

# Développement de l'application Vélib

Le parcours de  
documents XML



# Liens utiles

- <https://docs.oracle.com/javase/7/docs/api/org/w3c/dom/package-summary.html>
- <http://java.developpez.com/faq/xml/?page=dom>

# DOM

- // création d'une fabrique de documents
  - `DocumentBuilderFactory fabrique = DocumentBuilderFactory.newInstance();`
- // création d'un constructeur de documents
  - `DocumentBuilder constructeur = fabrique.newDocumentBuilder();`
- Nécessitent :
  - **import** `javax.xml.parsers.*;`
  - **import** `org.w3c.dom.*;`
- Lèvent l'exception :
  - `ParserConfigurationException`

# DOM (suite)

- // lecture du contenu d'un fichier XML avec DOM
  - `File xml = new File("ExempleDOM.xml");`
  - `Document document = constructeur.parse(xml);`
- // lecture d'un flux XML en ligne
  - `String urlCarto = "http://www.velib.paris.fr/service/carto";`
  - `Document document = constructeur.parse(urlCarto);`
- Lèvent :
  - `SAXException` et `IOException`
- Donc nécessitent :
  - **import** `org.xml.sax.SAXException;`
  - **import** `java.io.IOException;`

# DOM (suite)

- Il ne faut pas confondre Node et Element.
- **Node** (*org.w3c.dom.Node*) :
  - Un Node (ou noeud) est l'unité de base de l'arbre.
  - Cela peut être du texte, un élément, une portion CDATA ou encore une instruction.
  - Pour connaître le type d'un Node utilisez la méthode *getNodeTypes()*.
- **Element** (*org.w3c.dom.Element*) :
  - L'interface Element définit un élément au sens XML (XHTML ou HTML).
  - Un élément est constitué d'un tag, d'attributs et d'un contenu (autres nodes et éléments).

# Parcours de documents

- La première chose à faire pour commencer à parcourir votre arbre DOM est de **récupérer la racine**.
  - `Element racine = document.getDocumentElement();`
- A partir de cette racine, vous pouvez parcourir l'ensemble du document. Voici les méthodes à utiliser :
- **Méthodes de l'interface Element**
  - **`getElementsByTagName()`** : retourne une NodeList contenant les éléments enfants dont le tag correspond au nom passé en paramètre (\* pour renvoyer tous les éléments).
  - **`getAttribute("nom_att")`** : renvoie la valeur de l'attribut "nom\_att"
  - **`getTextContent()`** : retourne le contenu de l'Element

# Parcours de documents (suite)

- **Méthodes de l'interface Node**
  - **getChildNodes()** : retourne une NodeList contenant l'ensemble des nodes enfants.
  - **getFirstChild()** : retourne le premier Node enfant.
  - **getLastChild()** : retourne le dernier Node enfant.
  - **getNextSibling()** : retourne la prochaine occurrence du Node.
  - **getParentNode()** : retourne le noeud parent du Node.
  - **GetPreviousSibling()** : retourne la précédente occurrence du Node.

# Parcours de documents (suite)

- **Méthodes de l'interface NodeList**

- **getLength()** : retourne le nombre de Node de la liste.
- **item(i)** : retourne le ième item de la liste (0 pour le premier).
- Une boucle for-each n'est pas utilisable sur une NodeList.



# Développement de l'application Vélib

L'application



# Les liens utilisés

- <http://www.velib.paris.fr/service/carto>
- <http://www.velib.paris.fr/service/stationdetails/4017>

## Disponibilité des stations de Vélib

## Arrondissements de Paris

- ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5  
☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10  
☐ 11 ☐ 12 ☐ 13 ☐ 14 ☐ 15  
☐ 16 ☐ 17 ☐ 18 ☐ 19 ☐ 20

## Départements

- ☐ 92 ☐ 93 ☐ 94

## Autres

- ☐ mobile

## Disponibilité

1 RUE SAINT BON - 75004 PARIS

le 16/11/2014 à 12:12:11

Vélos disponibles : 10

Nombre total de points d'attache : 19

La station est ouverte.

Points d'attache disponibles : 9

Location par carte bancaire : OUI

Numéro	Adresse	Bonus	Ouvert
4006	FACE 1 BOULEVARD BOURBON - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4007	BOULEVARD BOURDON - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4009	6 RUE SAINT PAUL - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4010	105-109 TERRE PLEIN SAINT PAUL - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4011	FACE 18 RUE DE L'HOTEL DE VILLE - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4012	2 RUE TIRON - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4013	50 RUE VIEILLE DU TEMPLE - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4014	29 RUE DES BLANCS MANTEAUX - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4015	25 RUE DU PONT LOUIS PHILIPPE - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4016	3 RUE LOBAU - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4017	11 PLACE DE L'HOTEL DE VILLE - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4018	1 RUE SAINT BON - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4019	4 RUE DU CLOITRE SAINT MERRI - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4020	FACE 27 RUE QUINCAMPOIX - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4021	49 RUE RAMBUTEAU - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4101	11 RUE DE LA BASTILLE - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4103	1 RUE DES ARCHIVES - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4104	FACE 40 BOULEVARD SEBASTOPOL - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4105	17 BOULEVARD DU MORLAND - 75004 PARIS	<input type="checkbox"/>	<input checked="" type="checkbox"/>

## Station

- adresse: String
- arrondissement: String
- bonus: boolean
- numero: String
- ouvert: boolean

Station(String,String,boolean,boolean)  
getAdresse():String  
getArrondissement():String  
isBonus():boolean  
getNumero():String  
isOuvert():boolean  
toString():String

## Carte

Carte()  
ajouteStation(String,String,boolean,boolean):void  
chercher(String):Station  
getLaStation(int):Station  
nbStations():int

## FmStations

- adr: JLabel
- cbC: JLabel
- dispoC: JLabel
- libreC: JLabel
- ouvertC: JLabel
- res: JTable
- totC: JLabel
- maj: JLabel

FmStations()  
init():void  
actionPerformed(ActionEvent):void  
valueChanged(ListSelectionEvent):void  
changerLabels(boolean,HashMap<String,String>):void

-lesStations 0..\*

~mesStations  
0..\*

-laCarte 0..1

-maTable  
0..1

## MyTableModel

- nomColonnes: String[]

MyTableModel()  
getColumnCount():int  
getRowCount():int  
getColumnName(int):String  
getValueAt(int,int):Object  
getColumnClass(int):Class  
setLesStations(String):void

# Les classes métiers : Station

G Station	
•	adresse: String
•	arrondissement: String
•	bonus: boolean
•	numero: String
•	ouvert: boolean
•	Station(String,String,boolean,boolean)
•	getAdresse():String
•	getArrondissement():String
•	isBonus():boolean
•	getNumero():String
•	isOuvert():boolean
•	toString():String

# Les classes métiers : Station

- Créez la classe Station.
- Dans le constructeur, le *champ arrondissement (ou département)* sera obtenu à partir du *numero* ; en effet dans le *fichier xml (carto)* l'*attribut number* permet d'extraire l'*arrondissement (ou le département)*. Pour cela, observez bien la *construction du numéro de station* en vous connectant au site :  
***<http://www.velib.paris.fr/service/carto>***.








# Les classes métiers : Station

- Testez avec :

```
public class test {  
    public static void main(String[] args) {  
        Station s = new Station("20021", "15 rue petit", true, true);  
        Station s1 = new Station("31023", "11 rue Blanche", true,  
true);  
        Station s2 = new Station("8567", "45 rue Noire", true, true);  
  
        System.out.println(s.getArrondissement());  
        System.out.println(s1.getArrondissement());  
        System.out.println(s2.getArrondissement());  
    }  
}
```

- Vous devez obtenir : 20, 93 et 8.

# Les classes métiers : Carte

 Carte
 mesStations: ArrayList<Station>
 Carte()  ajouteStation(String,String,boolean,boolean):void  chercher(String):Station  getLaStation(int):Station  nbStations():int



# Les classes métiers : Carte

- Créez la classe Carte.
- Le constructeur est le suivant :












```
public Carte() {  
    mesStations = new ArrayList<Station>();  
}
```

# Les classes métiers : Carte

- Testez avec :

```
public class test {  
    public static void main(String[] args) {  
        Carte c = new Carte();  
        c.ajouteStation("20021", "15 rue petit", true, true);  
        c.ajouteStation("31023", "11 rue Blanche", true, true);  
        c.ajouteStation("8567", "45 rue Noire", true, true);  
  
        System.out.println(c.nbStations());  
        System.out.println(c.getLaStation(1));  
    }  
}
```

# La classe Passerelle

 <b>Passerelle</b>
 <u> urlCarto: String</u>
 <u> urlDispo: String</u>
 <u> Passerelle()</u>
 <u> getCarte(): Carte</u>
 <u> getDispo(String, String): HashMap&lt;String, String&gt;</u>

# La classe Passerelle

- Elle possède deux attributs *static* :

```
public class Passerelle {  
    private static String urlCarto =  
    "http://www.velib.paris.fr/service/carto";  
    private static String urlDispo =  
    "http://www.velib.paris.fr/service/stationdetails/";  
  
    public static Carte getCarte() {  
        ...  
    }  
}
```

- Créez la classe passerelle et sa méthode static getCarte() grâce aux méthodes de parsing de documents XML vues précédemment.

# La classe Passerelle

- Testez avec :

```
public class test {  
    public static void main(String[] args) {  
        Carte c = new Carte();  
        c = Passerelle.getCarte();  
  
        System.out.println(c.nbStations());  
        System.out.println(c.getLaStation(0));  
        System.out.println(c.chercher("4017"));  
    }  
}
```

# L'interface graphique : FrmStations

G FrmStations	
▪	adr: JLabel
▪	cbC: JLabel
▪	dispoC: JLabel
▪	libreC: JLabel
▪	ouvertC: JLabel
▪	maTable: MyTableModel
▪	res: JTable
▪	totC: JLabel
▪	maj: JLabel
●	FrmStations()
▪	init():void
●	actionPerformed(ActionEvent):void
●	valueChanged(ListSelectionEvent):void
▪	changerLabels(boolean,HashMap<String,String>):void

# L'interface graphique : FrmStations

- Vous pouvez maintenant créer la trame de l'interface graphique.
- Le constructeur sera :

```
public FrmStations() {  
    super("Disponibilité des stations de Vélib");  
    maTable = new MyTableModel();  
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    this.setSize(800, 600);  
    this.init();  
    this.setVisible(true);  
}
```

# L'interface graphique : FrmStations

- La liste des stations est affichée grâce à une JTable qui sera chargée avec MyTableModel.

```
res = new JTable(maTable);  
res.setPreferredSize(new Dimension(700,  
300));  
res.setFillsViewportHeight(true);  
res.setRowSelectionAllowed(true);  
res.setColumnSelectionAllowed(false);  
JScrollPane scrollPane = new JScrollPane(res);
```



# La classe MyTableModel

- Complétez la classe MyTableModel donnée en annexe.
- Elle permet de remplir la JTable avec les stations voulues.

# La classe MyTableModel

- Testez avec :

```
public class test {  
    public static void main(String[] args) {  
        Carte c = Passerelle.getCarte();  
        System.out.println(c.nbStations()+" stations  
chargées\n");  
  
        MyTableModel maTable = new MyTableModel();  
        System.out.println(maTable.getValueAt(1,0));  
        System.out.println(maTable.getValueAt(1,1));  
        System.out.println(maTable.getValueAt(1,2));  
        System.out.println(maTable.getValueAt(1,3));  
    }  
}
```

# L'interface graphique : FrmStations

- Puis testez votre fenêtre avec :

```
public class test {  
    public static void main(String[] args) {  
        Carte c = Passerelle.getCarte();  
        System.out.println(c.nbStations()+" stations  
chargées");  
        new FrmStations();  
    }  
}
```

- A ce stade, l'évènementiel n'est pas géré, nous allons donc nous en occuper.

# L'évènementiel de FrmStations

- Ecrivez le gestionnaire d'évènement suivant. Il devra être déclenché lors du choix d'un arrondissement ou département (choix d'un des JRadioButton).

```
public void actionPerformed(ActionEvent e) {  
    maTable.setLesStations(e.getActionCommand());  
    this.changerLabels(false, null);  
    res.revalidate();  
    res.clearSelection();  
    this.repaint();  
}
```

# L'évènementiel de FrmStations

- Ecrivez le gestionnaire d'évènement suivant.

```
public void valueChanged(ListSelectionEvent e) {  
    int numLigne = res.getSelectedRow();  
    if (numLigne != -1) {  
        String numStation = maTable.getValueAt(numLigne, 0).toString();  
        String adresse = maTable.getValueAt(numLigne, 1).toString();  
        HashMap<String, String> info = Passerelle.getDispo(adresse,  
numStation);  
        changerLabels(true, info);  
        this.repaint();  
    }  
}
```

- Il sera déclenché lors du choix d'une des stations dans la JTable grâce à :

```
29 res.getSelectionModel().addListSelectionListener(this);
```

# L'évènementiel de FrmStations

- Ecrivez la méthode `changerLabels` qui est utilisée dans les deux gestionnaires d'évènements précédents et qui permet de mettre à jour les labels d'informations d'une station.
- Attention, lorsqu'on clique sur un des `JRadioButton`, les labels redeviennent invisibles.

# La classe Passerelle

- Complétez la classe Passerelle en ajoutant la méthode static getDispo().
  - Pour cela, vous pouvez vous inspirer de la méthode getCarte() déjà écrite.
  - Elle permet de chercher les informations sur une station choisie dans la JTable.
- A partir d'un timestamp (String timeS) on peut créer une date et la formater avec :

```
SimpleDateFormat formater = new SimpleDateFormat("'le'  
dd/MM/yyyy 'à' HH:mm:ss");  
String date = formater.format(Long.parseLong(timeS)*1000);
```

# Les informations d'une station

- Par exemple, sur ce point d'accès :

```
- <station>
  <available>1</available>
  <free>26</free>
  <total>28</total>
  <ticket>1</ticket>
  <open>1</open>
  <updated>1416135251</updated>
  <connected>1</connected>
</station>
```

- 1 vélo est disponible
- 26 emplacements sont libres
- Ce point dispose de 28 emplacements (1 emplacement semble donc hors service).
- On peut payer par carte bleue :  
*ticket = 1*
- Ce point d'accès est ouvert :  
*open = 1*
- La dernière mise à jour a été faite au timestamp *1416135252*