potix

SIMPLY RICH

ZK™

The Quick Start Guide

Version 2.4.0

June 2007

Potix Corporation

Table of Contents

Before You Start

New to the Servlet Container (aka., Java Web Server)

Before developing Web applications in Java (and running ZK demo in your machine), you have to install a Servlet container first. Apache Tomcat is one of the most popular Servlet containers. It is easy to install and use.

Download Installer for Windows:
apache-tomcat-5.5.20.exe

A list of all available versions:
http://tomcat.apache.org/download-55.cgi

Documentation http://tomcat.apache.org/tomcat-5.5-doc/index.html

New to Java Language

You don't need to know Java to use ZK, since all rich user interface can be implemented in HTML-like markup language called ZUML. However, to complete a Web application, you or teammate need some basic knowledge about Java. Here are some good tutorial.

Language Basic http://java.sun.com/docs/books/tutorial/java/nutsandbolts/index.html

Class and Object http://java.sun.com/docs/books/tutorial/java/concepts/index.html
http://java.sun.com/docs/books/tutorial/java/javaOO/index.html
http://java.sun.com/docs/books/tutorial/java/IandI/index.html

New to Java Integrated Development Environment (IDE)

Eclipse is one of the most popular Java IDEs. With IDE, it is easier to develop and debug your Web applications. Moreover, you can understand ZK better by debugging through ZK's source codes.

The step-by-step setup guide can be found in one of our small talks:

http://www.zkoss.org/smalltalks/eclipse/ek.html

1. What to Download

File	Description
`zk-2.4.0.tar.gz` `zk-2.4.0.zip`	The binary distribution of ZK, including ZK libraries and the source codes of `zkdemo`. It is all you need to enrich your Web applications with ZK.
`zk-demo-2.4.0.zip`	Standalone `zkdemo` application in WAR and EAR format. It is the fastest way to test drive the features of ZK without configuring your Web server.
`zk-javadoc-2.4.0.zip`	The Java API document of ZK framework.
`zk-src-2.4.0.tar.gz`	The source codes of ZK framework, including the third-party libraries to build the source codes.

2. Run the Demo

The simplest way to run the demo is to download `zk-demo-2.4.0.zip` from `http://sourceforge.net/project/showfiles.php?group_id=152762`. After uncompress the file, you can deploy `zkdemo-all.war` or `zkdemo-all.ear` to your Web or application server. Most Web or application servers have a management console that allows you to deploy an application painlessly. Consult the corresponding manuals, or you can visit `http://en.wikibooks.org/wiki/ZK/How-Tos#Installation_and_Configuration`, where the ZK community maintains the installation guides for many Web or application servers.

For Tomcat server, you can copy it directly to the `$TOMCAT_DIR/webapps` directory, and then Tomcat will start the deployment automatically.

After `zkdemo-all.war` is deployed you can visit it at, say, `http://localhost:8080/zkdemo-all/userguide`. The port number depends on how you installed your Web or application servers.

Some application servers can deploy only the EAR file. Then, you have to deploy `zkdemos-all.ear` instead.

3. Installation

ZK consists of a set of libraries. There are two ways to install them to Tomcat Web server. First, copy them to the shared directory (for Tomcat, it is `shared/lib`), so all Web applications can use them. Second, copy them to the `WEB-INF/lib` directory of the Web application, such that you can deploy your Web application to any Web server you want.

Here we illustrate first how to install ZK libraries to the shared directory. It varies from one Web server to another.

Install ZK on Tomcat

1. Download Tomcat from `http://tomcat.apache.org` and install it, if you haven't installed it yet.

2. Stop Tomcat.

3. Uncompress `zk-2.4.0.zip` or `zk-2.4.0.tar.gz`

4. Copy `dist/lib/*.jar` to `$TOMCAT_HOME`[1]`/shared/lib`

5. Copy `dist/lib/ext/*.jar` to `$TOMCAT_HOME/shared/lib`

6. [Optional] Copy `dist/lib/zkforge/*.jar to $TOMCAT_HOME`[2]`/shared/lib`
   It depends whether you need components from ZK Forge[3], such as FCKeditor (http://www.fckeditor.net) and DOJO (http://dojotoolkit.org/).

7. Re-start Tomcat.

8. Deploy `demo/bin/zkdemo.war` to Tomcat. It can be done by use of the Tomcat manager, or by copying it to `$TOMCAT_HOME/webapps` directly. If you prefer copying directly, you have to stop Tomcat first.

9. Browse to `http://localhost/zkdemo/userguide` or `http://localhost:8080/zkdemo/userguide`, depending on your configuration.

---

1 $TOMCAT_HOME is where you installed Tomcat.
2 $TOMCAT_HOME is where you installed Tomcat.
3 ZK Forge (http://zkforge.sourceforge.net) is a collection of components from the community collaboration.

Install ZK on Jetty

1. Download Jetty from `http://www.mortbay.org/jetty/index.html` and install it[4], if you haven't installed it yet.

2. Stop Jetty.

3. Uncompress `zk-2.4.0.zip` or `zk-2.4.0.tar.gz`

4. Copy `dist/lib/*.jar` to `$JETTY_HOME`[5]`/ext`

5. Copy `dist/lib/ext/*.jar` to `$JETTY_HOME/ext`

6. [Optional] Copy `dist/lib/zkforge/*.jar to $JETTY_HOME/ext`
   It depends whether you need component from ZK Forge.

7. Deploy `demo/bin/zkdemo.war` to Jetty by copying it to `$JETTY_HOME/webapps` directly.

8. Start Jetty.

9. Browse to `http://localhost/zkdemo/userguide` or `http://localhost:8080/zkdemo/userguide`, depending on your configuration.

Deploy your Application as a WAR file or an EAR file

In additions to installing ZK libraries to the Web server, you can put them into your Web application such that you can deploy your Web application to any Web server.

To bundle ZK libraries with the WAR file

Copy `dist/lib/*.jar`, `dist/zkforge/*.jar`, and `dist/lib/ext/*.jar` from `zk-2.4.0.zip` to the `/WEB-INF/lib` directory in your WAR file.

To bundle ZK libraries with the EAR file

Copy `dist/lib/*.jar`, `dist/zkforge/*.jar`, and `dist/lib/ext/*.jar` from `zk-2.4.0.zip` to the root directory of your EAR file, such that all your Web applications (in the same EAR file) can share them.

---

4 Refer to http://www.mortbay.org/jetty/tut/GettingStarted.html
5 $JETTY_HOME is where you installed Jetty.

Working with MySQL[6]
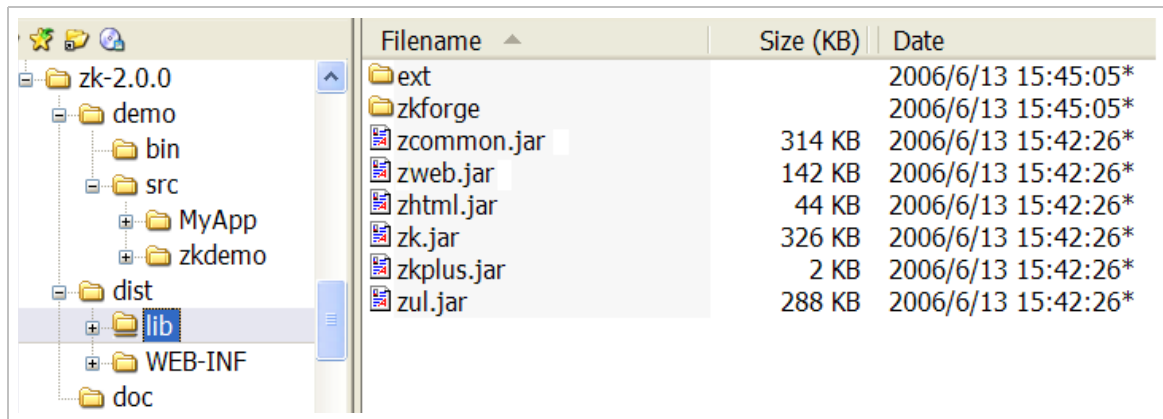
To open the connection under zscript, you have to put MySQL JDBC driver (mysql-connector-*.jar) under the `$TOMCAT_DIR/common/lib` directory.

---

6 http://ww.mysql.com

4. The Content of Distribution

This chapter describes the content of zk-2.4.0.zip.

demo

This directory holds the demo codes, including executable and source codes.

demo/bin

This directory holds the executable, `zkdemo.war`. Unlike `zkdemo-all.war` shipped with `zk-demo-2.4.0.zip`, `zkdemo.war` assumes the Web server is configured correctly with ZK. Refer to the **Installation** chapter for details.

demo/src

This directory holds the source codes of demo and samples.

demo/src/zkdemo

This directory holds the source codes of the live demo.

demo/src/MyApp

This directory holds an empty Web application which you can start your new Web application from.

doc

This directory holds the documents including Quick Start Guide and User Guide.

dist/lib

This directory holds the binary libraries required to run ZK.

dist/lib/zkforge

This directory holds the components from ZK Forge, such as FCKeditor and Dojo. It is optional depending on whether you need them.

File	Description
`dojoz.jar`	Required if you want to use ZK DOJO components.
`fckez.jar`	Required if you want to use ZK FCKeditor components.
`gmapsz.jar`	Required if you want to use ZK Google Maps components.
`timelinez.jar` `json_simple.jar`	Required if you want to use ZK Timeline components.

dist/lib/ext

This directory holds the external libraries required to run ZK. Since these libraries are common, you might have installed them in your container.

Here are optional jar files. You can choose whether to copy depending on your requirements.

File	Description
`commons-fileupload.jar` `commons-io.jar`	Required if you want to upload files with them. Version: Commons Fileupload 1.2 and Commons IO 1.3.1
`commons-el.jar`	Required if you want to evaluate EL expression with it. Version: Commons EL 1.0
`jcommon.jar` `jfreechar.jar`	Required if you want to use ZUL's chart component. Version: JFreeChart 1.0.5 and JCommon 1.0.9
`bsh.jar`	Required if you want scripting in Java interpreter (BeanShell). Version: BeanShell 2.0b4

File	Description
js.jar	Required if you want scripting in JavaScript (Rhino). Version: Rhino 1.6R5
groovy.jar	Required if you want scripting in Groovy. Version: Groovy 1.0 (groovy-all)
jruby.jar	Required if you want scripting in Ruby (JRuby). Version: JRuby 1.0.0RC3 (jruby-complete)
Filters.jar	Required if you want to use the captcha component. Version: JH Labs Java Image Filters

dist/src

This directory holds the source codes in JAR format. These JAR files are used for debugging in IDE, such as Eclipse. You cannot build the binary libraries from these. Rather, download and uncompress zk-src-2.4.0.tar.gz.

dist/WEB-INF

This directory holds the TLD and XSD files that might be useful to develop ZK applications.

5. My First ZK Application

Prepare WEB-INF/web.xml

Copy or merge the following content to the web.xml in the WEB-INF directory in your application. This step must be done once each time you created a new Web applications. Then, what you need to do is to copy files with .zul or .zhtml extension to the proper directories in your Web applications.

```xml
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

      <!-- //// -->
      <!-- ZK -->
      <listener>
             <description>Used to cleanup when a session is
destroyed</description>
             <display-name>ZK Session Cleaner</display-name>
             <listener-
class>org.zkoss.zk.ui.http.HttpSessionListener</listener-class>
      </listener>
      <servlet>
             <description>ZK loader for evaluating ZK pages</description>
             <servlet-name>zkLoader</servlet-name>
             <servlet-class>org.zkoss.zk.ui.http.DHtmlLayoutServlet</servlet-
class>


             <!-- Must. Specifies URI of the update engine
             (DHtmlUpdateServlet). -->
             <init-param>
                   <param-name>update-uri</param-name>
                   <param-value>/zkau</param-value>
             </init-param>
             <load-on-startup>1</load-on-startup><!-- MUST -->
      </servlet>
      <servlet-mapping>
             <servlet-name>zkLoader</servlet-name>
             <url-pattern>*.zul</url-pattern>
      </servlet-mapping>
      <servlet-mapping>
             <servlet-name>zkLoader</servlet-name>
             <url-pattern>*.zhtml</url-pattern>
      </servlet-mapping>
```

```xml
    <servlet>
        <description>The asynchronous update engine for ZK</description>
        <servlet-name>auEngine</servlet-name>
        <servlet-class>org.zkoss.zk.au.http.DHtmlUpdateServlet</servlet-
class>
    </servlet>
    <servlet-mapping>
        <servlet-name>auEngine</servlet-name>
        <url-pattern>/zkau/*</url-pattern>
    </servlet-mapping>
    <!-- //// -->

    <!-- MIME mapping -->
    <mime-mapping>
        <extension>gif</extension>
        <mime-type>image/gif</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>html</extension>
        <mime-type>text/html</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>htm</extension>
        <mime-type>text/html</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>jad</extension>
        <mime-type>text/vnd.sun.j2me.app-descriptor</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>jpeg</extension>
        <mime-type>image/jpeg</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>jpg</extension>
        <mime-type>image/jpeg</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>js</extension>
        <mime-type>application/x-javascript</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>png</extension>
        <mime-type>image/png</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>txt</extension>
        <mime-type>text/plain</mime-type>
    </mime-mapping>
```

```
        <mime-mapping>
                <extension>xml</extension>
                <mime-type>text/xml</mime-type>
        </mime-mapping>
        <mime-mapping>
                <extension>zhtml</extension>
                <mime-type>text/html</mime-type>
        </mime-mapping>
        <mime-mapping>
                <extension>zul</extension>
                <mime-type>text/html</mime-type>
        </mime-mapping>

        <welcome-file-list>
                <welcome-file>index.zul</welcome-file>
                <welcome-file>index.zhtml</welcome-file>
                <welcome-file>index.html</welcome-file>
                <welcome-file>index.htm</welcome-file>
        </welcome-file-list>
</web-app>
```
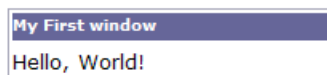
My First Hello World

Create a file called hello.zul with the following content. Then, you could use the browser to see the result, say http://localhost:8080/zkdemo/hello.zul.

```
<window title="My First window" border="normal" width="200px">
        Hello, World!
</window>
```

Then, the result is depicted as follow.

My First window
Hello, World!

Notice that, though the content of hello.zul is very similar to XUL[7], it is actually written in ZUML. ZK Loader parses it into a valid HTML page which can be interpreted correctly by a regular browser, such as Internet Explorer and Mozilla Firefox. Refer to the Developer's Guide for more details.

---

7 http://xul.sourceforge.net/mozilla.html