CookBook Spring

1) Importer le projet Maven

2) Écrire les classes métier dans le package business:

Pour chaque classe métier:

2.1.1) 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
```

2.1.2) ajouter un accesseur (méthode get) et un mutateur (méthode set) pour chaque attribut privé

2.1.3) 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

2.1.4) 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;
  …
}
```

3) Générer le diagramme de classes métier avec le plugin ObjetAid

4) Écrire les interfaces DAO. Chaque interface hérite de JpaRepository

Exemple:

```
public interface QuestionDao extends JpaRepository<Question,
Long> {

}
```

5) 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);
  }

}
```

6) Écrire le ou les contrôleurs Spring. Annoter chaque classe contrôleur avec @Controller.

6.1) (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;
```

6.2) (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;
  }

}
```

6.3) 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() {}

  @InitBinder
  private void initBinder(WebDataBinder binder, WebRequest
request) {
  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)));
        }
```

```
            });
            }
        }
```