potix

SIMPLY RICH

ZK™

The Quick Start Guide

Version 3.6.0

February 2009

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.

Prerequisites	Description
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.

Java	URLs
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

Download ZK Library:

http://www.zkoss.org/download/zk.dsp

File	Description

`zk-bin-std-3.6.0.tar.gz`
 	
`zk-bin-std-3.6.0.zip`
	The binary distribution of ZK with the minimal set of libraries that you need to enrich your Web applications with ZK.
	
	It is also called the standard edition.

`zk-bin—prof-3.6.0.tar.gz`
 	
`zk-bin-prof-3.6.0.zip`
	The binary distribution of ZK, including the standard edition, JFreeChart, Google Maps, Jasper Reports, and layout components.
	
	It is also called the professional edition.

`zk-bin-3.6.0.tar.gz`
 	
`zk-bin-3.6.0.zip`
	The binary distribution of ZK, including the professional edition, accessibility, the performance version of ZUL (`zkmax.jar`), and versatile plug-ins, such as Ruby, Groovy, JavaScript, MVEL and OGNL.
	
	It is also called the enterprise edition.

`zk-demo-3.6.0.zip`	Standalone `zkdemo` application in WAR and EAR format, and the source codes of `zkdemo`. It is the fastest way to test drive the features of ZK without configuring your Web server.
`zk-javadoc-3.6.0.zip`	The Java API document of ZK framework.
`zk-src-3.6.0.tar.gz`	The source codes of ZK framework.

2. Run the Demo

The simplest way to test drive the power of ZK is to visit http://www.zkoss.org/zkdemo/userguide.

If you want to run the demo in your local server, you can follow the following steps.

1. Download `zk-demo-3.6.0.zip` from here (sourceforge.net).

2. After uncompressing the file, you can deploy `zkdemo.war` or `zkdemo.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.

3. After `zkdemo.war` is deployed you can visit it at, say, `http://localhost:8080/zkdemo/userguide`. The port number depends on how you installed your Web or application servers. Some application servers deploy only the EAR file. That is, you have to deploy `zkdemos.ear` instead.

3. Begin Your First ZK Project

ZK consists of several libraries (the JAR files). There are two alternatives to install them.

First, you can bundle them with your Web application. In other words, copy them to the `WEB-INF/lib` directory of your Web application. The benefit is that you can deploy your Web application to any Web server without configuring the Web server.

Second, you can install them to your Web or application server. In other words, copy them to your Web application directory (for Tomcat, it is $TOMCAT_HOME/webapps/$PROJECT_NAME/WEB-INF`/lib`).

1. **Download ZK Library:**

   http://www.zkoss.org/download/zk.dsp

2. **Create project**

   Create a development directory under $TOMCAT_HOME/webapps.
   The structure of development directory is shown below:(ex.zkdemo)

```
+zkdemo
  +WEB-INF
    web.xml
  index.zul
```

3. **Unzip ZK library**
   Unzip zk-bin-prof-xxx.zip or zk-bin-prof-xxx.tar.gz (ex. zk-bin-prof-x.x.x.zip)

4. **Deploy ZK library**
   Copy the following jar files to the $TOMCAT_HOME/webapps/$PROJECT_NAME/WEB-INF/lib (ex.zkdemo/WEB-INF/lib)

```
dist/lib/*.jar
dist/lib/zkforge/*.jar
dist/lib/ext/*.jar
```

5. **Create web.xml**
   Create web.xml under $TOMCAT_HOME/webapps/zkdemo, and Copy the following lines into web.xml

```
<!-- ZK -->
```

```
  <listener>
    <description>Used to clean up 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 ZUML pages</description>
    <servlet-name>zkLoader</servlet-name>
    <servlet-class>org.zkoss.zk.ui.http.DHtmlLayoutServlet</servlet-class>
    <init-param>
      <param-name>update-uri</param-name>
      <param-value>/zkau</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </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>

  <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>
```

6. **Create first ZUL File**
   Create index.zul under $TOMCAT_HOME/webapps/zkdemo/, and copy the following lines into it.

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

7. **Start Tomcat**
   Execute $TOMCAT_HOME/bin/start.bat to activate Tomcat.
   (ex. C:\Program Files\apache-tomcat-5.5.23\bin\start.bat)

8. **Browse zkdemo**
   Browse to http://localhost/zkdemo/ or http://localhost:8080/zkdemo/ depending on your configuration for Tomcat.

4. The Content of Binary Distribution

This chapter describes the content of zk-bin-3.6.0.zip.

/doc

This directory holds the documents including copyrights and release notes.

/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. It is optional depending on whether you need them.

File	Description
`fckez.jar`	Required if you want to use ZK FCKeditor components. Version: 2.6.1_1
`gmapsz.jar`	Required if you want to use ZK Google Maps components. Version: 2.0_8
`timelinez.jar`	Required if you want to use ZK Timeline components. Version: 1.2_1

/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 1.2.1 and Commons IO 1.3.1
`jcommon.jar`	Required if you want to use ZUL's chart component.

File	Description
jfreechar.jar	Version: JFreeChart 1.0.10 and JCommon 1.0.13 *[not available in the standard edition]*
jasperreports.jar itext.jar jxl.jar poi.jar commons-collections.jar commons-logging.jar	Required if you want to use the jasperreport component. Version: Jasper Reports 3.0.0 (itext: 2.1.3, commons-collections: 2.1, commons-logging: 1.0.2, jxl: 2.6.8, poi: 3.0.1) Note: poi.jar is required if you want to use Apache POI to generate Microsoft Excel format. And, jxl.jar is required only if you want to use JExcelApi to generate the Microsoft Excel format. *[not available in the standard edition]*
bsh.jar	Required if you want scripting in Java interpreter (BeanShell). Version: BeanShell 2.0b4
js.jar	Required if you want scripting in JavaScript (Rhino). Version: Rhino 1.7R1 *[not available in the standard and professional edition]*
groovy.jar	Required if you want scripting in Groovy. Version: Groovy 1.5.6 (groovy-all) *[not available in the standard and professional edition]*
jruby.jar	Required if you want scripting in Ruby (JRuby). Version: JRuby 1.1.2 (jruby-complete) *[not available in the standard and professional edition]*
jython.jar	Required if you want scripting in Python (Jython). Version: Jython 2.2.1 *[not available in the standard and professional edition]*
Filters.jar	Required if you want to use the captcha component. Version: JH Labs Java Image Filters *[not available in the standard edition]*

File	Description
mvel.jar	Required if you want to use MVEL to evaluate the expressions. Version: MVEL 1.2.21 (for Java 1.4 or above) *[not available in the standard and professional edition]*
ognl.jar	Required if you want to use OGNL to evaluate the expressions. Version: OGNL 2.6.9 *[not available in the standard and professional edition]*

/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-3.6.0.tar.gz`.

/dist/xsd

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

/dist/WEB-INF

This directory holds the TLD files. These TLD files are part of JAR files so they are loaded automatically. We put them here mainly for your reference only.

5. The Content of Demo Distribution

This chapter describes the content of zk-demo-3.6.0.zip.

/

This directory holds the executable: `zkdemo.war`, `zkdemo-min.war`, `zkdemos.ear` and `zkdemos-min.ear`. Refer to the **Installation** chapter for details.

/zkdemo

This directory holds the source codes of the live demo.

/MyApp

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

6. 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.

```
<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>ZK listener for session cleanup</description>
        <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>
    <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>
```

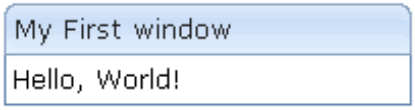
```
    <welcome-file-list>
        <welcome-file>index.zul</welcome-file>
        <welcome-file>index.zhtml</welcome-file>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
    </welcome-file-list>
</web-app>
```

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.



Notice that, though the content of hello.zul is very similar to XUL[1], 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.

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