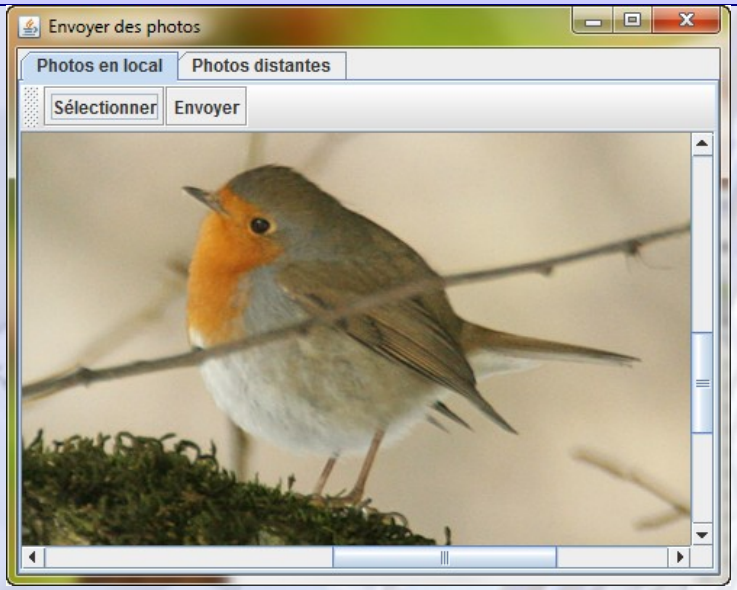
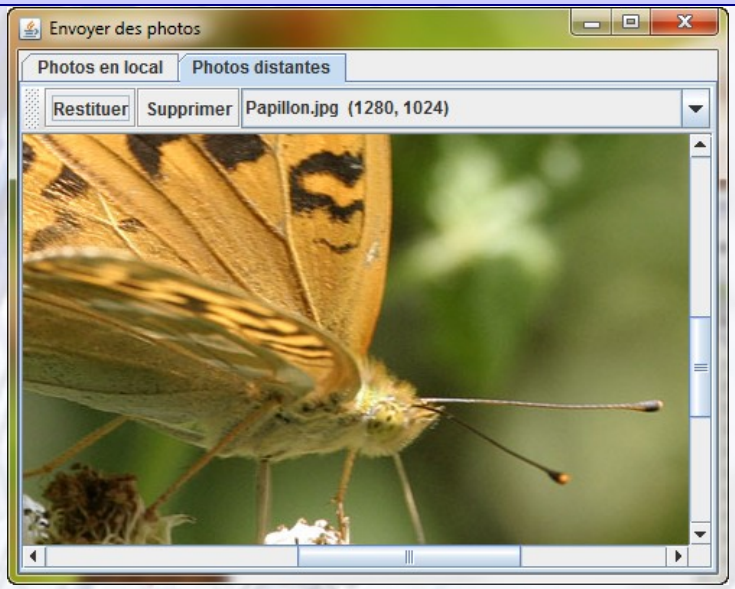
JAVA EE 6

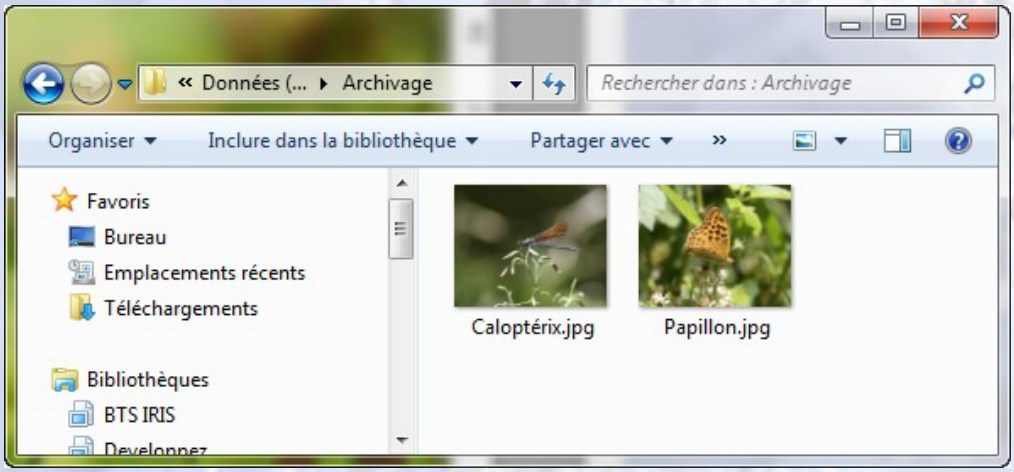
Ce TP permet de montrer comment résoudre les deux principaux problèmes importants lorsque nous mettons en œuvre les Web Services.

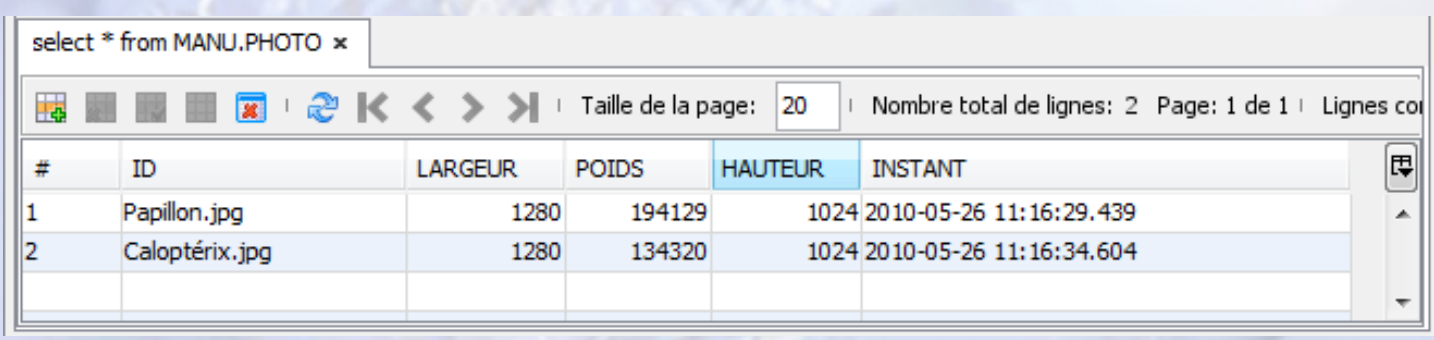
- *Le premier problème apparaît lorsque nous devons faire transiter des tableaux au travers du réseau, comme par exemple des tableaux d'octets représentant le contenu d'un fichier. La solution consiste à prendre systématiquement une collection de type **List** à la place du tableau. Une petite mise en forme est alors nécessaire.*
- *Le deuxième problème qui peut se présenter c'est lorsque nous devons faire passer des objets également au travers du réseau comme par exemple des beans entités. La solution ici consiste à utiliser la même technique que le Web Service en Java, c'est-à-dire l'API **JAXB**.*

VISUALISATION CÔTÉ CLIENT ET CÔTÉ SERVEUR









#	ID	LARGEUR	POIDS	HAUTEUR	INSTANT
1	Papillon.jpg	1280	194129	1024	2010-05-26 11:16:29.439
2	Caloptérix.jpg	1280	134320	1024	2010-05-26 11:16:34.604

x BEANS ENTITÉ

E:\BTS IRIS\TP Java\PhotoListe-ejb\src\java\entité\Photo.java

```
1 package entité;
2
3 import java.a
4 import java.io.
5 import java.uti
6 import javax.persist nce.*;
7 import javax.xml.bi d.annotation.XmlRootElement;
8
9 @XmlRootElement
10 @Entity
11 @NamedQuery(name="toutes", query="SELECT photo FROM Photo AS photo")
12 public class Photo implements Serializable {
13     @Id
14     private String id;
15     @Temporal(TemporalType.TIMESTAMP)
16     private Date instant;
17     private int largeur;
18     private int hauteur;
19     private long poids;
20
21     public String getId() { return id; }
22     public int getHauteur() { return hauteur; }
23     public Date getInstant() { return instant; }
24     public int getLargeur() { return largeur; }
25     public long getPoids() { return poids; }
26
27     public void setHauteur(int hauteur) { this.hauteur = hauteur; }
28     public void setId(String id) { this.id = id; }
29     public void setInstant(Date instant) { this.instant = instant; }
30     public void setLargeur(int largeur) { this.largeur = largeur; }
31     public void setPoids(long poids) { this.poids = poids; }
32
33     public Photo() { }
34
35     public Photo(String nom, long poids) {
36         id = nom;
37         instant = new Date();
38         this.poids = poids;
39     }
40
41     public void setDimensions(BufferedImage image) {
42         largeur = image.getWidth();
43         hauteur = image.getHeight();
44     }
45 }
```

Annotation qui permet de réaliser le parsing entre un document XML et le bean entité correspondant.

Doit être placé en premier.

Il est également impératif d'avoir systématiquement tous les getter et setter (toutes les propriétés) de tous les attributs que vous désirez faire transiter sur le réseau.

BEANS SESSION QUI SERT DE WEB SERVICE

E:\BTS IRIS\TP Java\PhotoListe-ejb\src\java\session\Archiver.java

```
1  package session;
2
3  import entité.Photo;
4  import java.awt.image.*;
5  import java.io.*;
6  import java.util.*;
7  import javax.ejb.*;
8  import javax.imageio.ImageIO;
9  import javax.jws.*;
10 import javax.persistence.*;
11
12 @WebService
13 @Stateless
14 public class Archiver  {
15    private final String répertoire = "E:/Archivage/";
16    @PersistenceContext
17    EntityManager persistance;
18
19    public void stocker(String nom, List<Byte> listeOctets) throws IOException {
20       byte[] octets = new byte[listeOctets.size()];
21       for (int i=0; i<listeOctets.size(); i++) octets[i] = listeOctets.get(i);
22       stocker(nom, octets);
23    }
24
25    public List<Byte> restituer(String nom) throws IOException {
26       List<Byte> octets = new ArrayList<Byte>();
27       for (byte octet : récupérer(nom)) octets.add(octet);
28       return octets;
29    }
30
31    private void stocker(String nom, byte[] octets) throws IOException  {
32       File fichier = new File(répertoire+nom);
33       if (fichier.exists()) return;
34       FileOutputStream photo = new FileOutputStream(fichier);
35       photo.write(octets);
36       photo.close();
37       enregistrer(nom);
38    }
39
40    @Asynchronous
41    private void enregistrer(String nom) throws IOException {
42       File fichier = new File(répertoire+nom);
43       BufferedImage image = ImageIO.read(fichier);
44       Photo photo = new Photo(nom, fichier.length());
45       photo.setDimensions(image);
46       persistance.persist(photo);
47    }
48
49    private byte[] récupérer(String nom) throws IOException {
50       File fichier = new File(répertoire+nom);
51       if (!fichier.exists()) return null;
52       FileInputStream photo = new FileInputStream(fichier);
53       byte[] octets = new byte[(int)fichier.length()];
54       photo.read(octets);
55       photo.close();
56       return octets;
57    }
58
59    public List<Photo> getListe() { return persistance.createNamedQuery("toutes").getResultList(); }
60
61    public Photo getPhoto(String nom) {  return persistance.find(Photo.class, nom);  }
62
63    public void supprimer(String nom) {
64       new File(répertoire+nom).delete();
65       Photo photo = getPhoto(nom);
66       persistance.remove(photo);
67    }
68 }
```

Méthode du Web service qui prend une liste d'octets en paramètre.

Méthode classique du bean session qui ne fera pas parti du Web Service et qui prend un simple tableau d'octets en paramètre.

Persistance des informations de chaque fichier image.

x PERSONNE GÉNÉRÉ AUTOMATIQUEMENT PAR WSIMPORT CÔTÉ CLIENT

E:\BTS IRIS\TP Java\ClientPhotoListe\build\generated-sources\jax-ws\ws\Photo.java

```
1 package ws;
2
3 import javax.xml.bind.annotation.*;
4 import javax.xml.datatype.XMLGregorianCalendar;
5
6 @XmlAccessorType(XmlAccessType.FIELD)
7 @XmlType(name = "photo", propOrder = {
8     "hauteur",
9     "id",
10     "instant",
11     "largeur",
12     "poids"
13 })
14 public class Photo {
15     protected int hauteur;
16     protected String id;
17     @XmlSchemaType(name = "dateTime")
18     protected XMLGregorianCalendar instant;
19     protected int largeur;
20     protected long poids;
21
22     public int getHauteur() {
23         return hauteur;
24     }
25
26     public void setHauteur(int value) {
27         this.hauteur = value;
28     }
29
30     public String getId() {
31         return id;
32     }
33
34     public void setId(String value) {
35         this.id = value;
36     }
37
38     public XMLGregorianCalendar getInstant() {
39         return instant;
40     }
41
42     public void setInstant(XMLGregorianCalendar value) {
43         this.instant = value;
44     }
45
46     public int getLargeur() {
47         return largeur;
48     }
49
50     public void setLargeur(int value) {
51         this.largeur = value;
52     }
53
54     public long getPoids() {
55         return poids;
56     }
57
58     public void setPoids(long value) {
59         this.poids = value;
60     }
61
62     @Override
63     public String toString() {
64         return id +"  ("+largeur+", "+hauteur+")";
65     }
66 }
```

Partie du code à rajouter pour que cela soit plus convicial côté client.

APPLICATION CLIENTE EN CONNEXION AVEC LE WEB SERVICE

E:\BTS IRIS\TP Java\ClientPhotoListe\src\photoliste\Client.java

```
1 package photoliste;
2
3 import java.awt.BorderLayout;
4 import java.awt.event.*;
5 import java.io.*;
6 import java.util.*;
7 import javax.imageio.ImageIO;
8 import javax.swing.*;
9 import ws.*;
10
11 public class Client extends JFrame {
12    private JTabbedPane onglets = new JTabbedPane();
13    private JPanel panneauLocal = new JPanel(new BorderLayout());
14    private JPanel panneauServeur = new JPanel(new BorderLayout());
15    private JToolBar outilsLocal = new JToolBar();
16    private JToolBar outilsDistant = new JToolBar();
17    private JLabel photoLocale = new JLabel();
18    private JLabel photoDistante = new JLabel();
19    private JLabel description = new JLabel();
20    private JComboBox listePhotos = new JComboBox();
21    private JFileChooser sélecteur = new JFileChooser();
22    private Photo photo;
23    private File fichier;
24    private byte[] octets;
25    private boolean effacer = true;
26
27    private static Archiver archivage;
28
29    public Client() {
30       super("Envoyer des photos");
31       add(onglets);
32       onglets.add("Photos en local", panneauLocal);
33       onglets.add("Photos distantes", panneauServeur);
34
35       panneauLocal.add(outilsLocal, BorderLayout.NORTH);
36       panneauLocal.add(new JScrollPane(photoLocale));
37       outilsLocal.add(new AbstractAction("Sélectionner") {
38          public void actionPerformed(ActionEvent e) {
39             sélecteur.setFileSelectionMode(JFileChooser.FILES_ONLY);
40             if (sélecteur.showOpenDialog(Client.this)==JFileChooser.APPROVE_OPTION) {
41                fichier = sélecteur.getSelectedFile();
42                photoLocale.setIcon(new ImageIcon(fichier.getPath()));
43             }
44          }
45       });
46       outilsLocal.add(new AbstractAction("Envoyer") {
47          public void actionPerformed(ActionEvent e) {
48             if (fichier!=null)
49                try {
50                   byte[] octets = new byte[(int) fichier.length()];
51                   FileInputStream lecture = new FileInputStream(fichier);
52                   lecture.read(octets);
53                   lecture.close();
54                   List<Byte> contenu = new ArrayList<Byte>();
55                   for (byte octet : octets) contenu.add(octet);
56                   archivage.stocker(fichier.getName(), contenu);
57                   listingPhotos();
58                }
59                catch (Exception ex) {
60                   setTitle("Impossible d'envoyer le fichier");
61                }
62          }
63       });
64
```

```
65      panneauServeur.add(outilsDistant, BorderLayout.NORTH);
66      panneauServeur.add(new JScrollPane(photoDistante));
67      panneauServeur.add(description, BorderLayout.SOUTH);
68
69      outilsDistant.add(new AbstractAction("Restituer") {
70         public void actionPerformed(ActionEvent e) {
71            sélecteur.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
72            if (sélecteur.showSaveDialog(Client.this)==JFileChooser.APPROVE_OPTION) {
73               try {
74                  fichier = new File(sélecteur.getSelectedFile() + "/" +photo.getId());
75                  FileOutputStream fluxImage = new FileOutputStream(fichier);
76                  fluxImage.write(octets);
77                  fluxImage.close();
78               }
79               catch (Exception ex) {  setTitle("Problème pour restituer la photo en local"); }
80            }
81         }
82      });
83      outilsDistant.add(new AbstractAction("Supprimer") {
84         public void actionPerformed(ActionEvent e) {
85            archivage.supprimer(photo.getId());
86            listingPhotos();
87         }
88      });
89      outilsDistant.add(listePhotos);
90      listePhotos.addActionListener(new ActionListener() {
91         public void actionPerformed(ActionEvent e) {
92            if (!effacer)
93               try {
94                  photo = (Photo) listePhotos.getSelectedItem();
95                  List<Byte> listeOctets = archivage.restituer(photo.getId());
96                  octets = new byte[listeOctets.size()];
97                  for (int i=0; i<listeOctets.size(); i++) octets[i] = listeOctets.get(i);
98                  ByteArrayInputStream fluxImage = new ByteArrayInputStream(octets);
99                  photoDistante.setIcon(new ImageIcon(ImageIO.read(fluxImage)));
100              }
101              catch (Exception ex) {  setTitle("Problème pour récupérer l'image du serveur");  }
102           }
103     });
104     listingPhotos();
105     setSize(500, 400);
106     setLocationByPlatform(true);
107     setDefaultCloseOperation(EXIT_ON_CLOSE);
108     setVisible(true);
109  }
110
111  private void listingPhotos() {
112     effacer = true;
113     listePhotos.removeAllItems();
114     for (Photo photo : archivage.getListe()) listePhotos.addItem(photo);
115     effacer = false;
116     listePhotos.setSelectedIndex(0);
117  }
118
119  public static void main(String[] args) {
120     archivage = new ArchiverService().getArchiverPort();
121     new Client();
122  }
123 }
124
```