mdf (connection serveur)

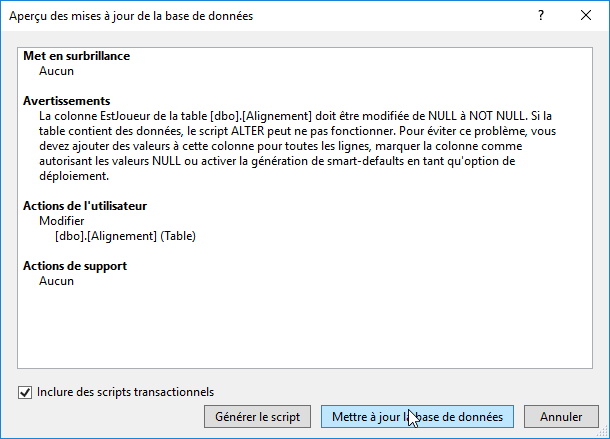


Ajout d’une table dessus :



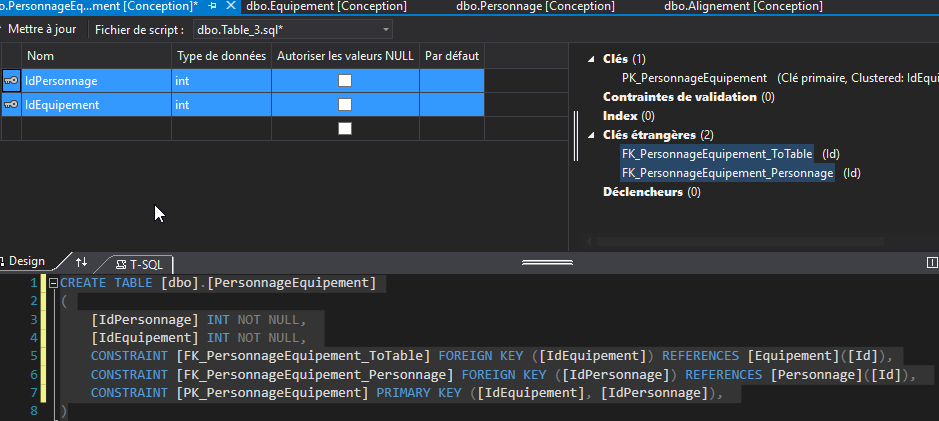
Nb : IDENTITY permet de créer une séquence, pour identifier en incrémentant.

On ajoute les autres propriétés, et on met à jour (en haut à gauche) :

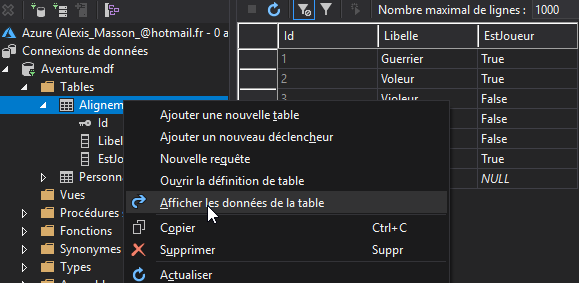


Création d’autres tables,

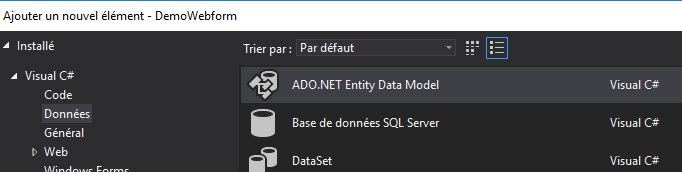
Et ici on définit sur DEUX lignes la PK, et attention aux FK aussi.

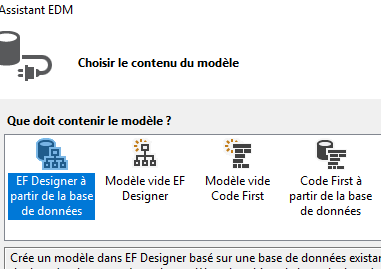


Ajout des données :

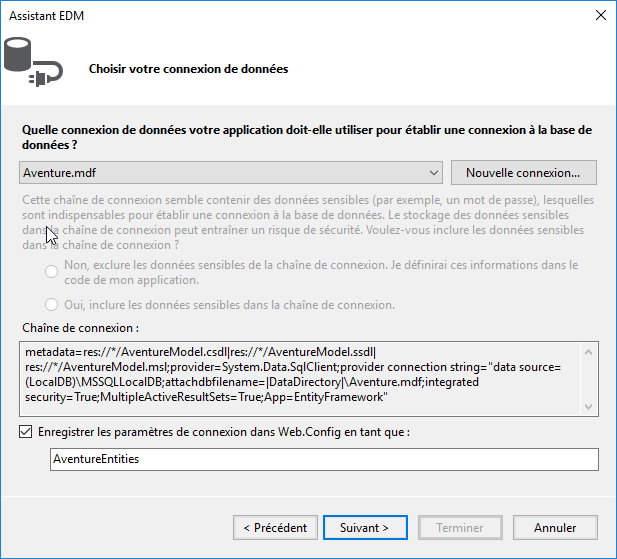


On crée ensuite via clic droit sur le projet -> nouvel élément (CTRL MAJ A)

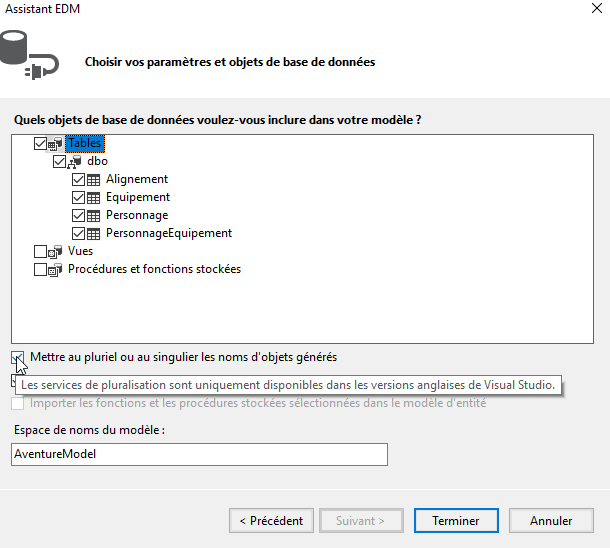


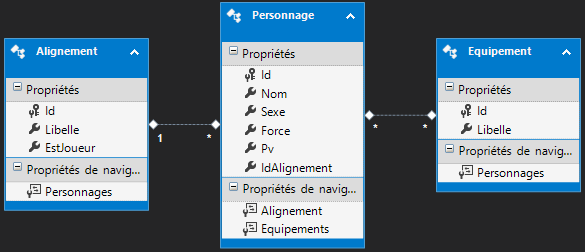
Ajout :  


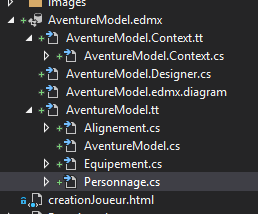
Suivant, la chaine de connexion est proposée sur notre AventureEntities :

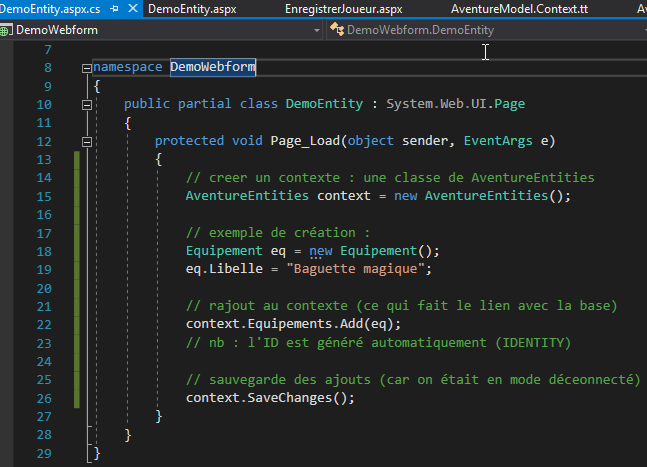


Bien cocher la case :





Dans le Model.tt on a des classes préfabriquées, ne pas y toucher car si on change un truc et met à jour le model, c’est ici que c’est changé :  




Voir :

<asp:GridView ID="gvEquipement" runat="server" AutoGenerateColumns="false"></asp:GridView>

Ne pas recharger le ViewState à chaque fois : enableViewState = « false »

On peut le faire directement dans la directive de page, la première ligne.

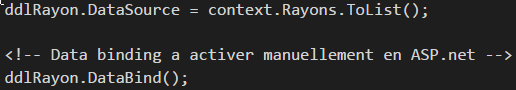
MASTER PAGE

1. Créer un projet  
   (ASP.NET 🡪 web application) 🡺.Net framework 🡪 boutique  
   🡪 projet VIDE
2. Créer une page Maitre (Arche), page Maitre Webform 🡪 insérer le contenu de la maquette
3. Créer une page ASPX (Accueil) 🡪 webform avec page Maitre
4. Créer une bdd « magasin »  
   créer dossier ASP.NET : AppData 🡪 bdd SqlServer 🡪 magasin
5. Ajout des tables  
   Article :  
   ID  
   Libelle  
   PrixUnitaire  
   Description  
   IDRayon

Pour appliquer uniquement aux div non asp.net :  
div :not(.aspNetHidden) ou  
.AspNetHidden  
{  
 display :none ;  
}

Pour les listes :





Dans CréerArticle.aspx.cs

btnEnregistrer\_Click 🡪 Article a = new Article();  
puis  
a.lib etc  
a.PrixUnitaire = double.Parse(txtPU.Text) ;  
a.IdRayon = int.Parse(ddlRayon.SelectedValue) ;

Puis ajout à la bdd

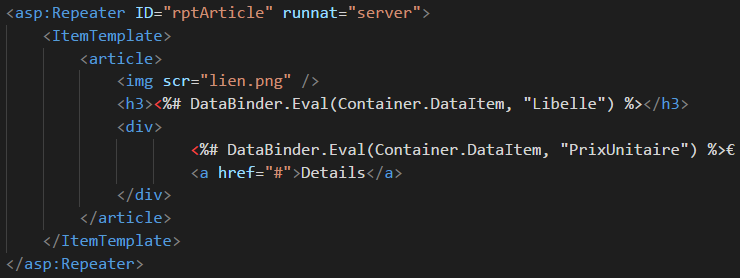
Boutique context = new BoutiqueEntities() ;  
context.Articles.Add(a) ;

Context.SaveChanges() ; 🡪 pour l’écriture dans la base.

Attention : initier la DropDown se fait bien dans le Page\_Load,  
MAIS il faut l’insérer dans un if(!IsPostBack)

Response.Redirect(«Accueil.aspx » ) ;  
  
code 302, c’est une redirection. En fait c’est d’abord dire au navigateur « va sur cette page », on repasse donc par le navigateur !

Repeteur (dans la boite à outils) : <ItemTemplate>  
Permet d’ajouter les articles en répetant une action.



Voir querystring  
a href= « ficheArticle.aspx ?id=293892»

Quand on voit <%# : le # veut dire que l’intérieur des <% %> ne sera évalué qu’à l’étape du DataBinding  
Généralement c’est utilisé pour les DataBinder.Eval.

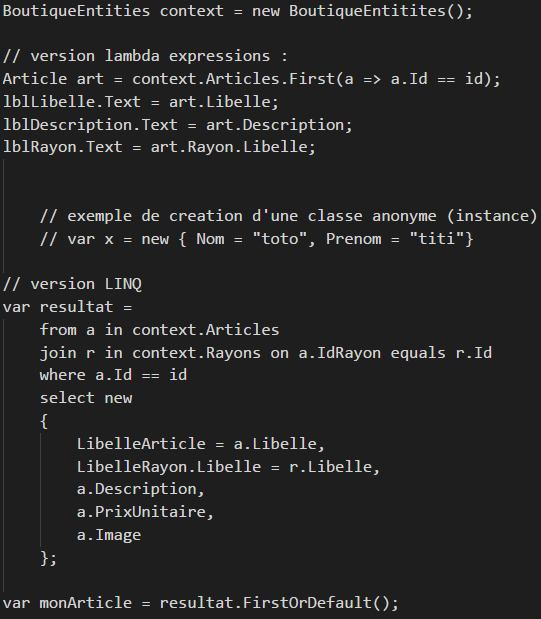
LINQ = Language Integrated Querying

Deux approches : l’approche lambda Expression, ou l’approche LINQ.  
  
C’est comme écrire dans du code C# avec des mots cles (select, from, etc : mots reconnus par C# et contrôlés à l’exécution).

Ex : context.Articles.Where(a => a.Prix > 10 && a.Prix < 20).ToList()

Cela utilise les IQueryable (sur dbSet)

Voir à ce sujet dans le poly C#



ASP.NET permet d’encapsuler ; permet création de pages web structurées à la manière de webform, mais c’est plus compliqué à mettre en place / gérer car il y a des échanges de données entre différents acteurs.

Page ASPX : Toto.aspx  
Dedans, il y a des balises <% page … Toto %> : celle là tout en haut du fichier  
Classe toto, dans le fichier **Toto.aspx.cs** (fichier C#) contenu de la partial class.  
On a ici le html tel quel, les balises html.

Dans le **designer Toto.aspx.designer.cs** (quasiment vide), on peut déclarer les contrôles, c’est l’autre bout de la partial class.

Dans la page aspx Toto.aspx, on fait glisser les contrôles :  
<asp :TextBox id= «txtNom» runnat= «server»></asp :TextBox>  
<asp :label id= « lblMsg » runnat= «server»></asp :label>  
<asp :Button id= «btnOk » runnat= «server»></asp :Button>

Les balises **asp:** ont forcément un attribut **runnat=server**

Lorsqu’on glisse les contrôles, ils sont créés aussi dans le designer.

Quand on est dans le fichier avec le html, le toto.aspx, on peut faire « afficher le code » il y a le Page\_Load : on peut aussi décider d’initialiser ici.

Ainsi on peut mettre à jour dans deux endroits, et ça à son importance dans le cycle de vie.

Cycle de vie : [clic](https://dotnet.developpez.com/faq/asp/csharp/?page=wflyfecycle#lifecycle)

**Etude du cycle de vie :**  
  
  
**ETAPE 1 : instanciation**  
Le client envoie un GET au serveur, qui crée un New Toto()  
Dans le Toto(), il y a un txtNom et un lblMessage avec des propriétés (backcolor et text, et forecolor et Text)  
Ces deux sont NULL à ce moment.  
Création aussi de Request et de Response qui contiennent déjà des choses.

**ETAPE 2 : chargement (load)**  
Parcours du Toto.aspx et chargement en fonction : le lblMsg a pour ForeColor fushia, et pour text « bonjour ».

***nb : les étapes en bleu sont s’il y a un postback***

**ETAPE 3 : chargement du viewstate (sorte de SAV)**  
  
**ETAPE 4 : chargement du formulaire (POST)**Les modifications par l’utilisateur sont prises en compte ici.

**ETAPE 5 : exécution du page\_load**Ici : if not isPostBack !  
Si on ne le met pas, les données entrées par l’utilisateur ne sont pas prises en compte, écrasées !!

**ETAPE 5bis : btnOk\_Click()**

**ETAPE 6 : création du Viewstate**

Si le mec revient sur la page, on doit mémoriser pour son prochain passage un viewstate, sorte de sauvegarde de la page avant de renvoyer !  
Sauvegarde l’état de tous les contrôles, et pas de html.

**ETAPE 7 : rendering**

**Response.Write()** : écriture de la réponse avec le html tel quel.  
Les balises <asp : blabla> sont interprétés, transformés en html !  
Ex pour le textbox : il devient <input type= « text » value= «Bob »>

Le viewstate est à cette étape mis dans un champs caché

Classe d’application globale [clic](https://dotnet.developpez.com/faq/asp/csharp/?page=globalasax)  
Global.asax 🡪 ne pas toucher au nom, et en créer une unique par projet

var monArticle = resultat.FirstOrDefault();

1° Ajouter en BDD les tables :  
- Commande  
- CommandeLigne

2° MAJ le modèle (edmx)  
3° Ajouter un nouveau Panier en session au démarrage  
4° Modifier :  
- ficheArticle (bouton Ajouter)  
- Arche (affichage du nombre d’Articles)

Dans le web.config on peut gérer le temps de sessions :  
sessionState timeout=«3»  
  
On peut ainsi compter le nombre de visiteurs totaux, ou en cours de visite.  
On utilise Global.asax et les fonctions incluses.

// envoie d'un cookie :

HttpCookie cookie = new HttpCookie("dateDebutPartie");

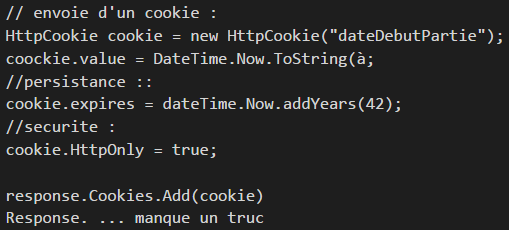
coockie.value = DateTime.Now.ToString(à;

//persistance ::

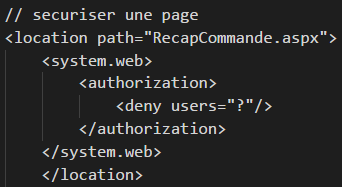
cookie.expires = dateTime.Now.addYears(42);

//securite :

cookie.HttpOnly = true;



Fichier de configuration pour accorder les droits :  
Web.config



Créer CTRL MAJ A : fichier de configurations  
On y met le

<authorization>  
<deny users=’’?’’/>

On gère les autorisations au niveau du web.config  
  
Dans le projet on crée Login.aspx à la racine (page de login).

Pour masquer le texte (visuellement uniquement, pas de cryptage) on ajoute au <asp : un attribut TextMode= « Password »

using System.Web.Security;

Quant à la connexion

