

Subscribe to DeepL Pro to edit this document.  
Visit www.DeepL.com/Pro for more information.

Spring CookBook

1. Import the Maven project
2. Write the business classes in the business package:

For each trade class:

* + 1. add an empty constructor, otherwise you get the exception: org.hibernate.InstantiationException: No default constructor for entity: : fr.telecom\_st\_etienne.fx.enquete.business.Role
    2. add an accessor (get method) and a mutator (set method) for each private attribute
    3. a toString() method: Spring will use this method to generate HTML forms using the <form:form> tags and give each element of the form the right default value
    4. Annotate business classes with Hibernate annotations (refer to the Annotations memento)

Example:

@Entity

**public class** Question { @Id

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

**private int** id;

**private** String libelle;

@ManyToOne

**private** Enquete enquete;

...

}

1. Generate the business class diagram with the ObjectAid plugin
2. Write the CAD interfaces. Each interface inherits from JpaRepository Example:

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

Long> {

}

1. Write the interfaces then the classes in the service package. Annotate each service class with @Service. Inject CAD objects into services with @Autowired annotation.

Example:

@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. Write the Spring controller(s). Annotate each controller class with @Controller.
   1. (deprecated way) Inject Service type objects into controllers with the @Autowired annotation.

NB: Each Service type object must be annotated @Autowired. Example:

@Controller

**public class** EnqueteController {

@Autowired

**private** EnqueteService enqueteService;

@Autowired

**private** QuestionService questionService;

* 1. (modern way, to be preferred) Add a constructor in the controller with in parameter all objects of type Service :

Example:

@Controller

**public class** EnqueteController {

**private** EnqueteService enqueteService;

**private** QuestionService questionService;

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

**super**();

**this**. enqueteService = enqueteService;

**this**. questionService = questionService;

}

}

* 1. Add the methods needed to process all HTTP requests, the annotated @PostConstruct method and the annotated @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 home() {

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

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 (*ParseLong*(text));

}

});

}

}