Технически Университет - Габрово

Катедра: КСТ

КУРСОВ ПРОЕКТ

Тема:

Да се създаде Cloud-базирано Web приложение от тип „Книга за гости”, като се използва Google App Engine

Изготвил: Борис Красимиров Гуцев

Ръководител: доц. д-р. Росен Иванов

магистратура, задочна форма на обучение

Факултетен №: 21035604

- 2012 -

Съдържание:

1. Увод

Писането на приложения за Google App Engine (GAE) на езика Java се осъществява през разработващата среда Eclipse и инсталиран към нея модул (plugin) предоставен от Google. Модулът съдържа библиотеки за работа с GAE, както и предварително изключва някой от стандартните JAVA библиотеки, тъй като в облака на Google, не се използва чиста JAVA, както и не се разрешава пълна функционалност. Примерно функции за достъп до файловата система на сървъра не се разрешават.

От гледна точка на поддръжка, следене и мониторинг на приложенията, GAE улеснява много администраторите на приложенията. Администраторът не се грижи за поддръжка на сървъри, на който е качено приложението, не се грижи за сървъри за базата от данни, не се грижи за сървър за балансиране на заявките (load balance), не се грижи на кой сървър се намира приложението, че трябва да го разпространи на няколко места в света за по бързо достъпване, не се грижи и за резервни копия на базата от данни (backup) и за още стотици неща, които получава на готово и не се грижи за тях. Получаваме и много подробна статистика за приложението, колко RAM и процесорно време употребява, колко заявки са отправени към приложението, ако се получават грешки (exceptions) се прихващат и се показват, показва логове и въобще какво се случва с това приложение. Предоставя се и функционалност за пазене на големи обеми от данни в „Blob store" структура.

2. Анализ на изискванията за приложението

Потребители:

- Потребителите да имат възможност за вход (login) чрез Google акаунт
- Да могат да пишат коментари
- Да прочитат последните 5 коментара

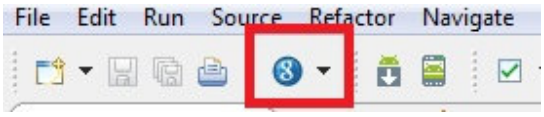
Визуализация / функционалност:

- Ако потребителя не е влязъл в акаунта си и пише коментар, да се изписва с име „anonymous"
- Да се визуализират последните 5 коментара в табличен вид, като се съдържа информация, от кого е коментара и датата на която е пуснат коментара.
- Данните се съдържат в нералационна база от данни (NoSQL – Data Store) на Google.
- Използва се Java и Eclipse за разработка.

3. Процес на разработка

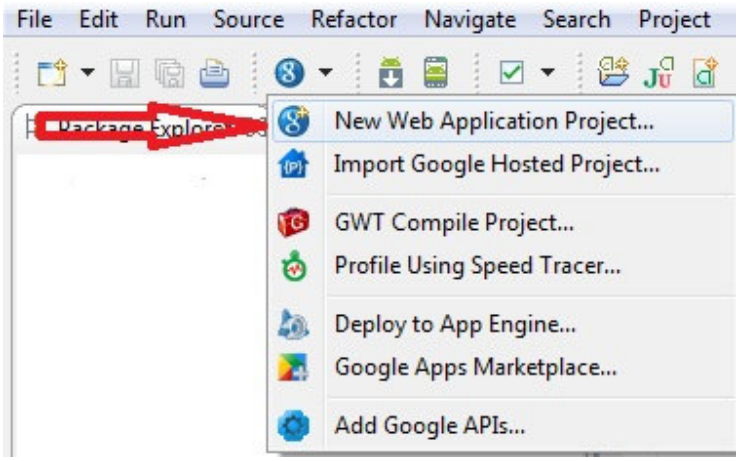
3.2. Създаване на проект

След инсталация на „App Engine SDK” към Eclipse се появява допълнително меню в Eclipse (фиг. 1 – заградено в червено).

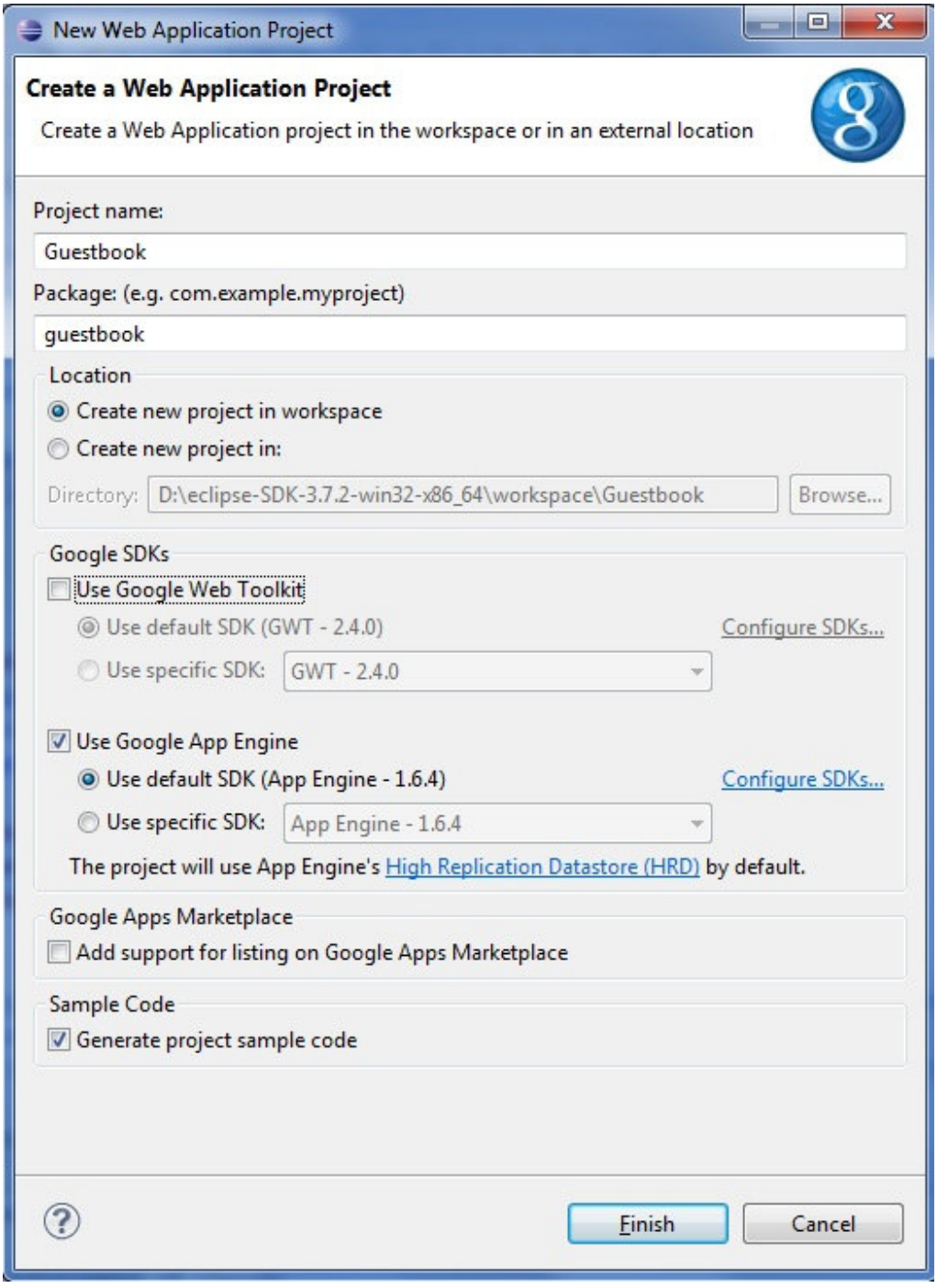


Фиг.1 Меню на GAE

От менюто се избира „New Web Application Project…”(фиг. 2), след което се попълва име на проекта (Project name) и се задава пакета в който ще се намира приложението (Package). Екрана дава възможност за още няколко допълнителни настройки, като примерно да се избере версия на SDK, къде ще се намира проекта и д.р.. **Важно** е тук да се от маркира „Use Google Web Toolkit” и да се отбележи „Use Google App Engine”. Формата в която се попълва тази информация е показана на фиг. 3. След което се натиска бутона „Finish”.



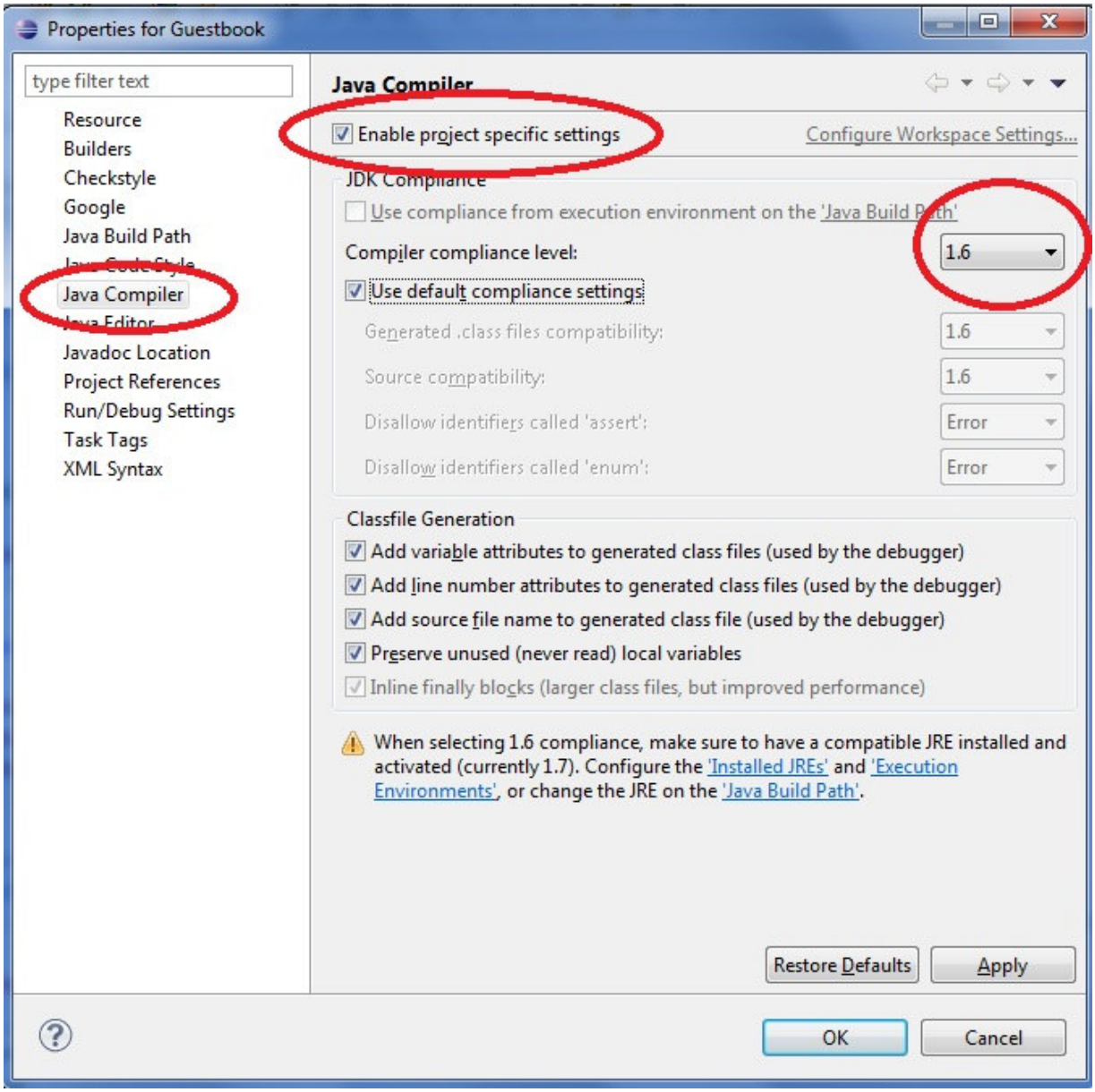
Фиг.2 Създаване на нов проект



Фиг.3 Форма за създаване на нов проект

След създаване на проекта, ако на компютъра има последна версия на JAVA (към момента JAVA 7) и Eclipse не е настроен да работи с друга версия или при създаването не е конфигурирано друго, това означава, че проекта е създаден с Java 7 (Java 1.7). За да може проекта да се качи в cloud-а на Google, версията на Java трябва да бъде “Java 1.5” или „Java 1.6”. За това,

новосъздадения проект се маркира, след което от меню „Project” се избира „Properties”, в меню “Java Compiler” се включва „Enable project specific settings”, след което от падащото меню на „Compiler compliance level” се избира „1.6” (фиг.4).



Фиг. 4 Настройка на Java версията

3.2. Структура на проекта.

Когато проекта се създаде, имаме следната структура фиг.5.

```
Guestbook/
  src/
    ...Java source code...
    META-INF/
      ...other configuration...
  war/
    ...JSPs, images, data files...
    WEB-INF/
      ...app configuration...
      lib/
        ...JARs for libraries...
      classes/
        ...compiled classes...
```

Фиг. 5 Структура на проекта

В папката “src” стои Java сорс кода на програмата. В директорията „war” се намират JSP страниците и други ресурси като картинки и файлове. Конфигурационните файлове се намират в поддиректорията “WEB-INF”. От фиг.5 се вижда че тук стоят библиотеките, който след това ще бъдат качени с приложението, стоят и основни настройки като файловете „appengine-web.xml” и „web.xml”, който са основните конфигурационни файлове.

При създаване на проекта се получава и автоматично първия сървлет „GuestbookServlet.java”, който функцията му е просто да визуализира текста „Hello, world”.

Във файла “web.xml”, се е изгенерирал автоматично кода за описание и направление на създадения сървлет (фиг. 6).

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
    <servlet>
        <servlet-name>guestbook</servlet-name>
        <servlet-class>guestbook.GuestbookServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>guestbook</servlet-name>
        <url-pattern>/guestbook</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

Фиг.6 WEB.xml файл генериран при създаване на проекта

Файлът „appengine-web.xml” се използва за конфигурация на приложението: къде, към кой проект в GAE да се качи, версия на приложението, сесии и др. Задължително трябва да се попълни името на проекта, (ID то на проекта, което се задава в GAE при създаване на нов проект) и версията на проекта. Първоначалните настройки са приложението да е версия „1” и ID-то на приложението не е попълнено (фиг.7).

```
<?xml version="1.0" encoding="utf-8"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
    <application></application>
    <version>1</version>
</appengine-web-app>
```

Фиг.7 Първоначален изглед на appengine-web.xml файла

3.3. Разработка на приложението:

3.3.1. Създаване на сървлет за вход изход.

Тъй като приложението трябва да има възможност потребители с Google акаунт да могат да влизат в акаунта си и да пишат от своето име, а не като анонимни, променям сървлета „GuestbookServlet.java”. Новия код е :

```
package guestbook;

import java.io.IOException;
import javax.servlet.http.*;
import com.google.appengine.api.users.User;
import com.google.appengine.api.users.UserService;
import com.google.appengine.api.users.UserServiceFactory;

public class GuestbookServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
              throws IOException {
        UserService userService = UserServiceFactory.getUserService();
        User user = userService.getCurrentUser();

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

Тук се използват предоставени от GAE методи за проверка дали потребителя е в профила си или не е. Редовете :

```
UserService userService = UserServiceFactory.getUserService();
User user = userService.getCurrentUser();
```

Прихващат информация за текущия потребител, като ако той не е в профила си, в променливата "**user**" ще има „**null**" стойност. След което чрез „**IF**" се проверява дали има данни за потребителя или не. Ако име, връща се стойност „Hello" като към нея се прикача потребителското име на потребителя.

Ако няма данни, се пренасочва към Google форма за вход в системата на Google. На метода за пренасочване му се подава като параметър метод за генериране на линк за вход, като на него му се подава параметър, къде да се върне след успешно влизане в системата.

```
resp.sendRedirect(userService.createLoginURL(req.getRequestURI()));
```

3.3.2. Създаване на основна JSP страница:

Потребителите ще имат възможност за влизане в системата, излизане от системата, писане на съобщения и четене на последните няколко съобщения. За целта създавам JSP страница, която ще визуализира всички функционалности на потребителя и през която той ще има достъп до тях. Името на страницата е "guestbook.jsp". Страницата съдържа следния код:

```
<%@page import="java.text.SimpleDateFormat"%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page import="java.util.List" %>
<%@ 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="com.google.appengine.api.datastore.DatastoreServiceFactory" %>
<%@ page import="com.google.appengine.api.datastore.DatastoreService" %>
<%@ page import="com.google.appengine.api.datastore.Query" %>
<%@ page import="com.google.appengine.api.datastore.Entity" %>
<%@ page import="com.google.appengine.api.datastore.FetchOptions" %>
<%@ page import="com.google.appengine.api.datastore.Key" %>
<%@ page import="com.google.appengine.api.datastore.KeyFactory" %>
<html>
    <head>
        <link type="text/css" rel="stylesheet" href="/stylesheets/main.css" />
    </head>
    <body>
    <%
        String guestbookName = request.getParameter("guestbookName");
        if (guestbookName == null) {
            guestbookName = "Boris's guestbook in cloud.";
        }
        UserService userService = UserServiceFactory.getUserService();
        User user = userService.getCurrentUser();
        if (user != null) {
    %>
          <p>Hello, <%= user.getNickname() %>! (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>
    <%
        };
    %>
```

```
    <%
        DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();
        Key guestbookKey = KeyFactory.createKey("Guestbook", guestbookName);
        Query query = new Query("Greeting", guestbookKey).addSort("date",
        Query.SortDirection.DESCENDING);
        List<Entity> greetings =
        datastore.prepare(query).asList(FetchOptions.Builder.withLimit(5));
        if (greetings.isEmpty()) {
    %>
      <p><%= guestbookName %> has no messages.</p>
    <%
        } else {
    %>
      <p>Messages in <%= guestbookName %>.</p><br/>
    <%
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss");
        for (Entity greeting : greetings) {
    %>
        <br>
        <table cellpadding="8" cellspacing="1">
            <tbody>
                <tr>
                    <td>
                        <div class="userNamePost">
                        <%
                            if (greeting.getProperty("user") == null) {
                        %>

                            <b><u>anonymous:</u></b>
                        <%
                            } else {
                        %>
                              <b><u>
                              <%= ((User)greeting.getProperty("user")).getNickname() %>
                              </u></b>
                        <%
                            };
                        %>
                        </div>
                            <div class="postDate">
                              <%= dateFormat.format(greeting.getProperty("date")) %>
                            </div>
                            <hr>
                                <span class="postContent">
                                    <%= greeting.getProperty("content") %>
                                </span>
                                <br><br>
                    </td>
                </tr>
            </tbody>
        </table>
    <%
          }
        };
    %>
    <br><br>
    <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="<%= guestbookName %>"/>
    </form>
    </body>
</html>
```

Тази JSP страница, може да се раздели на няколко части. Първата част съдържа, дали книгата за гости си има име, т.е. дали потребителя я достъпва за първи път, дали потребителя е влязъл в системата или не, като дава възможност да влезе в профила си. Генерират се линкове, ако в влязъл да излезе от Google акаунта си и обратно, ако не е да влезе.

Следващата част е извличане на данни от базата от данни в случая това е не релационната (noSQL) база от данни на Google – Datastore. Извличането на информацията става със следните редове код:

```
DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();

    Key guestbookKey = KeyFactory.createKey("Guestbook", guestbookName);

    Query query = new Query("Greeting", guestbookKey).addSort("date", Query.SortDirection.DESCENDING);

    List<Entity> greetings = datastore.prepare(query).asList(FetchOptions.Builder.withLimit(5));
```

Тук се използва предоставения от Google "DatastoreService". Създаваме си обект от него. Генерираме ключ, който сме подали и преди това при записването на постовете (показано е по долу), по него Datastore определя дали има някаква йерархия в структурата от данни. В тази база от данни няма таблици, а се сформират нещо като групи от данни, в Моя случай тази група от данни съм я кръстил „Greeting". Затова и заявката която правя, подавам името на групата и ключа. След което прибавям и сортиране по дата в низходящ ред. Резултата от заявката ще го пазя в лист от "Entity", като всъщност едно "Entity" ни е един цял пост, то съдържа цялата информация, която има в базата за този пост. Като му подавам и ограничение да ми върне само 5 резултата.

След изпълнението на заявката, проверявам дали има върнати записи, като ако няма, изписвам статичен текст, че няма все още записи в книгата.

В противен случай задавам формат за датата да бъде:

```
SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss");
```

След което сформирам динамична таблица с един "foreach" цикъл, с който минавам по всички върнати резултати от базата и за всеки от тях правя редове в таблицата. След таблицата, отпечатвам една форма с един бутон в която потребителя може да напише нов коментар.

Формата изпраща по „POST" заявка информацията към сървлет „SignGuestbookServlet.java".

3.3.3. Сървлет за обработка и запис на информация в базата от данни.

Аз съм създал сървлета с това име „SignGuestbookServlet.java", след което „web.xml" съм го променил със следното съдържание :

```
<?xml version="1.0" encoding="utf-8"?>
<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" version="2.5">
	<servlet>
		<servlet-name>Guestbook</servlet-name>
		<servlet-class>guestbook.GuestbookServlet</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>Guestbook</servlet-name>
		<url-pattern>/guestbook</url-pattern>
```

```
	</servlet-mapping>
	<servlet>
	    <servlet-name>sign</servlet-name>
	    <servlet-class>guestbook.SignGuestbookServlet</servlet-class>
	</servlet>
	<servlet-mapping>
	    <servlet-name>sign</servlet-name>
	    <url-pattern>/sign</url-pattern>
	</servlet-mapping>
	<welcome-file-list>
		<welcome-file>guestbook.jsp</welcome-file>
	</welcome-file-list>
	<security-constraint>
		<auth-constraint>
			<role-name>*</role-name>
		</auth-constraint>
	</security-constraint>
</web-app>
```

Промените са:

- Задал съм като първоначална страница да се показва моята jsp страница – „guestbook.jsp”
- Направил съм пренасочването към новия сървлет „SignGuestbookServlet.java”
- И съм задал права на достъп за всички потребители, чрез “security - constraint”

Преди да започна да пиша сървлета, в „appengine-web.xml” добавям възможност да правя и логове. Това става с добавяне на следния код:

```
<system-properties>
   <property name="java.util.logging.config.file" value="WEB-INF/logging.properties"/>
</system-properties>
```

На файла „loggin.properties” му променям съдържанието на :

```
.level = WARNING
guestbook.level = INFO
```

В „SignGuestbookServlet.java” съм реализирал вземане на информация от формата и записването и в “Datastore” на Google. Кода с който съм реализира това е :

```
package guestbook;

import com.google.appengine.api.datastore.DatastoreService;
import com.google.appengine.api.datastore.DatastoreServiceFactory;
import com.google.appengine.api.datastore.Entity;
import com.google.appengine.api.datastore.Key;
import com.google.appengine.api.datastore.KeyFactory;
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 java.util.Date;
import java.util.logging.Logger;

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

public class SignGuestbookServlet extends HttpServlet {
       private static final long serialVersionUID = 1L;
       private static final Logger log =
        Logger.getLogger(SignGuestbookServlet.class.getName());

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

          String guestbookName = req.getParameter("guestbookName");
          Key guestbookKey = KeyFactory.createKey("Guestbook", guestbookName);
          String content = req.getParameter("content");
          if (content != null && content != "") {
              Date date = new Date();
              Entity greeting = new Entity("Greeting", guestbookKey);
              greeting.setProperty("user", user);
              greeting.setProperty("date", date);
              greeting.setProperty("content", content);

              DatastoreService datastore =
                     DatastoreServiceFactory.getDatastoreService();
              datastore.put(greeting);
              if (user != null) {
                log.info("Greeting posted by user "+user.getNickname()+"date: "+
                                                                      date );
              } else {
                  log.info("Greeting posted anonymously date:  " + date);
              }
         }
         resp.sendRedirect("/guestbook.jsp?guestbookName=" + guestbookName);
    }
}
```

Тук прихващам името на книгата за гости и го поставям в променливата „guestbookName”. Образувам ключ, след което прихващам и поста на потребителя (полето „content”). Ако потребителя е въвел нещо, то се създава ново “entity“ към което се поставят полетата име на потребителя, дата на публикуване и съдържание на поста. При създаване на “entity”-то му се подава като параметър ключ и име на групата към която принадлежи. В моя случай аз имам само една група с име „Greeting”.

Създавам „datastore”обект, на който с метода „put()” записвам в базата от данни. На метода му се подава „entity” обекта който се е сформирал (greeting).

3.3.4. Създаване на CSS файл, за по добра визия на JSP страницата.

Остава да направя изгледа на JSP страницата по добър. За целта в използвам външен CSS файл с име “main.css” който съм създал в директория „stilesheets”. Включването на CSS файла JSP страницата става като в заглавната част съм го включил:

```
<head>
    <link type="text/css" rel="stylesheet" href="/stylesheets/main.css" />
</head>
```

Самият CSS файл има следното съдържание:

```
@CHARSET "UTF-8";
body {
    font-family: Verdana, Helvetica, sans-serif;
    background-color: #FFFFCC;
}
table{
       word-break: break-all;
       margin-top: 3px;
       border: 0;
       width: 80%;
       background-color: #ACCE69;
}
tr{
       background-color: #F7FAF4;
}
.userNamePost{
       margin-top: 0;
}
.postDate{
       color: gray;
       margin-top: -20px;
       float: right;
}
.postContent{
       font:normal bold 12pt;
       color: gray;
}
table tr td hr {
       color: #ACCE69;
}
```

3.3.5. Локален тест на приложението

След като всичко това е готово, стартира се приложението локално, това се прави, като се натисне на проекта с десен бутон на мишката и се избере или Run As или Debug As след което се избира „Web Application”. Това стартира локален сървър, на който се качва приложението. И вече може да се достъпва и да се дебъгва. След това съдържанието на файла „datastore-indexes.xml” може да се промени на:

```
<?xml version="1.0" encoding="utf-8"?>
<datastore-indexes
  autoGenerate="true">
    <datastore-index kind="Greeting" ancestor="true">
        <property name="date" direction="desc" />
    </datastore-index>
</datastore-indexes>
```

3.4. Създаване на акаунт и проект в GAE

Следващата стъпка е да се създаде акаунт в GAE и да се качи приложението.

При създаване на акаунт се попълва телефонен номер, след което на телефона се получава SMS със код, попълва се кода и регистрацията е готова.

Създаване на ново приложение, това се осъществява от администраторската конзола в GAE:

https://appengine.google.com/ . Натиска се бутона „Create Application" след което се следват инструкциите.

За „Application Title" и "Application Identifier" записвам „boris-guestbook". Записаното име в „Application Identifier" се записва и във файла „appengine-web.xml".

```
…
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <application>boris-guestbook</application>
  <version>1</version>
…
```

Следващата стъпка е избиране на „Authentication Options" аз съм избрал : „Open to all Google Accounts users (default)" което си е по подразбиране.

Следващата стъпка е избирана на типа база от данни, като има възможност да се избира два модела :

- High Replication (default) – този вариант се препоръчва, като той крие опасността в някои случай, когато някой чете от базата, да не получи последната версия на данните. Тъй като при записване, информацията се пръска на различни бази, записа в този случай е малко по бавен.

- Master/Slave – гарантира винаги последна версия на данните, тъй като се използва една инстанция „master" и асинхронно се прехвърлят данните в "backup" база от данни, което не бави записите в базата. Не се препоръчва за ползване "deprecated". Което означава, че постепенно ще изчезне като опция.

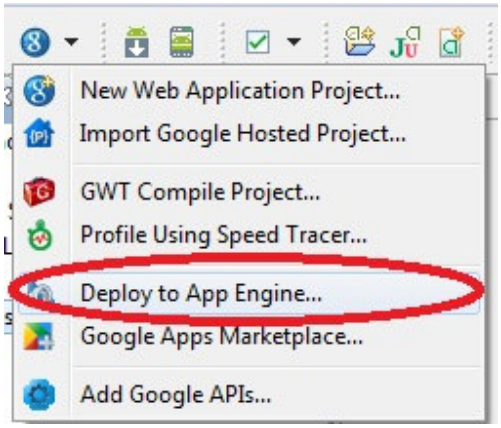
Въпреки това аз съм избрал втория вариант на базата – „Master/Slave". Важно е да се отбележи че тази конфигурация неможе да се променя в последствия. Единствения вариант е прави се ново приложение и се качва отново.

След което се натиска бутона „Create Application" и приложението е създадено.

4. Качване на приложението в GAE

След изпълнение на стъпка 3.4. приложението е готово за качване в Cloud.

В Eclipse от инсталираното GAE меню, се избира „Deploy to App Engine…” (фиг.8). Приложението вече е качено и се достъпва с името “boris-guestbook”, което записах в полето „Application Identifier” при създаването му следвано от „.appspot.com”.



Фиг.8 Качване на приложението в облака

На конзолата на Eclipse се изписва статуса и съобщения по време на качване. След което се изписва, че приложението е качено успешно.

Линк към приложението: http://boris-guestbook.appspot.com/

5. Използвана литература:

1. Сайта на Google App Engine, секция „Getting Started” : https://developers.google.com/appengine/

2. Материалите от Курс "Разработка на софтуер в cloud среда" на софтуерна академия на Телерик : http://academy.telerik.com/student-courses/cloud-software-development/resources

3. Видео материали от Курс "Разработка на софтуер в cloud среда" на софтуерна академия на Телерик : http://academy.telerik.com/student-courses/cloud-software-development/video