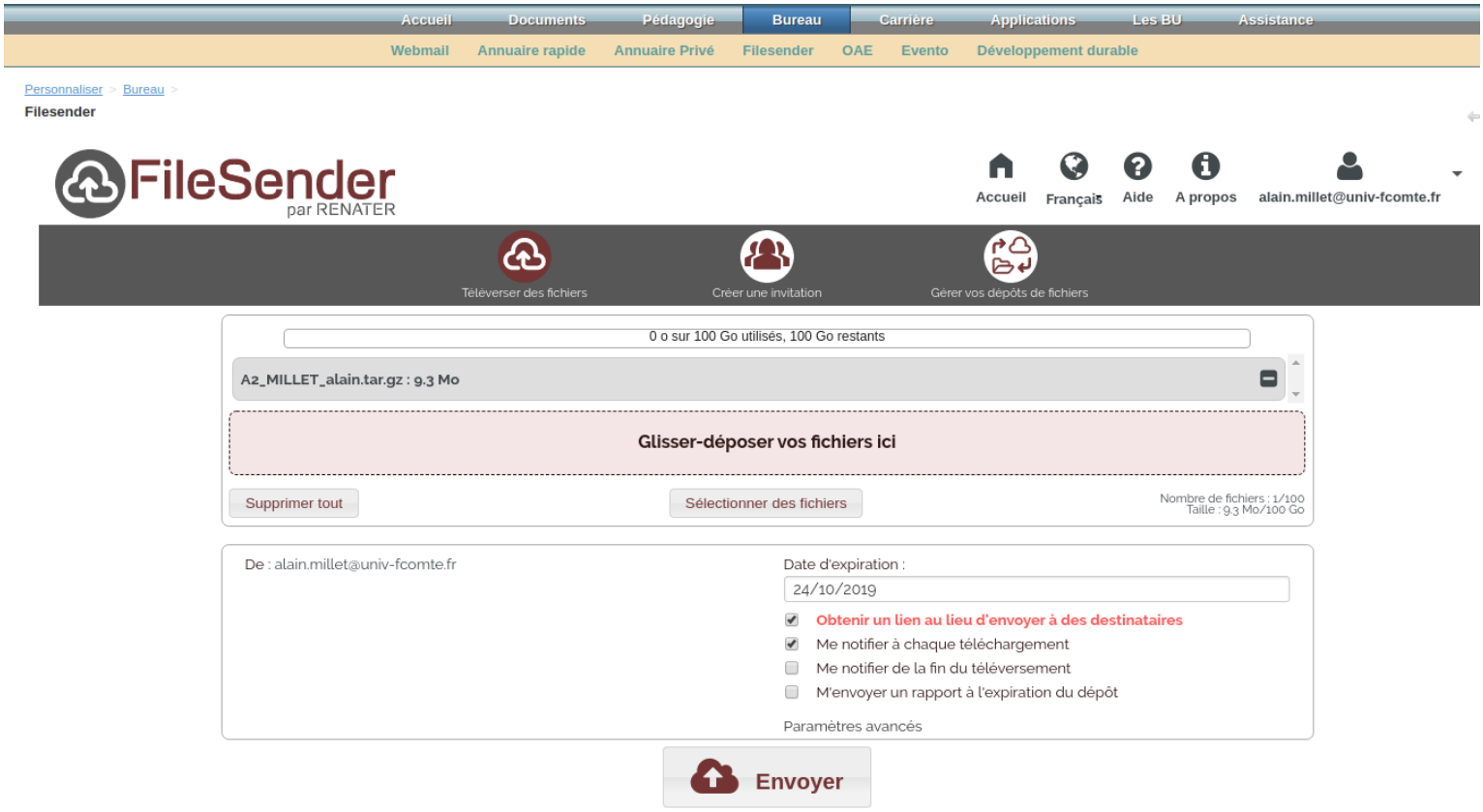
Document à rendre

Vous devez rendre un fichier avec :

- Dans le sujet du mail : **groupe nom prénom** **(sinon => pénalité)**
- Les codes sources dans une archive utilisable sous linux (**tar.gz** ou **zip**) dans un dossier qui porte dans l'ordre votre groupe, nom et votre prénom, exemple **A2_MILLET_alain.tar.gz** , **(sinon => pénalité)**
- une feuille (donnée en TP) que vous compléterez, qui indique les parties réalisées. Vous la rendrez en main propre la première séance de cours après la date ci dessous (sinon => pénalité, si vous la perdez, vous pouvez la ré-imprimer),
- Utiliser FileSender (sur l'ENT de l'université Bureau>FileSender).

Le document (fichier ressource) est à rendre pour le Samedi 2 Novembre 2019 …. minuit (-3 points par jour de retard),

Un oral de 10 à 15 minutes est prévu pour tester le fonctionnement et vérifier que vous avez bien compris le code que vous présentez.



Envoyer un lien à alain.millet@univ-fcomte.fr , vous serez notifié du téléchargement

****

Vous disposez de 2 tables (***table1*** et ***table2***) différentes pour chaque sujet
Vous remplacerez les noms *table1* et *table2* par les noms des tables de votre sujet (commencé en TP).

Si les consignes ci-dessus ne sont pas respectées, une grosse pénalité sera appliquée !

Le programme doit obligatoirement respecter l'architecture MVC vue en TD et TP sur symfony. Le programme doit obligatoirement utiliser des classes dans le frameWork symfony avec l'ORM doctrine et le moteur de template Twig.
Pour les modèles et les contrôleurs : Les classes et les méthodes doivent avoir comme nom : un nom qui fait référence à la ressource à modifier (le nom de la table).
Il en est de même pour le nom des vues, des routes et des urls : on doit retrouver une référence au nom de la table (ressource) que vous utilisez. Ceci afin d'éviter au maximum les copier/coller de code d'un projet à l'autre

Pour les vues dans le code HTML utilisé : pour réaliser les formulaires, vous n'utiliserez que des champs d'entrées <INPUT> de type « text » ou « submit », les champs de contrôle comme "date", "email", "number" …. ne sont pas autorisés dans l'HTML : ceci afin de vérifier que le contrôle des données est bien fait coté serveur.
DONC
REQUIRED, PATTERN, type= "number" , type="date" … sont interdits dans les vues « html »

Terminer le sujet donné en TP lors de l'évaluation

Modification à la maison

Partie 1 : Base de données : fixtures avec des dates (Prendre en compte le type « date »)

- Modifier les fixtures pour enregistrer des dates dans la base de données (validation en affichant la table dans un terminal)

Partie 2 : affichage de la *table1*

- **afficher les dates au format jj/mm/aaaa dans les vue showTable1s**

Partie 3 : suppression d'un enregistrement dans la *table1*

- vérifier la faille CSRF
- **Demander confirmation avant de supprimer un enregistrement dans un formulaire.**
- **Utiliser une méthode « DELETE » pour valider la suppression de l'enregistrement**
- **mettre un message « flash »**

Partie 4 : ajout et modification d'un enregistrement dans la *table1*

- **vérifier la faille CSRF**
- **Utiliser un format jj/mm/aaaa**
  - vérifier que les dates sont dans ce format et que ce sont bien des dates (exemple avec : *checkdate()*), utiliser des fonctions précédentes (ne pas utiliser <input type=« date » ...)
- **mettre un message « flash »**
- **Utiliser une méthode PUT pour valider la modification de l'enregistrement**

Partie 5 menu et affichage de la deuxième table (*table2*) +mise en forme

- Réaliser un menu sur la page principale qui affiche 2 liens pour :
- afficher la ***table1***
- afficher la ***table2***

ce menu posséde ainsi un lien :

- ***un formulaire de connexion ou un lien de deconnexion***

- ***réaliser le contrôleur et la vue pour afficher la deuxième table***

Partie 6 Gestion Authentificaton + FireWall

- les connexions (authentification) sont possibles et fonctionnelles :
- **afficher la table2 : ce lien sera affiché si une personne est connectée (elle possède des droits en session)**
- **si la personne est connectée et si elle possède les droits d'administrateur alors elle visualise les liens pour créer/modifier/supprimer les enregistrements dans la *table 1 sinon ces liens n'apparaissent pas***

Partie 7 **finalisation : fonction d'agrégation + bootstrap**

- ***Utiliser une requete de type « select » dans le « Repository » pour calculer le nombre + la moyenne par valeur de chaque champ dans la deuxième table et afficher un tableau qui résume ces résultats en fin de vue « showTable1s »***

- ***Utiliser bootstrap pour mettre en forme votre projet***

Partie 8 objet validator de symfony

* Créer un deuxième contrôleur de nom **Entite1V2** (Entite1 est le nom de votre première entité)
* Utiliser l'objet « Validator » pour vérifier les données
* Intégrer l'appel de ce contrôleur (méthode show) dans votre menu pour afficher tous les enregistrements de votre table principale.
* prévoir les routes et les vues pour modifier, ajouter ou supprimer des enregistrements dans la base de données
* **toutes les routes commencent par « /v2/ »** puis sont identiques à celles du premier contrôleur

Remarque : mettre des messages d'erreur différents pour bien distinguer les 2 contrôleurs

Partie 9 objet form (formulaire de symfony)

* Créer un troisième contrôleur de nom **Entite1V3** (Entite1 est le nom de votre première entité)
* Utiliser l'objet form pour vérifier les données
* Intégrer l'appel de ce contrôleur (méthode show) dans votre menu pour afficher tous les enregistrements de votre table principale
* prévoir les routes et les vues pour modifier, ajouter ou supprimer des enregistrements dans la base de données
* **toutes les routes commencent par « /v3/ »** puis sont identiques à celles du premier contrôleur

Remarque : mettre des messages d'erreur différents pour bien distinguer les 2 contrôleurs

Partie 10 api sur votre entité

* Créer un quatrième contrôleur de nom **Entite1V4** (Entite1 est le nom de votre première entité)
* Créer une api qui va retourner les données
* Prévoir des routes pour modifier, ajouter ou supprimer des enregistrements dans la base de données
* **toutes les routes commencent par « /api/ »** puis sont identiques à celle du premier contrôleur

Faire un jeu de test avec **PostMan** pour vérifier les routes de votre api