

# Welcome to PhotoXChange

## Table of contents

1 What's this? .....	2
2 Ancillary aims.....	2
3 History.....	3
4 Platforms.....	4
5 Tools used.....	4
6 License.....	4

## News:

- 5-Nov-2005: Version 0.3 available!

## 1. What's this?

Many times you find yourself at some event like a wedding, birthday party, sporting event, company event, or whatever event where pictures are taken. In many cases, you don't even know all of the people there, yet you do know that many people have been taking pictures. Wouldn't it be nice to have a means to easily exchange pictures with all these people afterwards?

This is where PhotoXChange comes in. PhotoXChange is a web application for exchanging photos over the web. It allows groups of people to share photos over the internet securely. Also it allows you to provide descriptions for photos and provides various options for uploading, downloading, and viewing photos.

In many cases for instance the photos themselves contain metadata describing when they were taken and this information can also be used for selecting or sorting photos. Another interesting feature is to adapt automatically to the client device. Of course, when you are using your mobile phone or other handheld device with a screen resolution of 320 x 320, it does not make much sense to send photos in this resolution. Also, the quality of the photos can be reduced a little to save bandwidth. Also, some privacy is also appreciated. As a small group of people you are probably most of the time not interested in publishing your photos for the entire world to view. Not all of these features have been implemented yet, but this should at least give you an idea of what we are after.

In other words, there are many applications or sites on the internet that allow you to share photos in some way, but this project is not about that. It is about making the ultimate application for doing that.

## 2. Ancillary aims

Apart from making a really usable application, the project has a number of other goals that are just as important:

- To provide a sound design for the application covering several concerns such as security, concurrency, and performance.
- To provide a clear separation of concerns.
- To document the most important design decisions.
- To identify reusable components that can be used for any other type of application.

- To keep it as simple as possible.
- To provide inspiration for starting other projects.
- To design for testability (and to use this for testing it!).
- To develop efficiently. Everything that can be automated should be automated or otherwise we should seriously consider not doing it.
- Last but not least: to have a lot of fun!

In other words, the application itself is not necessarily the most important thing.

### 3. History

Early 2004 I participated in a concert trip to Texas with an orchestra. I was replacing one of the guitarists who could not come. There we spent the first week practicing and giving some performances at the end of the week. The group was quite large consisting of our group and of many of the local people at which we were staying.

After getting back from this holiday I thought to my self how nice it would be to be able to exchange pictures with the people in the group but also with the people we had visited. Of course, several solutions for such a thing exist on the internet, but many of these are too public, or impose severe restrictions on the size and number of photos exchanged. I definitely wanted to exchange photos in the original quality.

So I sat down and in a few hours time, I hacked together a simple perl cgi application by which photos could be uploaded and viewed. This effort cost me about 8 hours in total to get it right. After I did this, I sent mails around notifying everyone and it was a big success. Many people uploaded their photos. Security was arranged simply through an apache configuration.

Being based on perl, the application was, ahum, not that maintainable and since I was interested in Java/J2EE, I decided to migrate the application to a Java/Servlets based version. This cost me a lot of time, since I had to learn the technology, but it worked and after some effort I had the same application but now written using Java Servlets, JSP, JSTL, and Struts.

Now history tends to repeat itself. I was not satisfied with this Struts application. The application had only two pages but already it was getting too complex for my taste. There should be a better way to design web applications without having to fully design the URL space and parameters passed with every request. It is simply not OO to develop applications in this way. Looking at this (1.5 years after the perl cgi application), I investigated many web frameworks and decided initially on Java Server Faces.

Nevertheless, after experimenting with JSF I became really disappointed. There was no

standard method for uploading files in Java Server Faces, and I had a lot of trouble working with request scope. Everything worked with session scope but I deliberately did not want to use session scope since it should be possible to independently browse photo albums in different browser windows. Finally, I decided to use Tapestry instead. I read *Tapestry in Action* like crazy and developed the same application again, but this time in about 4-6 hours. The same time it took me to develop the first application. Now this is a mature technology.

## 4. Platforms

The current version of the software has been tested on JBoss application server version 4 on SuSE Linux 9.1. Note that these are only the platforms I tested or use it on. Most likely, given the technologies used, it will already work on almost any platform. Please drop me a mail if you have gotten it working on another platform.

## 5. Tools used

Here is a list of tools that the project uses (or will use) to give you an impression of the project.

- **Hibernate:** For persistence
- **Spring:** For dependency injection and as a replacement for J2EE session beans (declarative transaction management etc.). This project will not use session and entity beans (at least not directly).
- **Tapestry:** To have true component-based development for the web interface. After using Struts and trying out Java Server Faces, tapestry is just so much easier and faster to use.
- **Ant:** For building it all. We use the ant task provided by maven 2 for downloading dependencies, just to be prepared for the time when maven 2 is ready and mature. Unfortunately, using maven 1 is not an option right now since maven 2 will be incompatible with maven 1.
- **subversion:** for version control
- **MySQL:** For the database. Supporting more databases is a piece of cake but we have to start somewhere.
- **Forrest:** For generating all documentation
- **MyEclipse:** For development within Eclipse (in particular the hotdeploy, debugging features and tapestry integration). The build environment is in no way dependent on MyEclipse.
- **MagicDraw UML:** For the UML diagrams.

*Welcome to PhotoXChange*

## **6. License**

The software is available as open source and is covered by the [Apache Software Foundation License version 2](#)