Noms des professeurs:
Stéphane MALDINI

Intitulé du cours : Groovy, Grails

Durée : 4h

Consignes

- L'épreuve doit avoir lieu en salle machine.
- Toute l'épreuve se déroule sur un unique projet GRAILS.
- Le répertoire projet sera renommé au nom de l'étudiant auteur et compressé au format ZIP
- Aucun projet envoyé aprés l'épreuve ne sera évalué.
- Les commentaires et la clarté du code feront parti de l'appréciation.
- Vérifier que l'archive envoyée est valide, un projet non décompressible équivaut un projet non rendu.
- L'accés à Internet et aux supports de cours est autorisé.
- Si nécessaire, la documentation supplémentaire sera fournie dans un fichier README à la racine projet
- Il s'agit d'un examen individuel

Le projet développé pendant cet examen est un Quizz Web. Le but de l'application est de permettre à un utilisateur d'éditer une série de questions à choix multiples et d'inviter d'autres utilisateurs à répondre. Un classement pourra être publié et commenté par les participants.

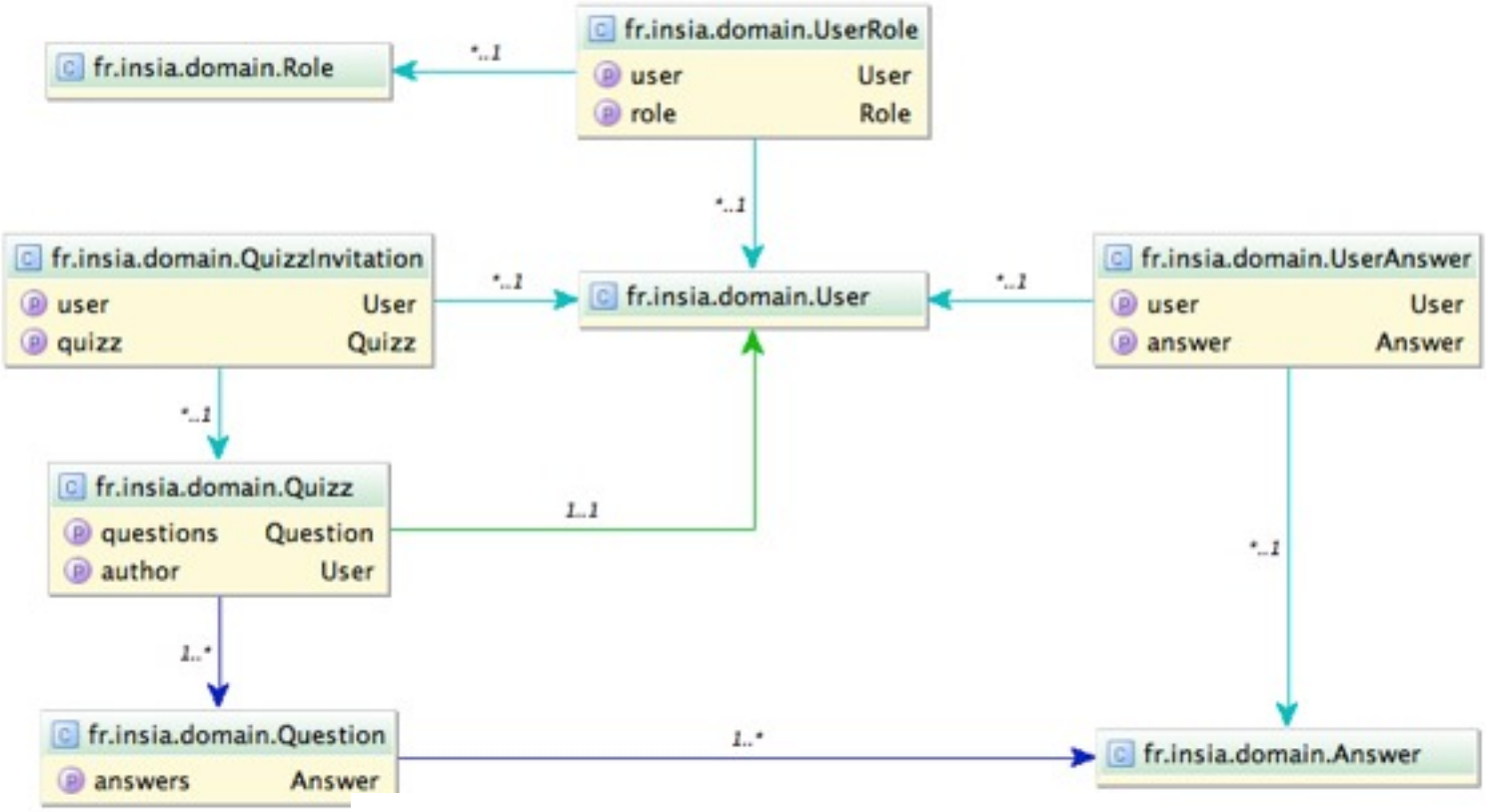
Vous utilisez l'archive zip fournie avec ce sujet et importerez le projet partielSIGL3 (à renommer avec nom-prenom sans accents/espaces) dans votre IDE. Utilisez soit l'option «Open projet», soit l'option «Import Grails application from existing sources» à partir du menu File>New Project.

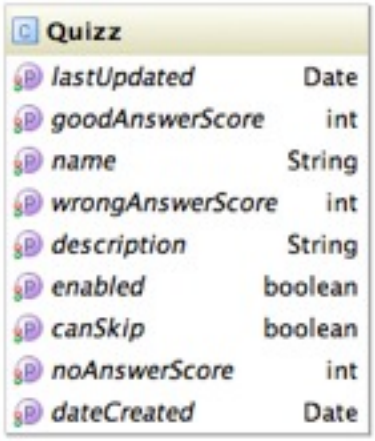
1 - Importer le projet

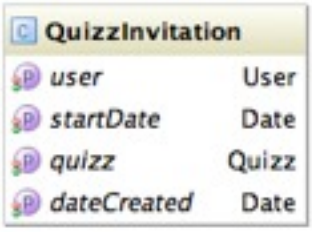
Les plugins JQuery et Spring-Security sont pré-installés.

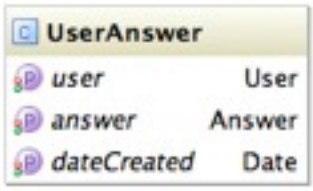
2 a- Implémenter le modèle métier suivant avec les contraintes listées ci dessous - 2,5 points

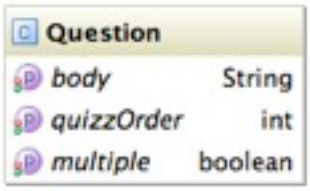
- User, UserRole et Role sont fournis par le projet
- Toutes les propriétés «body» sont des chaîne de caractères non-vides et d'au maximum 500 charactères
- Quizz.name ne doit pas être vide et doit contenir au maximum 150 charactères
- Quizz.description ne doit pas être vide et doit contenir au maximum 5000 charctères
- Quizz.questions doit contenir 40 éléments au maximum
- Question.quizzOrder est une propriété unique pour un quizz donné
- Les propriétés lastUpdated et dateCreated correspondent aux dates de mise à jour et de création en bd
- Par défaut, Quizz.goodAnswerScore = 1 , Quizz.wrongAnswerScore = -1, Quizz.noAnswerScore = 0
- UserAnswer est unique pour un user et une réponse donnée
- QuizzInvitation est unique pour un user et un quizz donné
- Quizz.enabled est à true par défaut

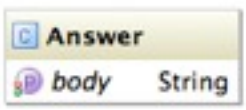

fr.insia.domain.Role
*..1
fr.insia.domain.UserRole
user User
role Role
*..1
fr.insia.domain.QuizzInvitation
user User
quizz Quizz
*..1
fr.insia.domain.User
*..1
fr.insia.domain.UserAnswer
user User
answer Answer
*..1
fr.insia.domain.Quizz
questions Question
author User
1..1
*..1
1..*
fr.insia.domain.Question
answers Answer
1..*
fr.insia.domain.Answer


Quizz
lastUpdated Date
goodAnswerScore int
name String
wrongAnswerScore int
description String
enabled boolean
canSkip boolean
noAnswerScore int
dateCreated Date


QuizzInvitation
user User
startDate Date
quizz Quizz
dateCreated Date


UserAnswer
user User
answer Answer
dateCreated Date


Question
body String
quizzOrder int
multiple boolean


Answer
body String

2 b - Ajouter une propriété définissant un QuizzInvitation.status systématiquement comprise dans les valeurs waiting/progress/complete - 0,5 point

3 a - Créer un service ImportService : - 4 points

- List<Quizz> parseFolder(String path) qui prend en paramètre un chemin de répertoire et retourne une liste de Quizz
  - Cette méthode devra lister tous les fichiers « .xml « du répertoire en paramètre
  - Les fichiers xml seront sous la forme « username_quizzname.xml»
  - Pour chaque xml, un Quizz doit être créé en appelant la méthode parseQuizz
- Quizz parseQuizz(File f) qui prend en paramètre un fichier de répertoire et retourne un objet Quizz
  - La partie username sera utilisée pour associer le nouveau quizz au user éponyme
  - Le contenu du xml sera parsé et analysé pour créer toutes les propriétés et tous les objets liés à un Quizz (Questions, Answers)
  - Un exemple complet de XML est fournit dans le répertoire data du projet. Tous les champs de ce fichier exemple doivent servir à générer un quizz , ses questions et les réponses associées
  - Utiliser l'attribut quizzOrder si présent sinon attribuer le quizzOrder d'une question selon l'ordre de parsing. RAPPEL : le quizzOrder doit être unique Quizz
  - La clarté, les commentaires, le niveau de Groovy et la performance du code seront évalués
- Utiliser ce service depuis le BootStrap et en mode «développement» afin d'importer les fichiers présents dans le répertoire data.

astuce : pour accéder au répertoire data, utiliser «BuildSettingsHolder.settings»

3b - Créer un service QuizzService : - 4 points

- Il exposera les méthodes :
  - int getScore(long QuizzId, long UserId = springSecurityService.principal.id)
    - Calcule le score d'un user sur un quizz
    - Si aucun userId n'est passé, récupérer la valeur de l'utilisateur connecté via springSecurityService.principal.id
  - void sendInvitations(long quizzId, List<Long> userIds)
    - Prend une liste de users Id et crée des QuizzInvitation
  - Question getCurrentQuestion(long quizzId, UserId = springSecurityService.principal.id)
    - Si aucun userId n'est passé, récupérer la valeur de l'utilisateur connecté via springSecurityService.principal.id
    - Retourne la première question non répondue de la liste des questions du Quizz classée par orderQuizz
  - List<Map> getRanking(long quizzId)

    - Retourne la liste ordonnée par score de tableaux associatifs [score : score , user : user] . Ex : [[score : 2, user: user1] , [score : 1, user:user2] ....] où user1 et users2 sont des objets de type User.
- void saveAnswer(long answerId, long UserId = springSecurityService.principal.id)
    - Créé un objet UserAnswer
    - Retourne une Exception si le Quizz.enabled est false
- void startQuizz(long quizzId, long UserId = springSecurityService.principal.id)
    - Démarre un Quizz en enregistrant la date courante dans l'objet QuizzInvitation correspondant au couple user/quizz
    - Retourne une Exception si le Quizz.enabled est false
- Collection<User> getParticipants(long quizzId)
    - Liste tous les participants d'un quizz via l'object QuizzInvitation
- List<User> getRannking(long quizzId)
    - Liste tous les participants d'un quizz via l'object QuizzInvitation
    - Ordonne par score le résultat
- List<Quizz> getLastCreatedQuizz(long UserId =null)
    - Si un user est utilisé en paramètre , retourner les 5 derniers quizz créés
    - Sinon affiché les 5 derniers quizz non filtrés
    - le résultat doit être ordonné par date décroissante
- List<Quizz> getLastAnswered(long UserId =null)
    - Si un user est utilisé en paramètre , retourner les 5 dernières réponses créées
    - Sinon retourner les 5 dernières réponses non filtrées
    - le résultat doit être ordonné par date décroissante
- boolean hasAccess(long quizzId , long userId = springSecurityService.principal.id)
    - retourne le résultat du prédicat : l'utilisateur possède-t-il une QuizzInvitation correspondant à ce quizzId
- boolean hasFinishedQuizz(long quizzId , long userId = springSecurityService.principal.id)
    - retourne le résultat du prédicat : l'utilisateur a son QuizzInvitation.status = complete
- Quizz save(Map params)
    - sauvegarde et retourne le nouveau Quizz

Il est demandé d'utiliser au maximum des requêtes SQL pour effectuer le traitement grâce notamment aux dynamic finders, Criteria, named queries, HQL. Attention, cela signifie également de prêter attention au nombre de requêtes sql effectuées sur une seule requête. Moins vous en faîtes, mieux vous vous porterez...

4 - Implémenter la couche visuelle - 6 points :

- Implémenter la logique de routage pour les URLs suivantes :
    - [adresse app]/ -> QuizzController - action:index
    - [adresse app]/uid/<username> -> QuizzController - action:index
    - [adresse app]/quizz/details/<id> -> QuizzController - action:show
    - [adresse app]/quizz/start/<id> -> QuizzController - action:show

  - [adresse app]/quizz/results/<id> -> QuizzController - action:results
  - [adresse app]/answer/<questionid> -> AnswerController - action:save
  - [adresse app]/question/<quizzid> -> QuestionController - action:showCurrent
  - [adresse app]/newQuizz -> QuizzController - action:create

- Le layout doit contenir :
  - Le titre WebQuizz
  - Seulement si authentifié :
    - Le menu new Quizz -> lien vers QuizzController - action:create
    - Le menu My Quizz -> lien vers QuizzController - action:index - param:username courrant
    - Le menu Username-> lien vers QuizzController - action:index - param:username courrant
    - Le menu sign off pour déconnecter (utiliser spring security)
  - Seulement si deconnecté :
    - Le menu pour se connecter

- Tous les controllers (Answer, Question, Quizz) doivent être accessible si et seulement si l'utilisateur a le rôle ROLE_USER
  - excepté pour l'action index de Quizz

- QuizzController - action:index - vue:oui (exemple mockup)
  - Afficher les 5 dernières réponses et les 5 derniers quizz filtrés par le user si paramètre username présent
  - Comme suggeré dans le mockup, les noms de quizz sont des liens vers [adresse app]/quizz/details/<id>
  - Egalement dans le mockup, les noms d'utilisateur sont des liens vers [adresse app]/uid/<username>

- QuizzController - action:show - param:quizzid - vue:oui (exemple mockup)
  - Afficher les détails de la question <quizzid>
  - Comme suggeré dans le mockup, les noms de quizz sont des liens vers [adresse app]/quizz/details/<id>
  - Dans le mockup, les nom de l'auteur est lien vers [adresse app]/uid/<username>
  - Si l'utilisateur courant est l'auteur
    - Le status est un lien vers [adresse app]/quizz/switch/<quizzid>
  - Sinon
    - Le status est affiché (actif si enabled:true et inactif si enabled:false)
  - Si lle quizz est enabled et que l'utilisateur en cours a une quizzInvitation dont le status est différent de complete
    - Afficher un lien «Démarrer» qui redirige sur [adresse-app]/quizz/star/<quizzId>
  - Sinon si

- QuizzController - action:switch - param:quizzid

  - Redirige sur show/<quizzid>
  - Si l'utilisateur courant est l'auteur
    - Le status est mis à jour

- QuizzController - action:results - param:quizzid - vue:oui
  - Afficher le classement sous forme de tableau html pour un quizz donné (colonnes score / username)

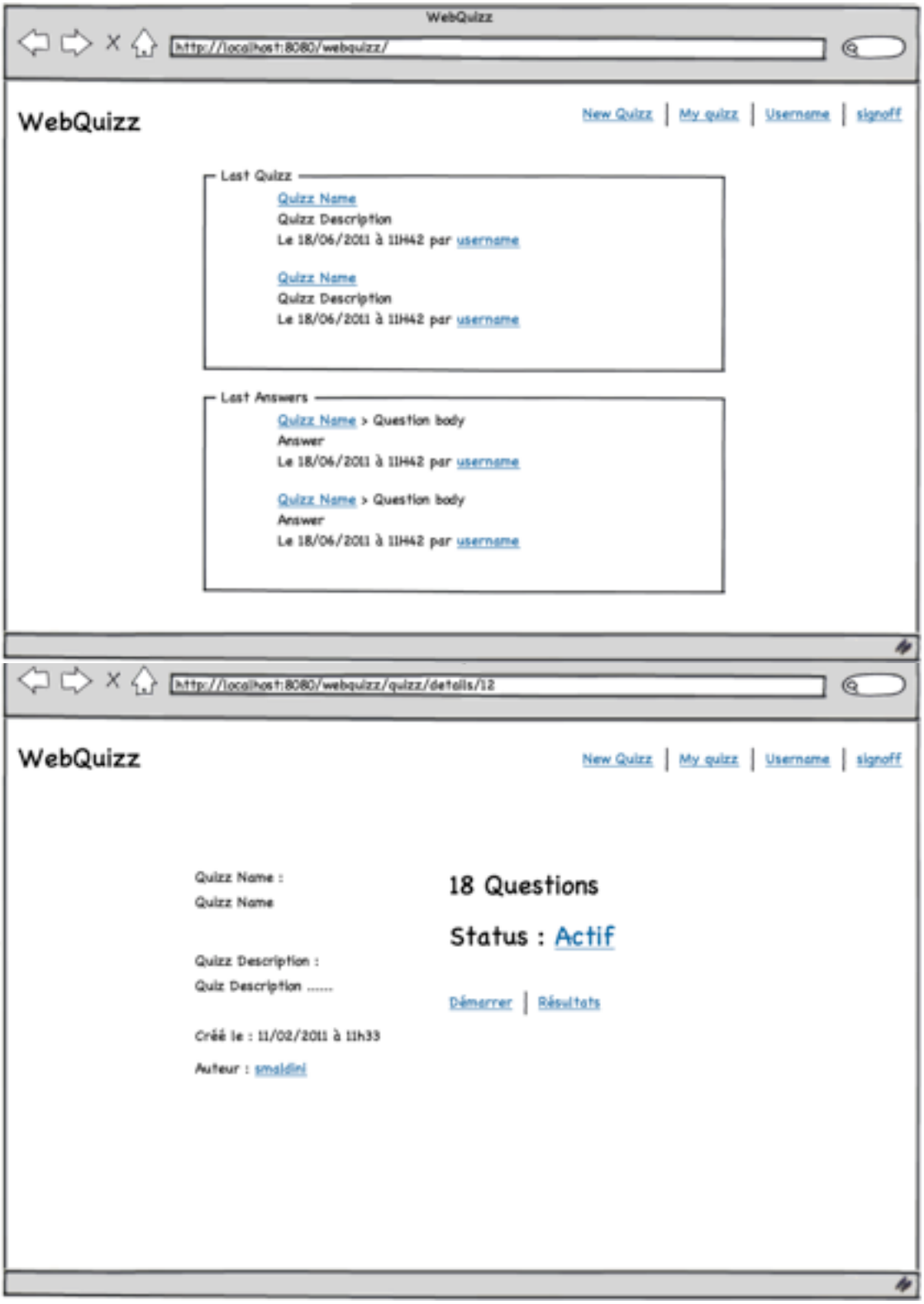
- AnswerController - action:save - param:questionid
  - Sauvegarde une réponse pour une question donnée et l'utilisateur en cours
  - Obligation d'utiliser une requête POST
  - Redirige sur showCurrent/<quizzid>
  - Stocker une erreur si la sauvegarde a échoué
- QuestionController - action:showCurrent - param:quizzid
  - Afficher la première question de liste fournie par un quizz, classée par quizzOrder, et pas répondue par l'utilisateur en cours (UserAnswer)
  - Afficher les réponses possibles dans un formulaire sous forme de liste radio (ajouter «Aucun» si le quizz en cours a la propriété canSkip = true). Si la réponse est à choix multiples, afficher une liste checkbox.
  - Afficher un bouton Suivant qui soumet le formulaire a [adresse app]/answer/save
  - Afficher une erreur si une sauvegarde de réponse a échoué précédemment

- QuizzController - action:create - vue:oui
  - Afficher un formulaire d'enregistrement d'un Quizz
  - Afficher les erreurs si un modèle «quizz» est présent dans les paramètres.
  - Utiliser une action QuizzController.save pour sauvegarder un nouveau Quizz
  - La page doit permettre d'éditer 3 questions et 3 réponses pour chacune
  - Le bouton Soumettre doit poster les données sur [adresse app]/quizz/save

- QuizzController - action:save params : formulaire
  - Sauvegarder le quizz soumis par formulaire
  - limite d'accés par POST
  - Si succès
    - Rediriger sur <adresse app]/quizz/details/<quizzid> où quizzid est l'identifiant du quizz sauvegardé
  - Sinon
    - Afficher la vue create avec le quizz erronné pour affichage des erreurs

- Les notions suivantes devront obligatoirement apparaître dans la réponse :
  - Utilisation d'au moins un TagLib

- Utilisation du Templating
- Utilisation du Layout
- Utilisation de Services dans les controllers/tagLib
- Ne pas hésiter à ajouter ses propres controllers/actions/vues si besoin est



5 - Implémenter le plugin quizzStats en intégrant le projet déjà créé dans le répertoire plugins/quizzStats : - 4 points

- Le sujet est libre, le plugin doit ajouter une fonctionnalité utile à l'application webquizz
- Le plugin doit
  - renseigner le champs description afin d'exposer la fonctionnalité du plugin
  - exposer au moins un bean Spring sans utiliser un artefact
  - ajouter au moins une méthode aux controllers
  - être compatible au possible avec le rechargement dynamique du mode développement