CookBook Spring

1. Importer le projet Maven
2. Écrire les classes métier dans le package business:

Pour chaque classe métier:

* + 1. ajouter un constructeur vide, sinon on obtient l’exception : org.hibernate.InstantiationException: No default constructor for entity: : fr.telecom\_st\_etienne.fx.enquete.business.Role
    2. ajouter un accesseur (méthode get) et un mutateur (méthode set) pour chaque attribut privé
    3. une méthode toString(): Spring va se servir de cette méthode pour générer les formulaires HTML utilisant les balises <form:form> et donner à chaque élément du formulaire la bonne valeur par défaut
    4. Annoter les classes business avec les annotations Hibernate (se référer au mémento Annotations)

Exemple:

@Entity

**public class** Question { @Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

**private int** id;

**private** String libelle;

@ManyToOne

**private** Enquete enquete;

...

}

1. Générer le diagramme de classes métier avec le plugin ObjetAid
2. Écrire les interfaces DAO. Chaque interface hérite de JpaRepository Exemple:

**public interface** QuestionDao **extends** JpaRepository<Question,

Long> {

}

1. Écrire les interfaces puis les classes dans le paquetage service. Annoter chaque classe service avec @Service. Injecter des objets DAO dans les services avec l'annotation @Autowired.

Exemple:

@Service

**public class** QuestionServiceImpl **implements** QuestionService {

@Autowired

**private** QuestionDao questionDao;

@Autowired

**private** EnqueteDao enqueteDao;

@Override

**public** Question recupererQuestion(Long id) {

**return** questionDao.findOne(id);

}

}

1. Écrire le ou les contrôleurs Spring. Annoter chaque classe contrôleur avec @Controller.
   1. (manière dépréciée) Injecter des objets de type Service dans les contrôleurs avec l'annotation @Autowired.

NB : Chaque objet de type Service doit être annoté @Autowired. Exemple:

@Controller

**public class** EnqueteController {

@Autowired

**private** EnqueteService enqueteService;

@Autowired

**private** QuestionService questionService;

* 1. (manière moderne, à préférer) Ajouter un constructeur dans le contrôleur avec en paramètre tous les objets de type Service :

Exemple:

@Controller

**public class** EnqueteController {

**private** EnqueteService enqueteService;

**private** QuestionService questionService;

**public** EnqueteController(EnqueteService enqueteService, QuestionService questionService) {

**super**();

**this**.enqueteService = enqueteService;

**this**.questionService = questionService;

}

}

* 1. Ajouter les méthodes nécessaires pour traiter toutes les requêtes HTTP, la méthode annotée @PostConstruct et la méthode annotée @InitBinder :

@Controller

**public class** EnqueteController {

**private** EnqueteService enqueteService;

**private** QuestionService questionService;

**public** EnqueteController(EnqueteService enqueteService, QuestionService questionService) {

**super**();

**this**.enqueteService = enqueteService;

**this**.questionService = questionService;

}

@RequestMapping(value = { "/index", "/" })

**public** ModelAndView accueil() {

ModelAndView mav = **new** ModelAndView("index"); mav.addObject("enquetes",

enqueteService.recupererEnquetes());

**return** mav;

}

@PostConstruct

**public void** init() {}

request) {

@InitBinder

**private void** initBinder(WebDataBinder binder, WebRequest

binder.registerCustomEditor(Enquete.**class**, "enquete", **new**

PropertyEditorSupport() {

@Override

**public void** setAsText(String text) **throws**

IllegalArgumentException {

System.***out***.println("Enquete setAsText: " + text); setValue((text.equals("")) ? **null** :

enqueteService.recupererEnquete (Long.*parseLong*(text)));

}

});

}

}