MACE Mobile Client presentation

Sheet one

MACE Mobile Client

MACE, european project

European project which has delivered REST based webservice for content delivery and creation. And for metatagging this content using a taxonomy.

Mobile Client

Smart phone using MACE webservice and GPS location as metadata.

Implementation

Tim de Jong implemented the webservice. I have built the mobile client for the smart phone, a HTC Touch Diamond. In the first phase we implemented content creation. Updating or deleting content will be implemented later.

Experiment in Barcelona

Students studying architecture will walk around in the city and take photo’s of buildings. By clicking on a photo they can add metadata to a part of the photo. Photo’s and tags are shared among students using MACE webservice.

Demo

I made this demo by grabbing screens from the smart phone.

Users have to log on so owner of content can be recognised.

On the first screen appears a list of real world objects. In the case of the experiment it will be buildings. Now you see only test data. As you can see on top, no GPS signal is yet found, so the list is shown alphabetically.

When you are outside, in a few seconds you get the real world objects listed ordered by distance. Distances are calculated on the server. This list is updated every minute. On top you see that GSP signal now is ok. It means the signal is based at least four satellites. When you are inside mostly no GSP signal is found, so the list stays the same. For the experiment real world objects will be added manually. In future an existing webservice delivering this data will be used.

When I click on a real world object, for instance chiba, I get a list of photo’s taken by everyone. Building this list does consume some time, depending on the amount of photo’s and the internet bandwidth. The photo’s have to downloaded. There are no thumbnails yet on the server.
When I click on a photo, I get the photo and on it the tags, indicated by the circles.

Clicking on a tag, shows the tag text. Other students have to be able to comment on tags. I'm still working on that.

So this is the content delivery part of the application. Now we will add some content. As you can see in the menu we can take a photo or add a real world object. I've also added options to stop GSP tracking temporarily, sparing the battery of the smart phone.

The photo is taken by opening the camera dialog of the smart phone. The application configures the camera to take a picture in the lowest possible resolution. So uploading and downloading the photo will be faster.

We take the photo and close the camera dialog. We then have to relate the photo to a real world object by clicking on it, I choose for chiba. As you can see below, at this point a real world object can be added too, if it's not in the list.

The photo is uploaded.

Now we can add a tag to the photo by clicking on it. Tag suggestions based on a taxonomy have to be added in this window.

The tag is uploaded and is shown on the photo.

Clicking on the tag, shows the tag text.

If we open the photo's tab, we see the photo added to the list.

Now we will add a real world object.

The real world object has GPS coordinates. Shown are the current coordinates. They can be adjusted. Of course this is not exactly the position of the building because I'm standing outside. But for the experiment it will do. As said before in future adding probably is not necessary anymore. In the mean time Google Maps could be used to add an object.

The real world object is uploaded.

And the real world object is added to the list.

Sheet two

MACE Mobile Client development

Experiment in Barcelona uses HTC Touch Diamond

HTC Touch Diamond runs Windows Mobile 6.1. This smart phone has a GPS receiver just as good as a TomTom.

Microsoft Visual Studio 2008
Because of Windows Mobile I used Visual Studio and choose C# as language. I had installation problems on my new pc, so I installed it on my old pc. It’s performing faster then Eclipse and is less memory consuming. I can start two instances simultaneously without problems.

**Windows Mobile 6 SDK**

You have to extend Visual Studio with it. You can develop your mobile app visually, by creating forms and dragging and dropping components on it. It has standard components for using the camera dialog and keyboard of the smart phone. It has emulators to test/debug your application without a smart phone. Of course taking a photo and getting GPS coordinates cannot be emulated. Then you have to debug your application on the smart phone. Application is delivered on smart phone and started using ActiveSync. Because this process is rather slow, I synchronise a Windows pc app with the mobile app to test the non mobile stuff, like using webservices, adding tags and adding real world objects. The SDK uses the compact framework. It’s a subset of the complete Visual Studio framework. This means it does not support all functionally. For instance I can detect a click on a text label within the complete framework, but not within the compact framework. Also I found a component for tracking and tracing errors was missing within the compact framework, so I’m not able to show the line number in my code where an exception took place. Further the compact framework is more strict in handling resources. I found example code to upload a picture using a HTTP request. It worked fine on the complete framework, but crashed on the compact framework. Problem was a requeststream was not closed before I read the response.

**GPS example project within Windows Mobile 6 SDK**

Luckily I could use this project to get the GPS coordinates. This project reads the GPS receiver, not only the GPS position but also the number of satellites used to calculate the position, so you get an idea of the accuracy. When I added this project as a reference project to my project I could use it. So it is quite simple to use other applications within your own.

**Newtonsoft.Json.Compact library**

The MACE webservices give their result back in Json. By adding this library as a reference within my project I could parse the Json and show the content on the screen.

**Sheet three**

**Windows Mobile 6 SDK**

**Visually programming**

You can develop your mobile app visually, by creating forms and dragging and dropping components on it.

**Standard Camera Dialog and Keyboard components**

It has standard components for using the camera dialog and keyboard of the smart phone.

**Smart phone emulators**

It has emulators to test/debug your application without a smart phone.
Debugging on the smart phone

Of course taking a photo and getting GPS coordinates cannot be emulated. Then you have to debug your application on the smart phone. Application is delivered on smart phone and started using ActiveSync.

Reference pc application

Because this process is rather slow, I synchronise a Windows pc app with the mobile app to test the non mobile stuff, like using webservices, adding tags and adding real world objects.

Compact Framework is subset

The SDK uses the compact framework. It’s a subset of the complete Visual Studio framework. This means it does not support all functionally. For instance I can detect a click on a text label within the complete framework, but not within the compact framework. Also I found a component for tracking and tracing errors was missing within the compact framework, so I’m not able to show the line number in my code where an exception took place.

Compact Framework is more strict

Further the compact framework is more strict in handling resources. I found example code to upload a picture using a HTTP request. It worked fine on the complete framework, but crashed on the compact framework. Problem was a requeststream was not closed before I read the response.

Sheet four

MACE Mobile Client experiences

Uploading/downloading content

Uploading and downloading can be slow depending on provided network and time of day. I have T-Mobile. It switches between GPRS (slower) and HSDPA (faster). I cannot influence this.

Sometimes unpredictable errors

Sometimes unpredictable error occurs when calling web services. Application stays working. Cannot be traced properly.

GPS tracking consumes energy

Delivering application: Mobile Packager

Building cab file within Visual Studio is quite easy. But building setup application not straightforward. I found Mobile Packager software. Installation file for pc using ActiveSync. Installation file for mobile. Trial version.

Showing application: Mobiola

I found Mobiola Screen Capture software for Stills, Video, Real-time. Trial version.