

M2 IAGL - Vieille technologique

Guestbook



# Google App Engine

Le 01/12/2014

Omar Chahbouni

Maxime Chaste

Pierre-Philippe Berenguer

## Table des matières

Guestbook.....	1
<b>Objectifs.....</b>	<b>3</b>
<b>Prérequis.....</b>	<b>3</b>
Installation de l'IDE sous XP.....	3
Ajout des plugins Google.....	3
<b>Exercice 1 : Déployer sur l'App Engine.....</b>	<b>5</b>
Etape 1 : Déclarer l'application auprès de Google.....	5
Etape 2 : création du projet sous Eclipse.....	5
Etape 3 : Lancer l'application.....	7
<b>Exercice 2 : Interaction avec votre compte Google.....</b>	<b>8</b>
Création d'une JSP Guestbook.....	8
Alternative à la JSP créer une servlet.....	10
<b>Exercice 3 : Ajout de données sur le Datastore.....</b>	<b>11</b>
Etape 1 : Créer une servlet SignGuestbookServlet.....	11
Etape 2 : Ajout d'image sur le Blobstore.....	14
<b>Exercice 4 : Récupérer les images.....</b>	<b>16</b>
Etape 1 : mise à jour de la jsp.....	16
Etape 2 : Redimensionnez l'image au format 200 * 200.....	17
Etape 3 : vérifier le format de l'image.....	17
<b>Exercice 5 : Associer l'utilisateur à des données.....</b>	<b>19</b>
<b>Aller plus loin.....</b>	<b>22</b>

# Guestbook

## Objectifs

Créer un service JEE pour saisir des messages dans un Guestbook en ligne, manipuler des données sur le Datastore, et déployer l'application sur les serveurs Google.

Attention, il y aura beaucoup de listes à puce dans ce TP.

## Prérequis

- Avoir un compte Google
- Machines de la fac : booter sur Windows XP
- Machines perso (Windows, OSX, Ubuntu...) : passez directement à l'étape « *Ajout des plugins Google* »

## Installation de l'IDE sous XP

Télécharger Eclipse IDE for Java EE developers, en 32 bits :

<https://eclipse.org/downloads/>

Lancer Eclipse.

Remplacer la JRE par le JDK Java 7 (*des screens sont disponibles sur la page suivante*) :

Windows → préférences → Java → Installed JREs → Add JRE

Choisir : Standard VM

Pour « JRE Home » mettre le chemin : C:\Program Files\Java\jdk1.7.0\_25

Cliquez sur « Finish », puis cochez ce nouveau JDK

Modifier le fichier de configuration :

Allez dans le répertoire d'Eclipse et ouvrez le fichier de configuration : *eclipse.ini*

Juste avant la ligne *-vmargs*, insérez sur deux lignes les valeurs suivantes :

```
-vm  
C:\Program Files\Java\jdk1.7.0_25\bin
```

## Ajout des plugins Google

Dans Eclipse.

Allez sur : Help → Install new software → Add :

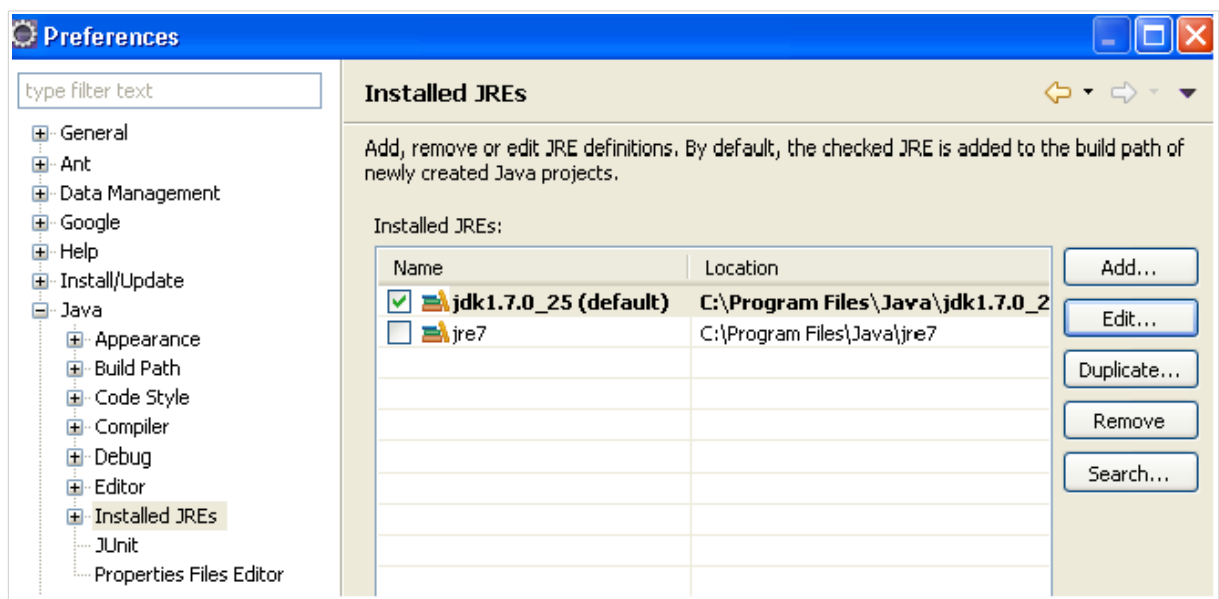
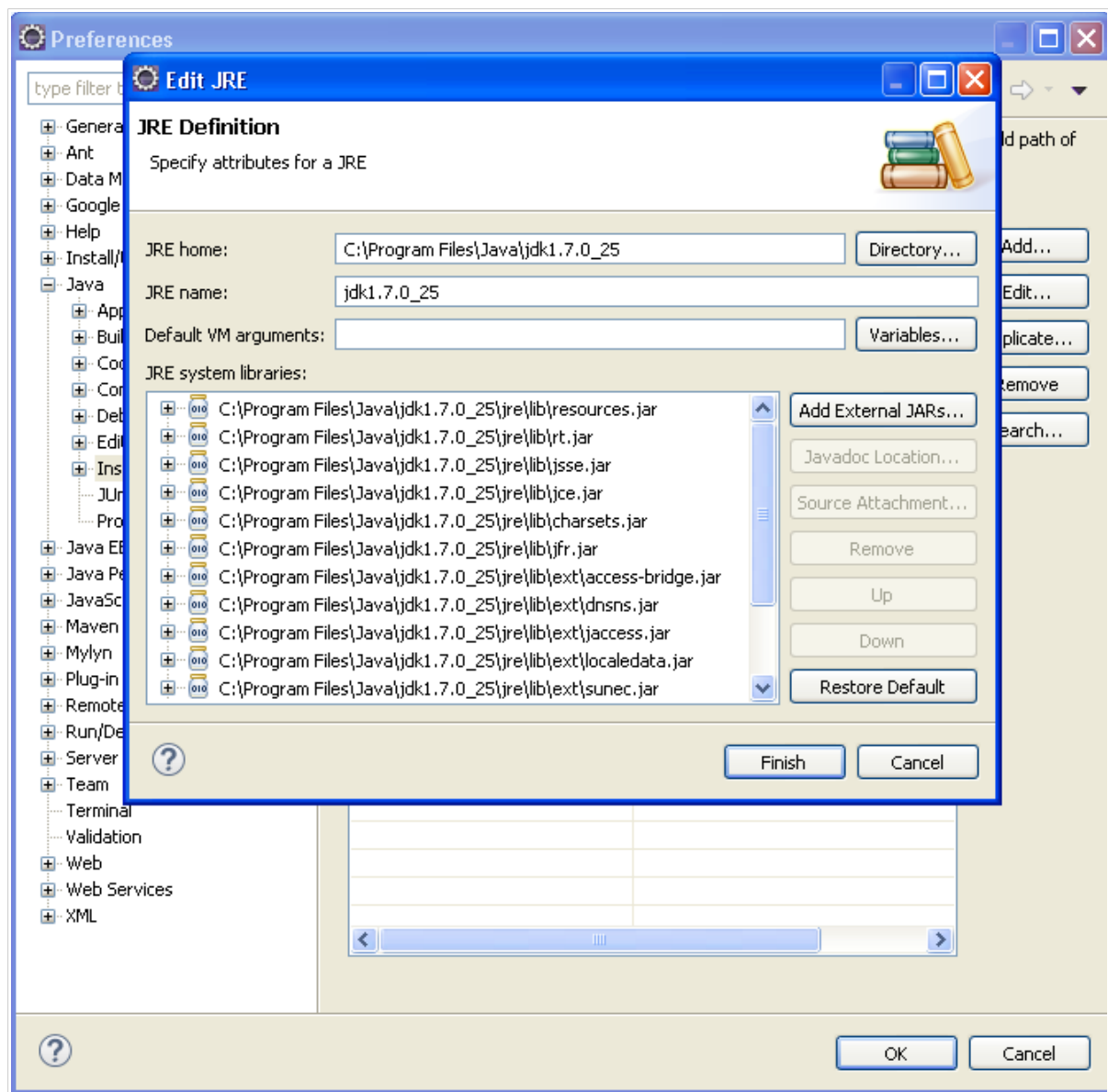
Insérez le lien suivant <https://dl.google.com/eclipse/plugin/4.4>

Remplacez « 4.4 » par la version de votre Eclipse

Choisissez les plugins :

Google Plugin for Eclipse  
et SDKs

**Fin de l'installation.**



# Exercice 1 : Déployer sur l'App Engine

Une JSP permettra de faire un premier « hello world », et de s'authentifier dans l'application grâce à votre compte Google.

Pour pouvoir exécuter cette application, il faut se rendre au préalable dans le tableau de bord en ligne de Google, et initialiser l'application.

## Etape 1 : Déclarer l'application auprès de Google

Allez sur <https://appengine.google.com/>

Une fois authentifié, cliquez sur : *Create new project (ou Create Application)*.

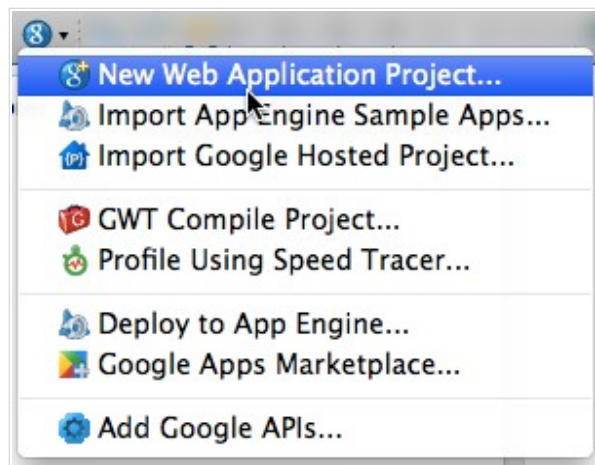
Renseignez le nom de votre première application, par exemple « *tpgae* ».

Attention, reprenez le **Application Identifier**, il permettra d'accéder à votre application :  
<http://<id-application>.appspot.com/guestbook/>

## Etape 2 : création du projet sous Eclipse

Pour lancer l'application, le plugin de Google ajoute un bouton *g*.

- Cliquez sur “new Web Application Project”. Comme sur la figure suivante :



Donnez un nom à votre premier projet, pour « Project name » entrez le nom saisi en ligne lors de l'étape 1 (« *tpgae* »). Pour « Package » mettez ce que vous voulez, par exemple « guestbook ».

- Vérifiez que les cases "Use Google App Engine" et "Generate project sample code" sont cochées. Décochez "Use Google Web Toolkit" si vous l'avez installé.
- Enfin, ajoutez le nom de votre projet dans le fichier *appengine-web.xml* entre les balises `<application>tpgae</application>`

**New Web Application Project**

Create a Web Application project in the workspace or in an external location

Project name: MaPremiereApp

Package: (e.g. com.example.myproject) maPremiereApp

Location

☒ Create new project in workspace

☐ Create new project in:

Directory: /Users/mateo21/dev/workspaces/MaPremiereApp [Browse...](#)

Google SDKs

☐ Use Google Web Toolkit

☒ Use default SDK (GWT - 2.5.1) [Configure SDKs...](#)

☐ Use specific SDK: GWT - 2.5.1

☒ Use Google App Engine

☒ Use default SDK (App Engine - 1.7.6) [Configure SDKs...](#)

☐ Use specific SDK: App Engine - 1.7.6

The project will use App Engine's [High Replication Datastore \(HRD\)](#) by default.

Google Apps Marketplace

☐ Add support for listing on Google Apps Marketplace

Sample Code

☒ Generate project sample code

[?](#) [Cancel](#) [Finish](#)

L'application devrait ressembler à ceci :

```
Guestbook/  
  src/  
    guestbook/  
      server/  
        GuestbookServlet.java  
    META-INF/  
      jdoconfig.xml  
      log4j.properties  
      logging.properties  
  war/  
    WEB-INF/  
      lib/  
        ...App Engine JARs...  
      appengine-web.xml  
      web.xml  
    index.html
```

## Etape 3 : Lancer l'application

Il est maintenant possible de déployer l'application avec le plugin Google.

- clic droit sur le projet → **Google** → **deploy to App Engine**



**Hello App Engine!**

Available Servlets:

[TPGAE](#)

Si la page ne s'affiche pas, l'application est accessible en remplaçant l'id de votre application à l'adresse :

<http://<id-application>.appspot.com/guestbook/>

**Une console** est disponible en ligne pour les développeurs. Elle permet entre autre d'administrer l'application, surveiller les pics d'activité, les données. N'hésitez pas à jeter un œil.

<https://console.developers.google.com/project>

Il est également possible de **lancer l'application en locale** sur sa propre machine. Les plugins permettent de simuler l'environnement d'exécution, en cliquant simplement sur « Run » via Eclipse. Par défaut l'application est accessible : <http://localhost:8888/>

## Exercice 2 : Interaction avec votre compte Google

### Création d'une JSP Guestbook

En lançant l'application, nous souhaitons remplacer le message d'accueil, et **pouvoir se connecter à Google**. En insérant un objet de type **UserService** dans une JSP, l'utilisateur pourra se loguer. Il suffit d'insérer le service dans le HTML de la JSP.

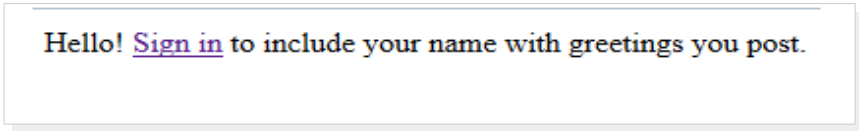
Si vous êtes déjà connecté, vous aurez un message « Sign out ».

En JEE les modules sont groupées dans des paquets. Par exemple :

- JAR : EJB class files + deployment descriptor
- WAR : servlet class files + JSP + fichiers statiques HTML, CSS, images... etc

La JSP devra donc se trouver dans war/

Voici sur que nous voulons obtenir sur la page d'accueil :



Hello! [Sign in](#) to include your name with greetings you post.

**A vous de jouer :**

- Créez la JSP dans le répertoire war :  
clic droit sur war, new → web → JSP File
- Indiquez la JSP comme page d'accueil dans le fichier web.xml :

```
<welcome-file-list>
  <welcome-file>Guestbook.jsp</welcome-file>
</welcome-file-list>
```



- Ensuite, faites appel à `UserService.getCurrentUser()`, et insérez un « Hello » dans la JSP. Voici le code correspondant :

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ page import="com.google.appengine.api.users.User" %>
<%@ page import="com.google.appengine.api.users.UserService" %>
<%@ page import="com.google.appengine.api.users.UserServiceFactory" %>
<%@ page import="java.util.List" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
<html>
<head>
</head>

<body>

<%
UserService userService = UserServiceFactory.getUserService();
User user = userService.getCurrentUser();
if (user != null) {
    pageContext.setAttribute("user", user);
}

%>
<p>Hello, ${fn:escapeXml(user.nickname)} (You can
    <a href="<%= userService.createLogoutURL(request.getRequestURI()) %>">sign out</a>
</p>
<%
} else {
%>
<p>Hello!
    <a href="<%= userService.createLoginURL(request.getRequestURI()) %>">Sign in</a>
    to include your name with greetings you post.
</p>
<%
}
%>
```

## Rappel : les formulaires en JEE

Ajoutez un formulaire pour l'envoi d'un message :

```
<form action="/Guestbook.jsp" method="get">
    <div><input type="text" name="guestbookName" value="&${fn:escapeXml(guestbookName)}" /></div>
    <div><input type="submit" value="Ok" /></div>
</form>
```

Pour récupérer les données il suffit d'utiliser `request.getParameter`. Faites précéder `UserService` par :

```
String guestbookName = request.getParameter("guestbookName");
if (guestbookName == null) {
    guestbookName = "default";
}
pageContext.setAttribute("guestbookName", guestbookName);
```

## Alternative à la JSP créer une servlet

Cette étape est simplement un rappel sur l'utilisation de servlet, elle ne sert pas à proprement parlé dans le guestbook. Passez cette étape si vous êtes à l'aise avec les servlets.

### A vous de jouer :

Essayez de créer une servlet pour afficher un « Hello » et le nom de l'utilisateur connecté à Google. Il faut également ajouter la servlet dans le web.xml.

### Solution :

Créez la servlet :

clic droit → new class. Comme toute servlet elle doit étendre HttpServlet.

```
import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;

import java.io.IOException;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class GuestbookServlet extends HttpServlet {
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Override
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws IOException {
        UserService userService = UserServiceFactory.getUserService();
        User currentUser = userService.getCurrentUser();

        if (currentUser != null) {
            resp.setContentType("text/plain");
            resp.getWriter().println("Hello, " + currentUser.getNickname());
        } else {
            resp.sendRedirect(userService.createLoginURL(req.getRequestURI()));
        }
    }
}
```

### Mapping :

```
<servlet>
    <servlet-name>guestBook</servlet-name>
    <servlet-class>VotrePackage.GuestbookServlet</servlet-class>
</servlet>

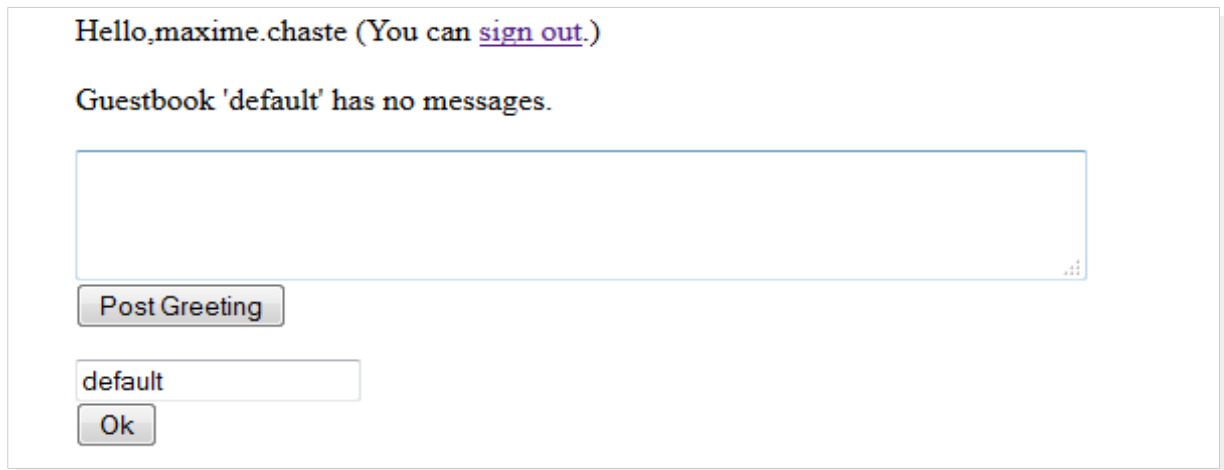
<servlet-mapping>
    <servlet-name>guestBook</servlet-name>
    <url-pattern>/guest</url-pattern>
</servlet-mapping>
```

## Exercice 3 : Ajout de données sur le Datastore

Cette étape permet d'interagir avec des données envoyées sur le Datastore Google.

### Etape 1 : Créer une servlet *SignGuestbookServlet*

Nous voulons ajouter un champ pour saisir un message. Une fois envoyé, le message apparaîtra au dessus du formulaire. De façon transparente, le message sera envoyé sur le Datastore.



The screenshot shows a web application interface. At the top, it says "Hello,maxime.chaste (You can [sign out.](#))". Below this, it says "Guestbook 'default' has no messages." There is a large text input field. Below the input field is a button labeled "Post Greeting". Below the button is a dropdown menu showing "default". At the bottom is an "Ok" button.

Créez une servlet *SignGuestbookServlet*. Ajoutez y la méthode suivante :

```
@Override
public void doPost(HttpServletRequest req, HttpServletResponse resp)
    throws IOException {
    UserService userService = UserServiceFactory.getUserService();
    User user = userService.getCurrentUser();

    String guestbookName = req.getParameter("guestbookName");
    String content = req.getParameter("content");
    Date date = new Date();
    // Creation de l'entite
    Entity greeting = new Entity("Greeting");
    greeting.setProperty("user", user);
    greeting.setProperty("date", date);
    greeting.setProperty("content", content);

    DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();
    datastore.put(greeting);

    resp.sendRedirect("/Guestbook.jsp?guestbookName=" + guestbookName);
}
```

### Modification de la JSP

Nous allons récupérer les messages dans la JSP. Ajoutez les imports pour le datastore :

```
<%@ page import="com.google.appengine.api.datastore.DatastoreService" %>
<%@ page import="com.google.appengine.api.datastore.DatastoreServiceFactory" %>
<%@ page import="com.google.appengine.api.datastore.Entity" %>
<%@ page import="com.google.appengine.api.datastore.FetchOptions" %>
<%@ page import="com.google.appengine.api.datastore.Query" %>
```

Juste avant le `<form action="/Guestbook.jsp" method="get">`, ajoutez :

```
<%
    DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();//use the DataStore
    Query query = new Query("Greeting").addSort("date", Query.SortDirection.DESENDING);//query in
    DataStore
    List<Entity> greetings =
    datastore.prepare(query).asList(FetchOptions.Builder.withLimit(5));//get only 5 item
    if (greetings.isEmpty()) {
        %>
        <p>Guestbook '${fn:escapeXml(guestbookName)}' has no messages.</p>
        %>
    } else {
        %>
        <p>Messages in Guestbook '${fn:escapeXml(guestbookName)}'.</p>
        %>
        for (Entity greeting : greetings) {
            pageContext.setAttribute("greeting_content",
                greeting.getProperty("content"));

                //A COMPLETER

        }
    }
    %>

    <form action="/sign" method="post">
        <div><textarea name="content" rows="3" cols="60"></textarea></div>
        <div><input type="submit" value="Post Greeting"/></div>
        <input type="hidden" name="guestbookName" value='${fn:escapeXml(guestbookName)}' />
    </form>
```

## À vous de jouer

Complétez le **code précédent** pour obtenir pour chaque élément greeting sous la forme suivante :

`<nickname> wrote <message>` ou « *An anonymous person wrote :<message>* »

Remarque : le bouton « ok » : permet uniquement de renommer votre guestbook puisque celui-ci ne crée aucune *Entity*.

## Aide :

1. `${fn:escapeXml(greeting_user.nickname)}` contient le message
2. `greeting.getProperty("user")` contient l'utilisateur

## Solution

```
if (greeting.getProperty("user") == null) {  
    <p>An anonymous person wrote:</p>  
} else {  
    pageContext.setAttribute("greeting_user",  
        greeting.getProperty("user"));  
    <p><b>${fn:escapeXml(greeting_user.nickname)}</b> wrote:</p>  
}  
    <blockquote>${fn:escapeXml(greeting_content)}</blockquote>
```

- Testez l'ajout d'une valeur lorsque vous envoyez le message. Ensuite, jetez un œil sur le contenu du Datastore en ligne :

<https://appengine.google.com/>, Data → Datastore Viewer

## Etape 2 : Ajout d'image sur le Blobstore

Le Blobstore est un système de stockage intégré à votre application. De la même façon que le Datastore stocke des données, le Blobstore stocke des fichiers volumineux.

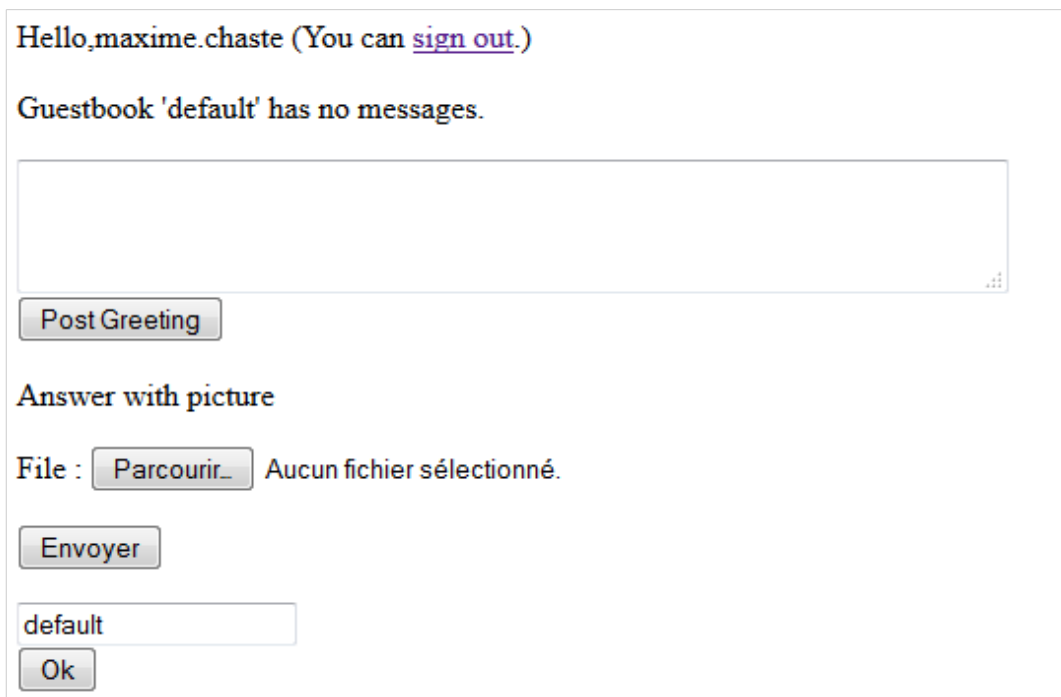
- Ajoutez l'import suivant dans la JSP :

```
<%@ page import="com.google.appengine.api.blobstore.*" %>
```

- Ajoutez le code suivant entre les deux formulaires créés précédemment :

```
<label> Answer with picture </label>
<%
    BlobstoreService blobstoreService = BlobstoreServiceFactory.getBlobstoreService();
%>
    <form action="<%= blobstoreService.createUploadUrl("/upload") %>" method="post"
    enctype="multipart/form-data">
        <p>
            <label>File: <input type="file" name="uploadedFile" /></label>
        </p>
        <p>
            <input type="submit" />
        </p>
    </form>
```

Votre page devrait ressembler à ceci :



The screenshot shows a web application interface. At the top, it says "Hello, maxime.chaste (You can [sign out.](#))". Below this, it says "Guestbook 'default' has no messages." and there is a large empty text area for posting a message. Below the text area is a button labeled "Post Greeting". Underneath the guestbook section, there is a section titled "Answer with picture". It contains a label "File :" followed by a file selection button labeled "Parcourir..." and the text "Aucun fichier sélectionné.". Below this is a button labeled "Envoyer". At the bottom, there is a text input field containing the word "default" and a button labeled "Ok".

- Essayez d'envoyer un fichier. De la même manière que pour le Datastore, vous pouvez voir les fichiers stockés en ligne dans Data → Blob Viewer.

Comme vous pouvez constater, cette page vous renvoie vers une url/upload. Il faut donc créer un servlet qui s'occupera de ces fichiers.

- Créez un servlet appelé *UploadServlet*, et ajoutez le code suivant :

```
...
...
import com.google.appengine.api.blobstore.BlobKey;
import com.google.appengine.api.blobstore.BlobstoreService;
import com.google.appengine.api.blobstore.BlobstoreServiceFactory;

public class UploadServlet extends HttpServlet {
    private BlobstoreService blobstoreService = BlobstoreServiceFactory.getBlobstoreService();

    @Override
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        Map<String, BlobKey> blobs = blobstoreService.getUploadedBlobs(req);
        BlobKey blobKey = blobs.get("uploadedFile");

        // A COMPLETER

    }
}
```

- Nous voulons rediriger sur une page « erreur » s'il n'y a pas de fichier et sur la page d'accueil s'il y a une image. Complétez **le code précédent**.

**Aide :** *BlobKey* contient le fichier, et une redirection a déjà été rencontrée dans le code précédent.

### Solution :

```
if (blobKey == null) {
    res.sendRedirect("/erreur");
} else {
    res.sendRedirect("/");
}
```

Pas très dur.

## Exercice 4 : Récupérer les images

De la même manière que pour l'affichage des 5 derniers messages, nous voulons afficher les images.

### Etape 1 : mise à jour de la jsp

Commencez par les imports dans votre jsp :

```
<%@ page import= "com.google.appengine.api.blobstore.BlobInfo" %>
<%@ page import= "com.google.appengine.api.blobstore.BlobInfoFactory" %>
<%@ page import= "java.util.Iterator" %>
<%@ page import= "com.google.appengine.api.images.ImagesService" %>
<%@ page import= "com.google.appengine.api.images.ImagesServiceFactory" %>
<%@ page import= "com.google.appengine.api.images.ServingUrlOptions" %>
```

Vous devez également utiliser le service image de Google à l'aide de :

```
ImagesService imagesService = ImagesServiceFactory.getImagesService();
```

### A vous de jouer

Ajoutez le contenu nécessaire dans la JSP. Vous devez trouver un moyen de récupérer tous vos « blobs », et pour chacun d'eux récupérer l'URL de l'image.

**Aide :**

1. Regardez les imports...
2. Un peu de [lecture](#) ?
3. `imagesService.getServingUrl`

### Réponse récupération des blobs

```
Iterator<BlobInfo> iterator = null;
iterator = new BlobInfoFactory().queryBlobInfos();
```

### Réponse récupération des URLs

```
ServingUrlOptions servingUrlOptions =
ServingUrlOptions.Builder.withBlobKey(iterator.next().getBlobKey());
```

A la fin vous devez obtenir :

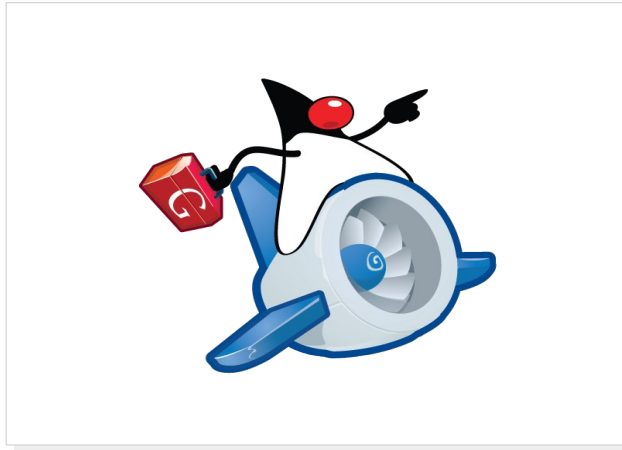
```
<%
    BlobstoreService blobstoreService = BlobstoreServiceFactory.getBlobstoreService();
    ImagesService imagesService = ImagesServiceFactory.getImagesService();
    Iterator<BlobInfo> iterator = null;
    iterator = new BlobInfoFactory().queryBlobInfos();
    while (iterator.hasNext()) {
        ServingUrlOptions servingUrlOptions =
        ServingUrlOptions.Builder.withBlobKey(iterator.next().getBlobKey());
        String imageUrl = imagesService.getServingUrl(servingUrlOptions);
        pageContext.setAttribute("image", imageUrl);
    }
%>
<p>  <br>
<%
}
%>
```



## Etape 2 : Redimensionnez l'image au format 200 \* 200

**A vous de jouer**

**Aide :** encore de la [lecture](#).



**Solution :**

```
String imageUrl = imageUrlService.getServingUrl(servingUrlOptions)+"=s200";
```

## Etape 3 : vérifier le format de l'image

Actuellement, si nous ajoutons un fichier qui n'est pas une image, nous obtenons une erreur. Quelque soit le type de fichier, lorsque nous cliquons sur « Envoyer » le fichier est envoyé directement dans le Blobstore.

**A vous de jouer**

Assurez vous que le fichier envoyé soit bien au format png.

**Aide**

Dans *UploadServlet* supprimez le blob si celui ci n'est pas au format voulu.

Il est possible d'obtenir les informations sur le blob :

```
BlobInfo blobInfo = new BlobInfoFactory().loadBlobInfo(blobKey);
```

## Solution

```
Map<String, BlobKey> blobs = blobstoreService.getUploadedBlobs(req);
BlobKey blobKey = blobs.get("uploadedFile");
BlobInfo blobInfo = new BlobInfoFactory().loadBlobInfo(blobKey);
if (blobKey == null) {
    res.sendRedirect("/erreur");
} else {
    String filename=blobInfo.getFilename();
    String extension=filename.substring(filename.lastIndexOf(".")+1);
    if(!extension.equals("png"))
    {
        blobstoreService.delete(blobKey);
        res.sendRedirect("/");
    }else{
        res.sendRedirect("/");
    }
}
```

## Exercice 5 : Associer l'utilisateur à des données

On veut maintenant pouvoir enregistrer le texte saisi, qui est envoyé en même temps que l'image.  
Voici le résultat attendu :

« un utilisateur wrote »  
le message  
l'image

Indices :

1. Nous avons donc uniquement besoin de *UploadServlet*
2. Utiliser le contenu de *SigGuestBookServlet*.

Réponse, *UploadServlet* devient :

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    Map<String, BlobKey> blobs = blobstoreService.getUploadedBlobs(req);
    BlobKey blobKey = blobs.get("uploadedFile");
    BlobInfo blobInfo = new BlobInfoFactory().loadBlobInfo(blobKey);
    if (blobKey == null) {
        res.sendRedirect("/erreur");
    } else {

        //get all usefull element
        UserService userService = UserServiceFactory.getUserService();
        DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();
        User user = userService.getCurrentUser();
        String filename=blobInfo.getFilename();
        String extension=filename.substring(filename.lastIndexOf(".")+1);
        String guestbookName = req.getParameter("guestbookName");
        String content = req.getParameter("content");
        Date date = new Date();
        Entity greeting = new Entity("Greeting");
        greeting.setProperty("user", user);
        greeting.setProperty("date", date);
        greeting.setProperty("content", content);

        if(!extension.equals("png"))
        {
            blobstoreService.delete(blobKey);
            greeting.setProperty("blobKey", null);
            res.sendRedirect("/");
            datastore.put(greeting);
        }else{
            //Ajout

            greeting.setProperty("blobKey", blobKey);
            datastore.put(greeting);

        }

        res.sendRedirect("/");}
}
```

La JSP devient :

```
<html>
<head>
</head>

<body>
<%
    String guestbookName = request.getParameter("guestbookName");
    if (guestbookName == null) {
        guestbookName = "default";
    }
%>
```

```

pageContext.setAttribute("guestbookName", guestbookName);
UserService userService = UserServiceFactory.getUserService();
User user = userService.getCurrentUser();
if (user != null) {
    pageContext.setAttribute("user", user);
}

<p>Hello,${fn:escapeXml(user.nickname)} (You can
    <a href="<%= userService.createLogoutURL(request.getRequestURI()) %>">sign out</a>.)</p>
<%
} else {
%>
<p>Hello!
    <a href="<%= userService.createLoginURL(request.getRequestURI()) %>">Sign in</a>
    to include your name with greetings you post.</p>
<%
}
%>
%>

BlobstoreService blobstoreService = BlobstoreServiceFactory.getBlobstoreService();//use the
BlobStore
DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();//use the DataStore
Query query = new Query("Greeting").addSort("date", Query.SortDirection.DESENDING);//query in
DataStore
List<Entity> greetings =
datastore.prepare(query).asList(FetchOptions.Builder.withLimit(5));//get only 5 item
if (greetings.isEmpty()) {
    <p>Guestbook '${fn:escapeXml(guestbookName)}' has no messages.</p>
} else {
    <p>Messages in Guestbook '${fn:escapeXml(guestbookName)}'.</p>
    for (Entity greeting : greetings) {
        pageContext.setAttribute("greeting_content",
            greeting.getProperty("content"));
        if (greeting.getProperty("user") == null) {
            <p>An anonymous person wrote:</p>
        } else {
            pageContext.setAttribute("greeting_user",
                greeting.getProperty("user"));
            <p><b>${fn:escapeXml(greeting_user.nickname)}</b> wrote:</p>
        }
        <blockquote>${fn:escapeXml(greeting_content)}</blockquote>
        ImagesService imagesService = ImagesServiceFactory.getImagesService();
        if (greeting.getProperty("blobKey") != null) {
            ServingUrlOptions servingUrlOptions =
            ServingUrlOptions.Builder.withBlobKey((BlobKey)greeting.getProperty("blobKey"));
            String imageUrl = imagesService.getServingUrl(servingUrlOptions)+"=s200";
            pageContext.setAttribute("image", imageUrl);
            <p>  <br>
        }
    }
}

<label> Answer with picture </label>

<form action="<%= blobstoreService.createUploadUrl("/upload") %>" method="post"
enctype="multipart/form-data">
    <p>
        <label>File : <input type="file" name="uploadedFile" /></label>
    </p> <div><textarea name="content" rows="3" cols="60"></textarea></div>
    <input type="hidden" name="guestbookName" value="${fn:escapeXml(guestbookName)}"/>
    <p>
        <input type="submit" />
    </p>
</form>

```

```
<form action="/Guestbook.jsp" method="get">
  <div><input type="text" name="guestbookName" value="{fn:escapeXml(guestbookName)}"/></div>
  <div><input type="submit" value="Ok"/></div>
</form>
```


Vous pouvez donc à présent supprimer la Servlet *SignGuestBookServlet* et modifier votre *web.xml*.  
 Vous devriez avoir une page comme celle-ci :

Hello,maxime.chaste (You can [sign out.](#))

Messages in Guestbook 'default'.

**maxime.chaste** wrote:

..



Answer with picture

File :  Aucun fichier sélectionné.

default

**OVER.**

## Aller plus loin

Pour améliorer cette application et pour vous permettre de choisir ce qui vous plaît le plus, voici une série d'exercices utilisant différents services de Google.

De plus, nous vous proposons d'autres APIs vous permettant de développer une toute nouvelle application.

Vous pouvez également aller consulter ce [lien](#) pour avoir accès à de nombreuses vidéos de présentation sur les différentes possibilités offertes par Google, et différents [articles](#).

### Proposition 1 : Ajout de vidéos de Youtube

Et pourquoi pas des playlists ? [Ici](#).

### Proposition 2 : Ajout d'un bouton « contactez nous » en utilisant des mails

Ne nécessite pas d'ajouter d'API et aucune configuration supplémentaire.

Pour vous aider, vous pouvez consulter ce [lien](#) montrant un exemple simple ou aller voir sur le [site officiel](#) pour plus de détail.

Toujours trop facile ?

Ajoutez un bouton « signaler » à côté de chaque message pour envoyer automatiquement un mail à l'administrateur.

### Proposition 3 : Ajout d'une gestion des utilisateurs

Vous souhaitez ajouter/supprimer des utilisateurs, leur donner différents droits ? Pour ce faire allez consulter ce [site](#).

Vous pouvez également utiliser le service de mail sur la proposition précédente pour améliorer votre gestion.

Toujours trop facile ?

Vos utilisateurs aimeraient peut être avoir en particulier des mails venant de leur propre contact.

Jetez donc un œil à [ceci](#).

Vous pensez encore faire mieux ? Utiliser la [map](#) de Google pour que l'on puisse savoir où sont les membres utilisant notre application.

### Proposition 4 : Ajout d'un calendrier

De base, un calendrier a pour vocation de gérer des rendez-vous. Pourquoi ne pas l'utiliser pour permettre à chaque utilisateur de connaître le nombre de messages ajoutés par jour ?

Utiliser donc l'[API](#) pour développer cette fonctionnalité.

Envie de développer votre propre application ? Voici une nouvelle série de propositions.

Coup de cœur : Utilisation du drive

Étant étudiant, nous utilisons très fréquemment le drive de Google, développez une application en rapport avec le [drive](#).

### Proposition 1: Envie de Big Data ?

Google propose une API permettant de manipuler d'énormes quantités de données, vous pouvez la découvrir [ici](#). Pour plus d'informations, nous vous invitons à regarder cette [video](#) (celle-ci étant particulièrement longue).

Vous devriez également regarder cette [API](#),

#### Proposition 2 Envie de développer un blog ?

L'api « [Blogger](#) » répond parfaitement à cette envie .

#### Proposition 3 : fan de Google Book ?

Alors développer votre propre application avec l'[API](#),

#### Proposition 4 : Sportif ?

Envie de gérer vos sport ? [Lien](#)

#### Envie d'autres choses ?

Allez consulter cette [page](#) et choisissez ce que vous voulez faire.