

Mise en oeuvre d'Apache Wicket

par PanzerKunst

Date de publication : 09 juin 2008 (v1.7)

Dernière mise à jour :

Cet article explique l'architecture et le détail du code (Java et HTML) d'un petit site web basé sur le framework Apache Wicket.

I - Introduction.....	3
I-A - Pourquoi Wicket ?.....	3
I-B - Concepts Wicket mis en oeuvre dans cet article.....	3
I-C - Application exemple.....	3
II - Architecture de l'application Wicket.....	4
II-A - Architecture générale.....	4
II-B - Architecture de la couche web.....	4
II-C - Stockage des fichiers image du programme de chaque mois.....	5
III - Point d'entrée de notre application.....	6
III-A - Descripteur web.xml.....	6
III-B - Classe DahPubProgramApplication.java.....	6
III-C - .properties générales à l'application.....	7
IV - Page de visualisation du programme d'un mois (accueil).....	8
IV-A - Classe VisuProgramme.java.....	8
IV-A-1 - Accès à des images internes à la webapp.....	9
IV-A-2 - Accès à des ressources externes.....	9
IV-A-3 - Gestion des paramètres de requête.....	10
IV-A-4 - Messages de feedback Wicket.....	10
IV-B - Page HTML VisuProgramme.html.....	10
IV-C - .properties de la page VisuProgramme.....	10
V - Page de saisie du programme d'un mois.....	11
V-A - Classe SaisieProgramme.java.....	11
V-B - Les formulaires Wicket.....	12
V-C - Appel de services métier.....	13
V-D - Redirection vers une autre page.....	13
V-E - Contraintes de saisie et validation.....	13
V-F - Internationalisation (aka i18n).....	13
V-F-1 - Via les fichiers MyPage.properties.....	14
V-F-2 - Via les fichiers MyPage.html.....	14
V-G - Page HTML SaisieProgramme.html.....	15
VI - Page de liste des programmes saisis.....	16
VI-A - Classe ListeProgrammes.java.....	16
VI-B - Page HTML ListeProgrammes.html.....	17
VII - Panel de navigation.....	19
VII-A - Classe NavigationPanel.java.....	19
VII-B - Page HTML NavigationPanel.html.....	19
VII-B-1 - Liens hypertexte.....	19
VII-C - Insertion du composant dans les autres pages.....	19
VII-C-1 - Classe VisuProgramme.java.....	19
VII-C-2 - Page HTML VisuProgramme.html.....	20
VIII - Conclusion.....	22
IX - Code source de l'application.....	23
X - Remerciements.....	24

I - Introduction

I-A - Pourquoi Wicket ?

Wicket est YAJF (Yet Another JavaEE Framework), mais orienté composants. Il se distingue par certaines grandes particularités :

- Pas de JSP !
- Aucun tag de logique (boucle, condition) mélangé au HTML
- Les composants web sont créés dans des classes Java "à la Swing" avant d'être simplement placés aux endroits souhaités dans le fichier HTML
- Une gestion particulièrement simple de tâches récurrentes dans le développement de sites web comme la validation de formulaires, le passage de paramètres entre les pages et la navigation
- Un framework créé dans l'idée de vouloir faciliter le développement de composants réutilisables

Pour une liste exhaustive des principes ayant donné naissance à Wicket, l'idéal est de lire la page d'introduction au framework sur le site du projet : <http://wicket.apache.org>

I-B - Concepts Wicket mis en oeuvre dans cet article

Cet article sur Wicket 1.3 met en oeuvre les concepts suivants :

- Architecture d'une application Wicket
- Composants web Wicket : Label, Image, zone de feedback (erreur/warning/info), formulaire, liste déroulante, TextField, FileUploadField, tableau, bouton image, lien hypertexte
- Fonctionnement des fichiers .properties
- Accès aux images internes à la webapp
- Accès aux ressources externes à la webapp
- Gestion des paramètres de requête
- Contraintes de saisie, validation de ces contraintes lors de la soumission d'un formulaire
- Appels de services métier
- Redirection vers une autre page
- Internationalisation
- Composants web réutilisables

I-C - Application exemple

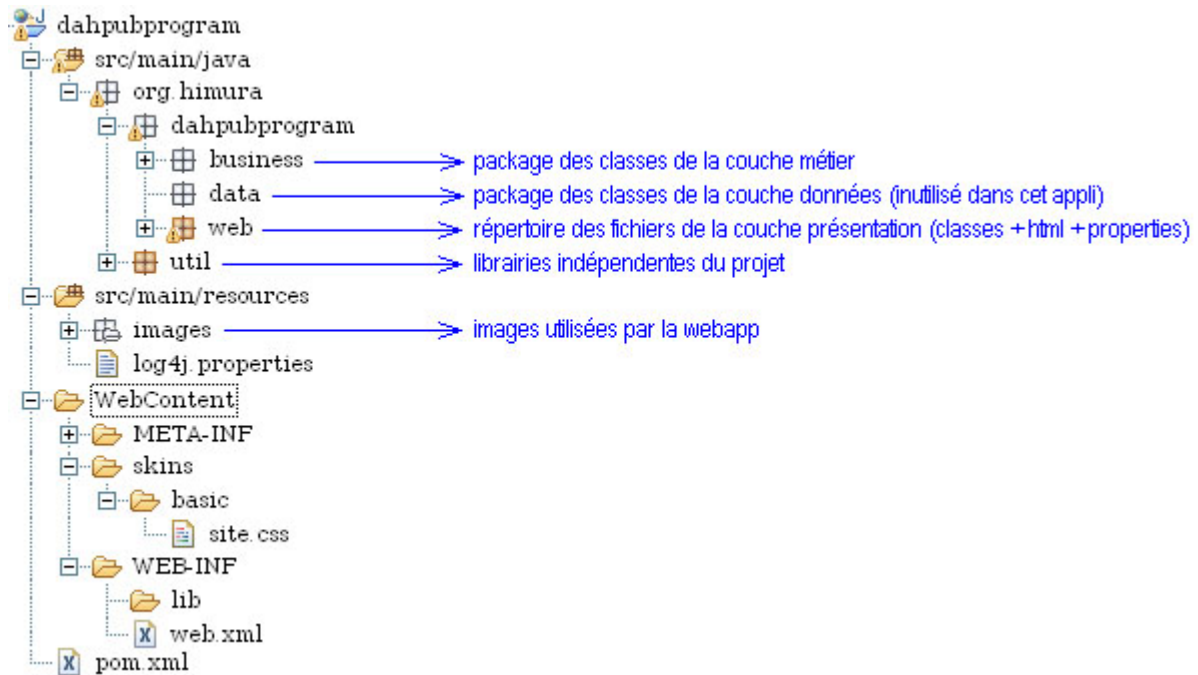
L'application décrite dans cet article se base sur l'activité d'un pub imaginaire : "Dah Pub". Ce pub organise plusieurs soirs par semaine des événements, listés sur un programme papier que le gérant imprime en de nombreux exemplaires. Il souhaite en plus diffuser le programme de chaque mois sur le site web du pub. La solution toute simple retenue est qu'il scanne le document puis upload le fichier image dans l'application.

II - Architecture de l'application Wicket

Notre application exemple DahPubProgram est une application Wicket 1.3. J'ai utilisé Java 1.5.0_15.

II-A - Architecture générale

L'architecture générale illustrée par le schéma ci-dessous résulte de la création d'un projet via le "Quickstart" du site web Wicket (<http://wicket.apache.org/quickstart.html>). Pour information, elle est basée sur Maven 2 :

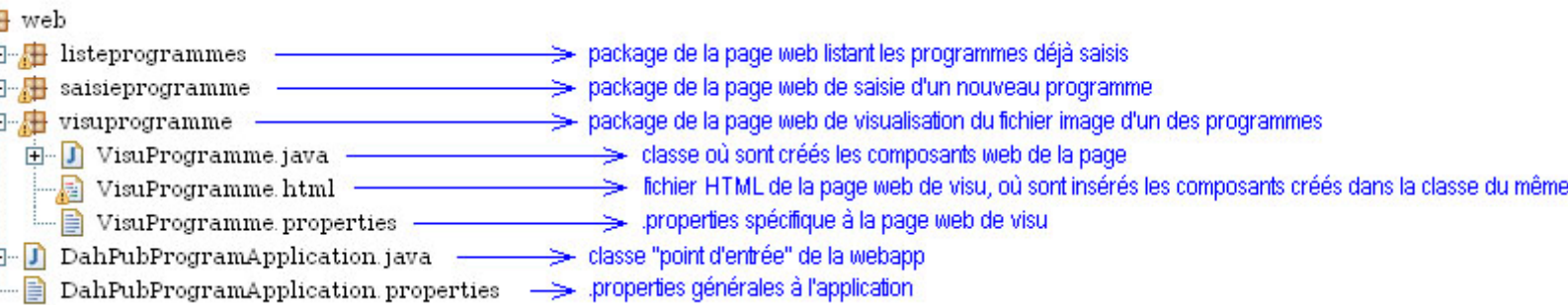


II-B - Architecture de la couche web

A la racine est créée une classe "MonApplication.java" qui hérite de WebApplication. C'est le point d'entrée de l'application web.

Ensuite, chaque page web sera caractérisée par son propre package (un choix parmi d'autres, ce n'est pas une restriction imposée par Wicket). Un tel package comprend :

- Une classe Java où sont créés les composants web de la page (composant = classe Wicket)
- Le (ou les si internationalisation) fichier HTML de la page, où seront insérés les composants créés dans la classe du même nom, via l'attribut *wicket:id="componentID"*
- Eventuellement un (ou plusieurs si internationalisation) fichiers .properties dédiés à cette page web



II-C - Stockage des fichiers image du programme de chaque mois

Les fichiers image seront stockés sous "C:\Repertoire_racine_de_stockage\AAAA\MM.ext" où "ext" est l'extension du fichier uploadé dans l'application. Le répertoire racine de stockage est paramétré dans le fichier DahPubProgramApplication.properties général à l'application.

III - Point d'entrée de notre application

III-A - Descripteur web.xml

Notre classe `DahPubProgramApplication.java` héritant de `WebApplication` est référencée dans le descripteur `web.xml` de la `webapp` :

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd"
  id="WebApp_ID" version="2.5">

  <display-name>dahpubprogram</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <filter>
    <filter-name>DahPubProgramApplication</filter-name>
    <filter-class>
      org.apache.wicket.protocol.http.WicketFilter
    </filter-class>
    <init-param>
      <param-name>applicationClassName</param-name>
      <param-value>org.himura.dahpubprogram.web.DahPubProgramApplication</param-value>
    </init-param>

    <!-- Uncomment to switch to production (deployment) mode
    <init-param>
      <param-name>configuration</param-name>
      <param-value>deployment</param-value>
    </init-param> -->
  </filter>

  <filter-mapping>
    <filter-name>DahPubProgramApplication</filter-name>
    <url-pattern>*</url-pattern>
  </filter-mapping>
</web-app>
```

III-B - Classe `DahPubProgramApplication.java`

La classe `DahPubProgramApplication.java` renvoie vers celle de la page d'accueil dans la méthode `getHomePage()`. Dans notre exemple, la page d'accueil sera la page d'affichage du mois en cours.

```
package org.himura.dahpubprogram.web;

public class DahPubProgramApplication extends WebApplication {

  public DahPubProgramApplication() {
    // Nothing
  }

  @Override
  public Class getHomePage() {
    return VisuProgramme.class;
  }
}
```

```
}  
}
```

III-C - .properties générales à l'application

```
rootProgramFilesStorageDir=c:/  
Repertoire_racine_où_seront_stockés_les_fichiers_image_du_programme_de_chaque_mois
```

Dans les classes de pages web Wicket (i.e. *extends WebPage*) et celles héritant de `org.apache.wicket.markup.html.panel.Panel`, les `.properties` peuvent être récupérées via la méthode `getString("clef")`. Wicket commencera par chercher dans les fichiers `.properties` du même répertoire que la page web, et s'il ne trouve pas ira chercher dans celui général à l'application.

IV - Page de visualisation du programme d'un mois (accueil)

Cette page web permet fonctionnellement de visualiser le fichier image d'un mois en particulier. Par défaut, c'est celui du mois en cours.

Elle met en oeuvre les concepts Wicket suivants :

- Composants web Wicket : Label, Image, zone de feedback (erreur/warning/info)
- Accès aux .properties spécifiques à la page web, et à celles générales à l'application
- Accès aux images internes à la webapp
- Accès aux ressources externes à la webapp
- Gestion des paramètres de requête
- Messages de feedback Wicket

IV-A - Classe VisuProgramme.java

Comprend 3 composants web Wicket :

- Un titre "Programme du mois MM / AAAA"
- L'image du programme du mois scannée puis uploadée dans l'application
- Un champ d'affichage des messages d'erreur/warning/info (dans le cas où il y aurait une exception, etc...)

```

package org.himura.dahpubprogram.web.visuprogramme;

public class VisuProgramme extends WebPage {
    public VisuProgramme() throws Exception {
        String yearParam = getRequest().getParameter("year");
        String monthStartingAtOneParam = getRequest().getParameter("month");
        if (yearParam != null && !" ".equals(yearParam) && monthStartingAtOneParam != null &&
            !" ".equals(monthStartingAtOneParam)) {
            int yearParamAsInt = Integer.parseInt(yearParam);
            int monthStartingAtOneParamAsInt = Integer.parseInt(monthStartingAtOneParam);

            init(yearParamAsInt, monthStartingAtOneParamAsInt);
        }
        else {
            init(DateUtils.currentYear(), DateUtils.currentMonth());
        }
    }

    private void init(int year, int monthStartingAtOne) throws Exception {
        // Title
        Label title = new Label("title", getString("title") + " " + monthStartingAtOne + " / " + year);
        add(title);

        /**
         * Month program image initialization
         */
        File programImageFilesDir = new File(getString("rootProgramFilesStorageDir") + "/" + year);
        File programImageFile = FileUtils.uniqueFileStartingWith(programImageFilesDir,
            String.valueOf(monthStartingAtOne));

        Image programImage;
        if (programImageFile != null) {
            programImage = new Image("programImage", new FileResource(programImageFile));
        }
        else {
            // Affichage d'un message INFO en bas de la page (feedbackPanel)
    
```

```

        info(getString("noProgramForThisMonth"));

        // Dirty trick to access a resource outside current package
        URL noProgramImageURL = getClass().getClassLoader().getResource("images/transparentPixel.png");
        File noProgramImageFile = new File(noProgramImageURL.toURI());

        programImage = new Image("programImage", new FileResource(noProgramImageFile));
    }
    add(programImage);
    /**
     * END - Month program image initialization
     */

    // Construct feedback panel
    add(new FeedbackPanel("feedback"));
}
    
```

On peut remarquer que si pour le mois courant aucun fichier n'a été uploadé (i.e. *programImageFile* == null), on est obligé de créer tout de même un composant image "vide" car sinon lors de la génération du HTML Wicket lèvera une exception disant que le tag `` n'a pas de composant associé.

IV-A-1 - Accès à des images internes à la webapp

Le composant image "vide" affiche en fait un pixel transparent. Ce fichier est "dahpubprogram\src\main\resources\images\transparentPixel.png". Le répertoire "src\main\resources" fait partie du classpath de l'application, donc le fichier image peut être accédé par le chemin relatif à ce répertoire : "images/transparentPixel.png".

Si le fichier image avait été dans le répertoire de la page web (ou un sous-répertoire), la création de l'objet Image aurait été plus simple :

```
Image myImage = new Image("myImage", new Model("subDir/myImage.jpg"));
```

IV-A-2 - Accès à des ressources externes

Les images de programme du mois peuvent être stockées n'importe où sur le serveur, par exemple dans un autre répertoire que la webapp.

L'image à afficher doit donc être encapsulée dans une classe de type org.apache.wicket.Resource.

Pour notre cas d'une ressource de type fichier, on peut créer une classe FileResource.java :

```

package org.himura.util.wicket;

public class FileResource extends WebResource {
    private File file;

    public FileResource(File file) {
        this.file = file;
    }

    @Override
    public IResourceStream getResourceStream() {
        return new FileResourceStream(file);
    }
}
    
```

IV-A-3 - Gestion des paramètres de requête

Dans Wicket, les paramètres de requête peuvent être récupérés dans une classe héritant de `WebPage` via la méthode `getRequest().getParameter("parameterID")`.

Dans notre application comme illustré dans la classe `VisuProgramme.java`, il est possible de choisir le programme qu'on souhaite visualiser en passant les bons paramètres dans l'URL. Par exemple celui du mois de mai 2008 peut être visualisé en entrant l'URL `http://<server_host>:<server_port>/dahpubprogram/?year=2008&month=5`.

IV-A-4 - Messages de feedback Wicket

Wicket met à disposition une classe permettant d'afficher aisément des messages d'information, d'avertissement et d'erreur dans les pages web : `org.apache.wicket.markup.html.panel.FeedbackPanel` (cf. `VisuProgramme.java`). On peut alors dans le code Java spécifier le type de message à y afficher (erreur, avertissement, info) et son contenu via les méthodes `error("message")`, `warn("message")` et `info("message")` respectivement.

IV-B - Page HTML VisuProgramme.html

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="skins/basic/site.css" />
  </head>

  <body>
    <h1><span wicket:id="title" id="title"></span></h1>

    <!-- Spacer -->
    <p></p>

    <img wicket:id="programImage" alt="programImage" />

    <div id="feedbackPanel">
      <span wicket:id="feedback"></span>
    </div>
  </body>
</html>
```

IV-C - .properties de la page VisuProgramme

Contient les propriétés spécifiques à la page `VisuProgramme`.

Ces propriétés peuvent être accédées dans la classe `VisuProgramme.java` via la méthode `getString("clef")` et dans la page HTML `VisuProgramme.html` par le tag Wicket `<wicket:message key="clef" />`.

Les composants ajoutés à une page web (comme un objet `Form` par exemple) peuvent accéder aux `.properties` de la page à laquelle ils sont attachés via la méthode `getPage().getString("clef")`.

Ici, ce sont les libellés par défaut utilisés dans la page. En effet, leur équivalent en français sera dans un autre fichier `.properties` utilisé pour l'internationalisation (cf. **§V-F**).

Contenu de `VisuProgramme.properties` :

```
title=Program for month
noProgramForThisMonth=No program for this month
```

V - Page de saisie du programme d'un mois

Cette page web permet fonctionnellement de saisir l'année, le mois et le fichier image du programme pour ce couple [mois, année]. Un bouton permet de valider les informations saisies et d'ajouter le fichier au "repository" des fichiers programme.

Elle met en oeuvre les concepts Wicket suivants (en plus de ceux déjà abordés) :

- Composants web Wicket : formulaire, liste déroulante, TextField, FileUploadField
- Contraintes de saisie, validation de ces contraintes lors de la soumission du formulaire
- Appel d'un service métier lors de la soumission du formulaire
- Redirection vers une autre page après l'appel du service métier
- Internationalisation

V-A - Classe SaisieProgramme.java

```

package org.himura.dahpubprogram.web.saisieprogramme;

public class SaisieProgramme extends WebPage {
    public SaisieProgramme() {
        // Construct the form
        SaisieProgrammeForm form = new SaisieProgrammeForm("form");
        add(form);
        form.init();

        // Construct feedback panel
        add(new FeedbackPanel("feedback"));
    }

    /**
     * THE form
     *
     */
    public class SaisieProgrammeForm extends Form {
        private DropDownChoice monthsC;

        // Year text field initialization
        final FormComponent yearTextField = new RequiredTextField("year",
            new Model(String.valueOf(DateUtils.currentYear())),
            Integer.class).add(NumberValidator.minimum(2000));

        // File upload field initialization
        final FileUploadField fileUploadField = new FileUploadField("programImageFile");

        public SaisieProgrammeForm(String id) {
            super(id);

            add(yearTextField);
            add(fileUploadField.setRequired(true));
        }

        public void init() {
            /**
             * Months combo-box initialization
             */
            // TreeMap and not HashMap because we want to keep insertion order
            final Map<Integer, String> mapMonthsKeyIDValueName = new TreeMap<Integer, String>();
            mapMonthsKeyIDValueName.put(new Integer(1), getPage().getString("jan"));
            mapMonthsKeyIDValueName.put(new Integer(2), getPage().getString("feb"));
            mapMonthsKeyIDValueName.put(new Integer(3), getPage().getString("mar"));
            mapMonthsKeyIDValueName.put(new Integer(4), getPage().getString("apr"));
        }
    }
}

```


V-C - Appel de services métier

Une fois ces informations récupérées, elles seront transmises à la couche métier pour être traitées. Dans notre exemple, une fois que l'année, le mois et le fichier sont récupérés du formulaire, ces informations seront transmises à la méthode `ProgramFileService.uploadProgramFile()` qui s'occupera de stocker l'image uploadée dans le bon répertoire et le bon nom ("C:\Repertoire_racine_de_stockage\AAAA\MM.ext").

V-D - Redirection vers une autre page

La redirection dans Wicket se fait via la méthode `setResponsePage(MyNextPage.class)`. La plupart du temps des paramètres doivent être passés à la page suivante. La manière classique de le faire est de les transmettre via un constructeur à arguments de `MyNextPage.java`. Dans notre exemple, on veut afficher le programme qui vient d'être saisi après la validation de la page. On va donc initialiser la classe `VisuProgramme.java` avec l'année et le mois saisis. Ensuite le constructeur `VisuProgramme(int year, int monthStartingAtOne)` se débrouille pour afficher le bon fichier image.

Remarque : le code de la classe `VisuProgramme.java` ci-dessus ne contient pas ce constructeur pour rester simple dans un premier temps, cf. §VII-C-1 pour le code complet.

V-E - Contraintes de saisie et validation

Wicket a son propre système de contraintes de saisie et de leur validation. Les contraintes sont créées en même temps que les composants.

Contraintes utilisées dans l'application d'exemple :

- `new RequiredTextField()` -> Champ requis
- `new Model("componentID", Integer.class)` -> Doit être un entier
- `myComponent.add(NumberValidator.minimum(2000))` -> Doit être supérieur ou égal à 2000
- `setRequired(true)` -> Champ requis

Sur validation du formulaire, par défaut Wicket va opérer à la validation selon ces contraintes. Si une des contraintes n'est pas respectée et qu'un composant `FeedbackPanel` est présent dans la page, il sera automatiquement rempli avec le message d'erreur par défaut de Wicket, dans la langue de la **locale** par défaut du navigateur.

Il est possible de créer ses propres messages d'erreur de validation si ceux par défaut de Wicket ne sont pas satisfaisants.

Dans certains cas, il est nécessaire de désactiver la validation du formulaire, par exemple sur un bouton d'annulation ou de réinitialisation. L'exemple <http://wicketstuff.org/wicket13/forminput> illustre bien cette désactivation via la méthode `setDefaultFormProcessing(false)` :

```
add(new Button("resetButton") {
    public void onSubmit() {
        setResponsePage(MyPage.class);
    }
}).setDefaultFormProcessing(false);
```

V-F - Internationalisation (aka i18n)

L'internationalisation d'un site web avec Wicket est un jeu d'enfant. Il y a 2 méthodes :

V-F-1 - Via les fichiers MyPage.properties

Dans le cas où une page web ne contient pas trop de texte, il est préférable d'externaliser tous les libellés traduisibles dans le fichier .properties correspondant. Le fichier MyPage.properties contiendra les libellés dans la langue par défaut (l'anglais admettons). Pour avoir sa page en français il suffit alors d'initialiser un fichier MyPage_fr.properties avec les mêmes clés.

Exemple dans notre application :

SaisieProgramme.properties

```
title=Input a program

month=Month
file=File

jan=Jan
feb=Feb
mar=Mar
apr=Apr
may=May
jun=Jun
jul=Jul
aug=Aug
sep=Sep
oct=Oct
nov=Nov
dec=Dec
```

SaisieProgramme_fr.properties

```
title=Saisie d'un programme

month=Mois
file=Fichier

jan=Janv
feb=Fevr
mar=Mars
apr=Avr
may=Mai
jun=Juin
jul=Juil
aug=Août
sep=Sept
oct=Oct
nov=Nov
dec=Déc
```

V-F-2 - Via les fichiers MyPage.html

Si la page web contient beaucoup de texte, externaliser le texte dans des .properties devient vite très lourd. La solution est de créer le fichier MyPage.html pour la langue par défaut et MyPage_fr.html pour l'équivalent en français par exemple.

V-G - Page HTML SaisieProgramme.html

Le code de la page SaisieProgramme.html pour pouvoir visualiser comment les différents composants sont insérés dans le HTML :

```

<html>
  <head>
    <link rel="stylesheet" type="text/css" href="skins/basic/site.css" />
  </head>

  <body>
    <h1>
      <wicket:message key="title" />
    </h1>

    <!-- Spacer -->
    <p></p>

    <form wicket:id="form">
      <div>
        <table>
          <tr>
            <td>
              <wicket:message key="month" />
            </td>
            <td>
              <select wicket:id="months" name="months"></select>
              <input wicket:id="year" type="text" size="3" maxlength="4" />
            </td>
          </tr>
          <tr>
            <td>
              <wicket:message key="file" />
            </td>
            <td>
              <input wicket:id="programImageFile" type="file" id="programImageFile" />
            </td>
          </tr>

          <!-- Spacer -->
          <tr>
            <td>&nbsp;</td>
          </tr>

          <tr>
            <!-- Spacer -->
            <td>&nbsp;</td>

            <td>
              <input type="submit" value="Upload" />
            </td>
          </tr>
        </table>
      </div>
    </form>

    <div id="feedbackPanel">
      <span wicket:id="feedback"></span>
    </div>
  </body>
</html>

```

VI - Page de liste des programmes saisis

Cette page présente un tableau de tous les programmes saisis. Chaque ligne du tableau indique :

- L'année du programme
- Le mois du programme
- Un bouton image permettant de visualiser le programme (i.e. afficher la page VisuProgramme pour ce programme et non celui du mois en cours)
- Un bouton image permettant de supprimer le programme (i.e. supprimer le fichier image du repository)

Elle met en oeuvre les concepts Wicket suivants (en plus de ceux déjà abordés) :

- Composants web Wicket : tableau, boutons image
- Forcer le rafraîchissement d'une page

VI-A - Classe ListeProgrammes.java

L'API Wicket offre plusieurs classes d'implémentation de tableaux, illustrées dans l'exemple <http://wicketstuff.org/wicket13/repeater>. J'ai utilisé une des plus simples : `org.apache.wicket.markup.repeater.data.DataView`, construite à partir d'une liste de programmes (`java.util.List<Program>`).

```
package org.himura.dahpubprogram.web.listeprogrammes;

import java.util.List;

public class ListeProgrammes extends WebPage {
    public ListeProgrammes() throws Exception {
        // Construct the form panel
        ListeProgrammesForm form = new ListeProgrammesForm("form");
        add(form);
        form.init();

        // Construct feedback panel
        add(new FeedbackPanel("feedback"));
    }

    /**
     * THE form
     */
    public class ListeProgrammesForm extends Form {
        public ListeProgrammesForm(String id) {
            super(id);
        }

        public void init() throws Exception {
            List<Program> listPrograms =
                ListProgramsService.listPrograms(getPage().getString("rootProgramFilesStorageDir"));

            DataView dataViewPrograms = new DataView("dataViewPrograms", new
                ListDataProvider(listPrograms)) {
                @Override
                protected void populateItem(final Item item) {
                    try {
                        final Program program = (Program) item.getModelObject();
                        item.add(new Label("year", String.valueOf(program.getYear())));
                        item.add(new Label("monthStartingAtOne",
                            String.valueOf(program.getMonthStartingAtOne())));
                    }
                }
            };
        }
    }
}
```



```
</head>

<body>
  <h1><wicket:message key="title" /></h1>

  <!-- Spacer -->
  <p></p>

  <form wicket:id="form">
    <table cellpadding="0" cellspacing="0" class="dataview">
      <tr>
        <th><wicket:message key="year" /></th>
        <th><wicket:message key="month" /></th>
        <th>&nbsp;</th>
        <th>&nbsp;</th>
      </tr>
      <tr wicket:id="dataViewPrograms">
        <td><span wicket:id="year"></span></td>
        <td><span wicket:id="monthStartingAtOne"></span> </td>
        <td><input wicket:id="viewButton" type="image" /></td>
        <td><input wicket:id="deleteButton" type="image" /></td>
      </tr>
    </table>
  </form>

  <div id="feedbackPanel">
    <span wicket:id="feedback"></span>
  </div>
</body>
</html>
```

VII - Panel de navigation

Nous allons créer un panel de navigation commun aux 3 pages du site permettant de naviguer de l'une à l'autre. Il sera inséré dans chacune, permettant ainsi de factoriser le code HTML.

Il met en oeuvre les concepts Wicket suivants (en plus de ceux déjà abordés) :

- Composant web réutilisable dans plusieurs pages web
- Liens hypertexte

VII-A - Classe NavigationPanel.java

```
package org.himura.dahpubprogram.web.navigationpanel;

public class NavigationPanel extends Panel {
    public NavigationPanel(String id) {
        super(id);

        add(new BookmarkablePageLink("linkVisuProgramme", VisuProgramme.class));
        add(new BookmarkablePageLink("linkListeProgrammes", ListeProgrammes.class));
        add(new BookmarkablePageLink("linkSaisieProgramme", SaisieProgramme.class));
    }
}
```

VII-B - Page HTML NavigationPanel.html

```
<wicket:panel>
  <a wicket:id="linkVisuProgramme"><wicket:message key="viewMonthProgram" /></a>
  <br />
  <a wicket:id="linkListeProgrammes"><wicket:message key="allPrograms" /></a>
  <br />
  <a wicket:id="linkSaisieProgramme"><wicket:message key="addOrModifyProgram" /></a>
</wicket:panel>
```

VII-B-1 - Liens hypertexte

Wicket permet d'insérer des liens hypertexte de plusieurs manières. L'exemple ci-dessus montre comment le faire via l'objet BookmarkablePageLink.

Il est aussi possible de créer des liens hypertexte classiques (tag `myLinkLabel`). Dans ce cas ils doivent être entourés de tags `<wicket:link>` pour être interprétés correctement par Wicket.

Un tel lien vers la page VisuProgramme aurait été codé par : `<wicket:message key="viewMonthProgram" />`.

VII-C - Insertion du composant dans les autres pages

Dans la page web VisuProgramme par exemple :

VII-C-1 - Classe VisuProgramme.java

```
package org.himura.dahpubprogram.web.visuprogramme;
```

```

public class VisuProgramme extends WebPage {
    public VisuProgramme() throws Exception {
        String yearParam = getRequest().getParameter("year");
        String monthStartingAtOneParam = getRequest().getParameter("month");
        if (yearParam != null && !" ".equals(yearParam) &&
            monthStartingAtOneParam != null && !" ".equals(monthStartingAtOneParam)) {
            int yearParamAsInt = Integer.parseInt(yearParam);
            int monthStartingAtOneParamAsInt = Integer.parseInt(monthStartingAtOneParam);

            init(yearParamAsInt, monthStartingAtOneParamAsInt);
        }
        else {
            init(DateUtils.currentYear(), DateUtils.currentMonth());
        }
    }

    public VisuProgramme(int year, int monthStartingAtOne) throws Exception {
        init(year, monthStartingAtOne);
    }

    private void init(int year, int monthStartingAtOne) throws Exception {
        // Navigation panel
        Panel navigationPanel = new NavigationPanel("navigationPanel");
        add(navigationPanel);

        // Title
        Label title = new Label("title", getString("title") + " " + monthStartingAtOne + " / " + year);
        add(title);

        /**
         * Month program image initialization
         */
        File programImageFilesDir = new File(getString("rootProgramFilesStorageDir") + "/" + year);
        File programImageFile = FileUtils.uniqueFileStartingWith(programImageFilesDir,
            String.valueOf(monthStartingAtOne));

        Image programImage;
        if (programImageFile != null) {
            programImage = new Image("programImage", new FileResource(programImageFile));
        }
        else {
            // Affichage d'un message INFO en bas de la page (feedbackPanel)
            info(getString("noProgramForThisMonth"));

            // Dirty trick to access a resource outside current package
            URL noProgramImageURL = getClass().getClassLoader().getResource("images/
transparentPixel.png");
            File noProgramImageFile = new File(noProgramImageURL.toURI());

            programImage = new Image("programImage", new FileResource(noProgramImageFile));
        }
        add(programImage);
        /**
         * END - Month program image initialization
         */

        // Construct feedback panel
        add(new FeedbackPanel("feedback"));
    }
}

```

VII-C-2 - Page HTML VisuProgramme.html

```

<html>
<head>

```

```
<link rel="stylesheet" type="text/css" href="skins/basic/site.css" />
</head>
<body>
  <table>
    <tr>
      <td><div wicket:id="navigationPanel"></div></td>

      <td>
        <h1><span wicket:id="title" id="title"></span></h1>

        <!-- Spacer -->
        <p></p>

        <img wicket:id="programImage" alt="programImage" />

        <div id="feedbackPanel">
          <span wicket:id="feedback"></span>
        </div>
      </td>
    </tr>
  </table>
</body>
</html>
```

VIII - Conclusion

Je commencerai par une mise en garde : la documentation du framework Wicket est aussi faible que son API est grande. La plus précieuse source d'information est la mailing-list, qui n'est pas forcément mise en avant. Elle est archivée sur le site <http://www.nabble.com/Wicket---User-f25133.html> qui offre une fonctionnalité de recherche tout simplement indispensable. De nombreux exemples de mise en oeuvre sont par ailleurs disponibles sur les sites <http://wicketstuff.org> et <http://wicket.sourceforge.net/wicket-extensions>, avec leur code source.

Par ailleurs, Wicket est clairement un framework à haut niveau d'abstraction, il faudrait vérifier qu'il est possible de créer des sites web complexes sans être forcé de sortir du framework et coder certaines parties avec du code plus bas niveau.

IX - Code source de l'application

 [*Mise_en_oeuvre_d'Apache_Wicket_Src_Eclipse_v1.7.zip*](#)

X - Remerciements

Merci à **giffftane**, **djo.mos** et **Baptiste Wicht** pour leurs remarques constructives sur cet article, ainsi qu'à **RideKick** pour sa relecture.