TME SERVLETS

Avant de commencer vous devez installer tout ce qui est nécessaire (JBoss ou un autre serveur d'application, Eclipse avec support de votre serveur d'application).

Pour les deux exercices ci-dessous il est possible de n'utiliser que des servlets http mais que cela ne vous empêche pas de tester avec des servlets génériques ou des servlets de base. Ne pas faire de CSS !

1. CREATION D'UN FORMULAIRE D'AUTHENTIFICATION

Objectif : créer plusieurs versions d'un formulaire d'authentification, permettant à un utilisateur de rentrer un login et un mot de passe pour s'authentifier et le rester (avec le contexte ou la session). Faire aussi en sorte que le titre de la page, les logins et mots de passe soient des **paramètres d'initialisation** de la servlet (pas de base de données pour l'instant).

Version 1 : le formulaire est créé via un fichier html classique et seule la validation se fait via une servlet. La servlet doit afficher un message pour valider ou invalider l'authentification. En cas de login/mot de passe invalide, le formulaire doit être affiché à nouveau.

Version 2 : le formulaire est lui aussi créé avec la même servlet, par exemple via une méthode de la classe.

Version 3 : deux servlets distinctes sont utilisée, une pour créer le formulaire et une pour l'authentification. Selon le cas, l'un peut appeler l'autre (**forward** ou **include**).

2. CREATION D'UN CHAT

Objectif : créer un « chat » pour les utilisateurs authentifiés.

Créer une classe Chat pour enregistrer tous les messages entrés par les utilisateurs, par exemple avec un attribut List<String> avec quelques méthodes pour ajouter un message, afficher la liste des messages (rien de bien complexe, ce n'est pas le but)... En parallèle, créer les formulaires nécessaires pour qu'un utilisateur puisse ajouter des messages.

Créer le chat, insérer l'objet dans la **session** et tester.

Ajouter un **listener** pour qu'à chaque fois qu'un utilisateur se connecte ou se déconnecte des messages soient ajoutés automatiquement dans le chat.

Perfectionner le chat à volonté.

TME JSP

L'objectif est de créer un site web de vente en ligne simpliste. Le but n'est pas faire des classes Java complexes mais d'utiliser à bon escient les beans, librairies de tags, …

1. FORMULAIRES POUR AJOUTER/SUPPRIMER UN PRODUIT DANS UN PANIER

Objectif : créer plusieurs classes JAVA qui serviront de beans par la suite.

- Gestion d'article : nom, description, prix, photo…
- Gestion de panier : ajout/suppression d'article,…

Créer les pages JSP associés :

- Une page qui affiche le contenu du panier et permet d'enlever des articles ;
- Une page qui affiche les articles disponibles et permet d'en ajouter dans le panier.

Création d'une servlet minimale pour traiter les formulaires si besoin et insérer/supprimer les choses dans le panier.

Tout cela doit se faire au niveau session.

2. UTILISATION DE TAGS

Objectif : tester les fonctionnalités de tag en jsp, créer par exemple des tags pour afficher un article ou le panier.

3. FORMULAIRES D'ADMINISTRATION

Objectif : refaire « la même chose » pour qu'un administrateur puisse modifier les articles disponibles et leur prix. Cela doit se faire au niveau application.

TME JDBC

Avant de commencer, vous devez installer un serveur mysql.

1. OPERATIONS DE BASE AVEC BASES DE DONNEES

Ecrire une classe de connexion à la base de données, permettant de se connecter et d'exécuter des requêtes SQL.

Prendre vos projets antérieurs et faire en sorte que tout se qui devrait se trouver en base de données y soit :

- Application chat : les logins/mot de passe doivent être en base de données. Ajouter des fonctionnalités permettant à un utilisateur de créer un compte.
- Application magasin. Les produits doivent être ajoutés dans la base et pas dans un bean de niveau applicatif. Le panier est toujours stocké dans un bean mais les articles associés sont dans la base.

2. CLASSE DE LOGIN

Ecrire une classe de login générique permettant :

- La création d'un compte
- La connexion
- La modification des informations personnelles.

Tout doit être configurable au maximum (nom de la table, contenu des informations utilisateurs, …)

TME JAVASCRIPT

1. EXERCICES DE BASE

Créez des fonctions javascript pour :

- faire un jeu « plus petit plus grand » (même format que la fonction factorielle vue en cours) ;
- afficher le contenu d'un fichier Json dans un tableau. Le fichier Json sera par exemple dans le format suivant, avec un nombre de champs d'entête quelconque et un nombre d'entrées quelconque:
  {entete: ["e1",...], contenu: { "user1": ["ad",...], "user7": ["dg",...], ...}}
- finir le « moteur de recherche » vu en cours en Ajax. Coté serveur vous pouvez utiliser du Java ou plus simplement du Json.

2. VALIDATION DE FORMULAIRE

Commencez par vous faire la main en créant des fonctions simples pour valider un formulaire, par exemple des fonctions isEmail, isInteger, isNotEmpty, checkRegexp, …

Transformez votre code en un prototype de validation de formulaire qui permettra :

- de choisir l'identifiant du formulaire à valider ;
- d'ajouter des validateurs de champs avec plusieurs paramètres tels que : identifiant du champ, obligatoire ou pas, type du champ (entier, texte libre, email, …), …
- un slot de validation globale du formulaire.

Vous pouvez ensuite modifier votre classe pour offrir d'autres services, par exemple :

- si le formulaire contient deux champs email, il faudrait pouvoir vérifier que les deux emails sont identiques ;
- le choix de l'affichage des erreurs devrait être laissé libre, par exemple une méthode « addErrorZone » pourrait permettre de spécifier ou afficher les erreurs. De même si un champ est en erreur la validation pourrait redonner le focus à cette zone.

TME JQUERY

1. EXERCICES DE BASE

Créez le code nécessaire pour transformer un tableau simple en échiquier (les lignes et colonnes du tableau sont alternativement de deux couleurs).

Testez les appels ajax en jQuery. Par exemple en récupérant un fichier Json.

Créez un menu déroulant à plusieurs niveaux qui s'ouvre et se ferme avec un simple survol de la souris.

Créez une fonction pour rendre un bloc éditable, c'est-à-dire qu'en cliquant sur une balise on le transforme en input en qu'en cliquant en dehors on le retransforme en la balise précédente. Voir un exemple sur http://www.appelsiini.net/projects/jeditable/default.html . L'objectif n'est bien sur pas de refaire le plugin complet mais juste de pouvoir tester la transformation.

2. PLUGIN JQUERY

Créez un plugin jQuery complet avec plusieurs méthodes associées pour l'un des exercices de base plus haut.

TME CANVAS

L'objectif du TME est de développer un casse-brique complet et fonctionnel en HTML5 via l'utilisation de la balise canvas, comme par exemple :

http://kodogames.com/game/brickout/desktop/#

Les différentes fonctions utiles sont décrites plus précisément dans la documentation :

http://www.w3schools.com/tags/ref_canvas.asp

1. CREATION DE LA BALISE CANVAS

La balise canvas est une balise HTML5 normale :

```
<canvas id="canvas" width="300" height="300"></canvas>
```

Pour créer le canvas en JavaScript il suffit de récupérer l'élement du DOM associé :

```
var canvas = document.getElementById("canvas");
if (canvas.getContext) {
  console.log(canvas.width + " " + canvas.height);
} else {
  console.log("canvas non disponible");
}
```

Et enfin de récupérer le contexte (dessin 2D ici) :

```
  contexte = canvas.getContext("2d");
```

A partir de maintenant on peut commencer à dessiner directement via le contexte.

2. DESSIN DE LA BALLE

Utilisation de la méthode arc(centre_x, centre_y, rayon, angle_debut, angle_fin, horloge) : trace un arc de cercle de centre et de rayon fixé allant de l'angle angle_debut à angle_fin (de 0 à 2*PI pour un cercle complet), dans le sens des aiguille d'une montre, ou inversement.

```
ctx.beginPath();
ctx.arc(100,75,50,0,1*Math.PI);
ctx.fill();
```

```
ctx.arc(100,75,50,1*Math.PI, 2*Math.PI);
ctx.stroke()
```

Méthodes utiles :

beginPath() : début d'un chemin

stroke() : tracé du chemin

fill() : remplissage du chemin

3. MOUVEMENT DE LA BALLE

Nécessite d'effacer tout régulièrement puis d'afficher la balle à sa nouvelle position. Par exemple avec un callback appelé toutes les 50ms :

```
function drawBall() {
  ctx.clearRect(0, 0, largeur, hauteur);
  ...
}
setInterval(drawBall, 50);
```

Avec une fonction draw qui calcule la nouvelle position de la balle et l'affiche. Par exemple la balle peut se déplacer d'un pixel vers la droite et un pixel vers le haut. La balle n'est donc pas uniquement une position et un rayon mais également un vecteur de déplacement.

Pour que tout fonctionne, vous pouvez créer un objet balle contenant tout ça. Prévoyez qu'il pourra éventuellement y avoir plusieurs balles en jeu à terme.

Assurez que la balle ne disparaisse pas hors du canvas mais au contraire rebondisse sur ses limites. Il faut donc détecter si la balle sort des limites et si c'est le cas changer son vecteur de direction. Pour cela vous pouvez simplement comparer les coordonnées de la balle à celles de la zone, où tester la méthode isPointInPath() :

```
ctx.rect(20,20,150,100);
if (ctx.isPointInPath(20,50)) {
  ...
}
```

4. BRIQUES

Les briques sont des rectangles (pour faire simple) mais d'autres formes sont possibles. Le tracé générique se fait avec les méthodes déjà vues, mais également :

moveTo() : se déplacer à une position sans rien dessiner

lineTo() : de déplacer en dessinant

closePath() : revenir au point initial du chemin

Mais il existe également des fonctions pour dessiner directement des rectangles plus simple que de tracer les 4 cotés :

rect() : crée un chemin rectangulaire

fillRect(), strokeRect() : crée et dessine ou rempli le rectangle

Prévoir un moyen de stocker des briques (rectangulaires pour faire simple) et de les afficher. Il faut également penser à un détecteur de collision afin de savoir si la balle entre en contact avec une brique. C'est un peu de géométrie mais une version simple (et certainement incorrecte) peut se faire avec isPointInPath(). Il faut bien entendu détruire toute brique touchée.

5. RAQUETTE

La raquette (ou le vaisseau spatial) est également un rectangle placé en bas du canvas qui se contrôle soit avec la souris, soit avec le clavier. La version souris a été vue en cours et, pour le clavier, cela peut se faire de la manière suivante avec un événement jQuery :

```
$(document).keydown(
  function (evt) {
    if (evt.keyCode == 39) { /*right*/ }
    else if (evt.keyCode == 37) { /*left*/ }
  }
);
```

Comme pour les bordures, il faut prévoir que la balle puisse rebondir sur la raquette. Par contre si la balle touche la bordure basse le joueur a perdu.

6. FINITIONS

Pour rendre le jeu plus agréable, il faudrait prévoir les choses suivantes, par ordre croissant de difficulté :

- Le fond peut être un dégradé (createLinearGradient) ou une image (drawImage) au lieu d'être uniforme.
- La vitesse de déplacement de la balle peut augmenter au fur et à mesure.
- En fonction de l'endroit où la balle touche la raquette, l'angle de rebond change.
- Plusieurs balles peuvent être en jeu.
- Les briques peuvent nécessiter plusieurs coups pour être détruites.
- La destruction des briques rapporte des points et le score est affiché (voir fonctions texte).

- Les scores peuvent être stockés en localStorage.
- Il peut y avoir plusieurs niveaux de jeu.
- Les briques peuvent contenir des options (agrandissement ou rétrécissement de la raquette, option multiballes, option raquette collante, option laser).
- Etc.