**Exemples complets de codes en Java - EE**

|  |  |
| --- | --- |
| Envoi de de fichier via formulaire  vérification d'un fichier envoyé par formulaire avant sa copie sur le disque (avec gestion des erreurs) | Classe métier |
| public class UploadFiles {  public static final int TAILLE\_TAMPON = 10240; // 10 ko  private static final String CHEMIN\_FICHIERS = "c:/fichiersTmp/";  private static final String CHAMP\_FICHIER = "fichier";  private static final String CHAMP\_DESCRIPTION = "description";  private String resultat;  private Map<String, String> erreurs = new HashMap<String, String>();  public String getResultat() {  return resultat;  }  public Map<String, String> getErreurs() {  return erreurs;  }  public Fichier enregistrerFichier(HttpServletRequest request) throws IOException, ServletException {  /\* initialisation du bean représentant le fichier \*/  Fichier fichier = new Fichier();  /\*  \* Récupération du contenu du champ "description" du formulaire  \* (paramètre classique)  \*/  String description = getValeurChamp(request, CHAMP\_DESCRIPTION);  /\* Récupération du contenu du champ "fichier" du formulaire via Part \*/  String nomFichier = null;  InputStream contenuFichier = null;  try {  Part part = request.getPart(CHAMP\_FICHIER);  // On vérifie qu'on a bien reçu un fichier  nomFichier = getNomFichier(part);  // Si on a bien un fichier  if (nomFichier != null && !nomFichier.isEmpty()) {  // Corrige un bug du fonctionnement d'Internet Explorer  nomFichier = nomFichier.substring(nomFichier.lastIndexOf('/') + 1)  .substring(nomFichier.lastIndexOf('\\') + 1);  // Récupération du contenu du fichier  contenuFichier = part.getInputStream();  }  } catch (IllegalStateException e) {  /\*  \* Exception retournée si la taille des données dépasse les limites  \* définies dans <multipart-config> de notre servlet dans web.xml \*>  \*  \*/  e.printStackTrace();  erreurs.put(CHAMP\_FICHIER, "Les données envoyées sont trop volumineuses.");  } catch (IOException e) {  /\*  \* retournée si erreur au niveau du répertoire temporaire de  \* stockage défini dans web.xml  \*/  e.printStackTrace();  erreurs.put(CHAMP\_FICHIER, "Erreur de configuration du serveur lié au repertoire de stockage temporaire.");  } catch (ServletException e) {  /\*  \* retournée si la requête n'est pas de type "multipart/form-data".  \* Si essaie de contacter la servlet avec un autre formulaire que  \* celui proposé  \*/  e.printStackTrace();  erreurs.put(CHAMP\_FICHIER,  "Type de requête non supporté, merci d'utiliser le formulaire prévu pour envoyé votre fichier.");  }  /\*  \* Si aucune erreur n'est survenue : on a bien un fichier stocké dans le  \* dossier temporaire. On vérifie son contenue et celui de la  \* description  \*/  if (erreurs.isEmpty()) {  // validation du champ "description"  try {  validationDescription(description);  } catch (Exception e) {  erreurs.put(CHAMP\_DESCRIPTION, e.getMessage());  }  fichier.setDescription(description);  // Validation du champ fichier  try {  validationFichier(nomFichier, contenuFichier);  } catch (Exception e) {  erreurs.put(CHAMP\_FICHIER, e.getMessage());  }  fichier.setNom(nomFichier);  }  /\* Si aucune erreur alors écriture du fichier sur le disque \*/  if (erreurs.isEmpty()) {  try {  ecrireFichier(contenuFichier, nomFichier, CHEMIN\_FICHIERS);  } catch (Exception e) {  erreurs.put(CHAMP\_FICHIER, "Erreur lors de l'écriture du fichier sur le disque.");  }  }  /\* Initialisation du résultat global \*/  if (erreurs.isEmpty())  resultat = "Succés de l'envoi du fichier.";  else  resultat = "Echec lors de l'envoi du fichier.";  return fichier;  }  private static String getValeurChamp(HttpServletRequest request, String nomChamp) {  String valeur = request.getParameter(nomChamp);  if (valeur == null || valeur.trim().length() == 0)  return null;  else  return valeur;  }  private static String getNomFichier(Part part) {  for (String contentDisposition : part.getHeader("content-disposition").split(";")) {  if (contentDisposition.trim().startsWith("filename")) {  return contentDisposition.substring(contentDisposition.indexOf('=') + 1).trim().replace("\"", "");  }  }  return null;  }  private static void validationDescription(String description) throws Exception {  if (description != null) {  if (description.length() < 15) {  throw new Exception("La description du fichier doit contenir au moins 15 caractères.");  }  } else {  throw new Exception("Merci d'entrer une description du fichier.");  }  }  private static void validationFichier(String nomFichier, InputStream contenuFichier) throws Exception {  if (nomFichier == null || contenuFichier == null)  throw new Exception("Merci de sélectionner un fichier à envoyer.");  }  private static void ecrireFichier(InputStream contenuFichier, String nomFichier, String chemin) throws IOException {  /\* prépare les flux \*/  BufferedInputStream entree = null;  BufferedOutputStream sortie = null;  try {  /\* Ouvre les flux \*/  entree = new BufferedInputStream(contenuFichier, TAILLE\_TAMPON);  sortie = new BufferedOutputStream(new FileOutputStream(new File(chemin + nomFichier)), TAILLE\_TAMPON);  /\*  \* Lit le fichier reçu et écrit son contenu dans un fichier sur le  \* disque  \*/  byte[] tampon = new byte[TAILLE\_TAMPON];  int longueur;  while ((longueur = entree.read(tampon)) > 0) {  sortie.write(tampon, 0, longueur);  }  } finally {  try {  sortie.close();  } catch (IOException ignore) {  }  try {  entree.close();  } catch (IOException ignore) {  }  }  }  } | ATTENTION : ne gère pas le cas de fichiers de même nom ni de fichiers de contenu identique ! |
|  |  |
| Téléchargement de fichier  fait le lien entre un fichier demandé dans URL et le fichier sur le disque pour permettre son téléchargement | Servlet |
| public class Download extends HttpServlet {  private static final long serialVersionUID = 1L;  private static final String CHEMIN\_FICHIERS = "c:/fichiersTmp/";  private static final int DEFAULT\_BUFFER\_SIZE = 10240; // 10 ko  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  /\*  \* Récupère le nom du fichier placé en fin d'URL (avant les paramètres)  \*/  String fichierDemande = request.getPathInfo();  /\* Vérifie qu'un fichier a bien été demandé \*/  if (fichierDemande == null || "/".equals(fichierDemande)) {  // Envoie d'un erreur 404  response.sendError(HttpServletResponse.SC\_NOT\_FOUND);  return;  }  /\*  \* Décode le nom de fichier récupéré (gère espaces et caractères  \* spéciaux)  \*/  fichierDemande = URLDecoder.decode(fichierDemande, "UTF-8");  /\* Préparation de l'objet File \*/  File fichier = new File(CHEMIN\_FICHIERS, fichierDemande);  /\* Vérifie que le fichier existe bien dans le dossier \*/  if (!fichier.exists()) {  // alors envoie erreur 404  response.sendError(HttpServletResponse.SC\_NOT\_FOUND);  return;  }  /\*  \* Récupération du type du fichier afin de pouvoir remplir l'en-tête de  \* la réponse  \*/  String type = getServletContext().getMimeType(fichier.getName());  /\* Si le type du fichier est inconu, on initialise un type par défaut \*/  if (type == null) {  type = "appication/octet-stream";  }  /\* Initialisation de la réponse HTTP \*/  response.reset();  response.setBufferSize(DEFAULT\_BUFFER\_SIZE);  response.setContentType(type);  response.setHeader("Content-Length", String.valueOf(fichier.length()));  response.setHeader("Content-Disposition", "attachment; filename=\"" + fichier.getName() + "\"");  /\* Prépare les flux \*/  BufferedInputStream entree = null;  BufferedOutputStream sortie = null;  try {  entree = new BufferedInputStream(new FileInputStream(fichier), DEFAULT\_BUFFER\_SIZE);  sortie = new BufferedOutputStream(response.getOutputStream(), DEFAULT\_BUFFER\_SIZE);  // lit le fichier et écrit son contenu dans la réponse HTTP  byte[] tampon = new byte[DEFAULT\_BUFFER\_SIZE];  int longueur;  while ((longueur = entree.read(tampon)) > 0) {  sortie.write(tampon, 0, longueur);  }  } finally {  try {  sortie.close();  } catch (IOException ignore) {  }  try {  entree.close();  } catch (IOException ignore) {  }  }  }  } | ATTENTION : ne gère pas l'interruption de téléchargement pouvant reprendre par la suite (fonctionnalité intéressante pour les gros fichiers uniquement).  attachment; => permet l'ouverture ou le téléchargement du fichier  inline; => fichier s'ouvre en grand dans la page du navigateur (pas besoin de créer une JSP pour l'affichage) |
|  |  |
| Sérialisation  sauvegarde d'un objet dans un fichier externe (avec gestion des erreurs) | Classe métier |
| public class SaveObject {  private static final String CHEMIN\_FICHIERS = "c:/ZfichiersTmp/";  private Map<String, String> erreurs = new HashMap<String, String>();  private String resultat;  public String getResultat() {  return resultat;  }  public Map<String, String> getErreurs() {  return erreurs;  }  public void saveObjectInFile(ClasseObject monObjet) {  ObjectOutputStream oos = null;  try {  // localization du fichier cible ou création  oos = new ObjectOutputStream(  new BufferedOutputStream(new FileOutputStream(new File(CHEMIN\_FICHIERS, "monFichier.txt"))));  // Ecriture de l'objet dans le fichier  oos.writeObject(monObjet);  } catch (IOException e) {  // captée lors d'une erreur d'écriture  erreurs.put("Erreur écriture", e.getMessage());  } finally {  // On s'assurer de fermeture du flux  System.out.print("Début de Finally");  try {  if (oos != null)  oos.close();  } catch (IOException e) {  erreurs.put("Erreur fermeture", e.getMessage());  }  }  if (erreurs.isEmpty()) {  resultat = "Copie de l'objet dans fichier réussie";  } else {  resultat = "Erreur lors de la copie de l'objet dans le fichier";  }  }  } |  |
|  |  |
| Déserialisation  lecture d'un objet sérialisé dans un fichier externe (avec gestion des erreurs) | Classe métier |
| public class ReadObject {  private static final String CHEMIN\_FICHIERS = "c:/ZfichiersTmp/";  private Map<String, String> erreurs = new HashMap<String, String>();  private String resultat;  public String getResultat() {  return resultat;  }  public Map<String, String> getErreurs() {  return erreurs;  }  public ClassObject readObjectInFile() {  ObjectInputStream ois = null;  ClassObject monObject = null;  try {  // localization du fichier cible  ois = new ObjectInputStream(  new BufferedInputStream(new FileInputStream(new File(CHEMIN\_FICHIERS, "monFichier.txt"))));  // Lecture de l'objet dans le fichier  monObject = (ClassObject) ois.readObject();  } catch (FileNotFoundException e1) {  // captée si ois ne trouve pas le fichier indiqué  erreurs.put("Fichier non trouvable", e1.getMessage());  } catch (IOException e2) {  // captée lors d'une erreur de lecture  erreurs.put("Erreur de lecture", e2.getMessage());  } catch (ClassNotFoundException e3) {  // captée si classe de l'objet inadaptée  erreurs.put("Erreur de cast", e3.getMessage());  } finally {  // On s'assurer de fermeture du flux  try {  if (ois != null)  ois.close();  } catch (IOException e) {  erreurs.put("Erreur de fermeture", e.getMessage());  }  }  if (erreurs.isEmpty()) {  resultat = "Lecture de l'objet dans le fichier réussie";  } else {  resultat = "Erreur lors de la lecture de l'objet dans le fichier";  }  return monObject;  }  } |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |