XSLTBuddy Reference

Author: Chico Charlesworth
Revision: $Revision: 1.7 $
Last Modified: $Date: 2004/05/07 11:29:03 $

Table Of Contents

1. Preface

XSLTBuddy - XSLT + Java = XSLT rulez

XSLTBuddy hides the complex integration of XSLT and Java using XSL template functionality and provides the XSLT user with a set of extensive Java libraries. It is of course 100% Java and platform independent.

XSLTBuddy is to XSLT what TagLibs are to JSP. Yet XSLTBuddy has distinct advantages over JSP and TagLib technology (click here to find out why).

If you are new to XSLTBuddy and XSL, please follow these steps:

- Read the first chapter Quickstart for some quick tutorials to get you up and running with XSLTBuddy.
- To better understand how XSLTBuddy can help you in developing XSL driven application read Chapter 2 - XSLTBuddy Framework.
- To better understand XSL it's suggested you visit http://www.w3cschools.com.

If you have questions about XSLTBuddy or it's use, please use the forum available at XSLTBuddy's sourceforge website.

2. Getting Started

This section gives an overview of requirements and installation steps are given. A quickstart guide is also given.

The examples in the quickstart guide discuss a setup of XSLTBuddy in a standalone environment and with the Apache Tomcat servlet container.

XSLTBuddy works well with all major J2EE application servers, or within a standalone application.

2.1. Requirements

XSLTBuddy requires your environment meets some minimum requirements:

- XSLTBuddy requires a fully compliant JDK 1.4 or higher.
- XSLTBuddy is a 100% Java application and should run correctly on any system that has a compliant Java implementation.

2.2. Installation

Installing XSLTBuddy is very straightforward. Just follow the below steps.

2.2.1. Extraction Steps

- Downloaded the latest version of XSLTBuddy (available here)
- Extract the downloaded file to a directory of your choosing, let's call it <xsltbuddy install dir>.
- If you haven't got ANT installed, install it now.
- Open a command-prompt under <xsltbuddy install dir>, and run: ant

- If building a standalone application:
  - Make sure all jar files in the <xsltbuddy install dir>/lib directory are in your classpath.
  - Make sure all the XSLTBuddy XSL Templates (<xsltbuddy install dir>/templates/*.xsl) are also in your classpath.

- If building a web application:
  - Put the <xsltbuddy install dir>/lib/xsltbuddy.jar file in the /WEB-INF/lib directory
  - Put all the XSLTBuddy XSL Templates (<xsltbuddy install dir>/templates/*.xsl) files in the /WEB-INF/classes directory.
  - Your directory structure should now look something like this:

    $WEB_APPLICATION\WEB-INF\lib\xsltbuddy.jar

    $WEB_APPLICATION\WEB-INF\lib\xercesImpl.jar

    $WEB_APPLICATION\WEB-INF\lib\xml-apis.jar

    $WEB_APPLICATION\WEB-INF\lib\log4j-1.2.7.jar

    $WEB_APPLICATION\WEB-INF\lib\JAMon.jar

    $WEB_APPLICATION\WEB-INF\lib\jcommon-0.9.1.jar

    $WEB_APPLICATION\WEB-INF\lib\jfreechart-0.9.16.jar

    $WEB_APPLICATION\WEB-INF\lib\dnsjava-1.3.3.jar

    $WEB_APPLICATION\WEB-INF\lib\HTTPClient.jar

    $WEB_APPLICATION\WEB-INF\classes\xsltbuddy.xsl

    $WEB_APPLICATION\WEB-INF\classes\xsltbuddy-util.xsl

    $WEB_APPLICATION\WEB-INF\classes\xsltbuddy-http.xsl

    $WEB_APPLICATION\WEB-INF\classes\xsltbuddy-chart.xsl

    $WEB_APPLICATION\WEB-INF\classes\xsltbuddy-struts-html.xsl

- Note: All jars within <xsltbuddy install dir>/lib might not be needed depending on your purposes (see below table)

  XSLTBuddy 3rd party libraries

3rd party library	Purpose
xercesImpl.jar (required)	Used to perform XSL Transformations
xml-apis.jar (required)	Used to perform XSL Transformations
log4j-1.2.7.jar (required)	Used for logging messages
JAMon.jar (required)	Used for monitoring statistics
struts.jar (optional)	Required only if using Struts
jcommon-0.9.1.jar (optional)	Required only if chat generation is a necessity
jfreechart-0.9.16.jar (optional)	Required only if chat generation is a necessity
dnsjava-1.3.3.jar (optional)	Required only if performing DNS lookups
HTTPClient.jar (optional)	Required only if performing external HTTP request (e.g. Accesing XML Feeds)

2.2.2. Installation Steps

These installation steps are only required when building web applications.

- Amend $WEB_APPLICATION/WEB-INF/web.xml to include:

```
<servlet>
  <servlet-name>XSLTBuddy</servlet-name>
  <servlet-class>net.sf.xsltbuddy.servlet.ActionServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
...
<!-- XSLTBuddy Sevlet Mapping -->
<servlet-mapping>
  <servlet-name>XSLTBuddy</servlet-name>
  <url-pattern>*.xslt</url-pattern>
</servlet-mapping>
```

- If chart generation is a requirement, then also include in $WEB_APPLICATION/WEB-INF/web.xml:

```
<!-- Display Chart servlet -->
<servlet>
  <servlet-name>DisplayChart</servlet-name>
  <servlet-class>org.jfree.chart.servlet.DisplayChart</servlet-class>
</servlet>
...
<!-- Display Chart mapping -->
<servlet-mapping>
  <servlet-name>DisplayChart</servlet-name>
  <url-pattern>/servlet/DisplayChart</url-pattern>
</servlet-mapping>
```

2.2.3. Logging and Debugging

XSLTBuddy uses Jakarta Commons Logging for logging any messages.

Please see the Jakarta Commons Logging documentation for details on logging configuration.

2.3. Quickstart

This quickstart guide will give you a jump start to getting to grips with XSLTBuddy.

2.3.1. Simple XSLTBuddy Sample

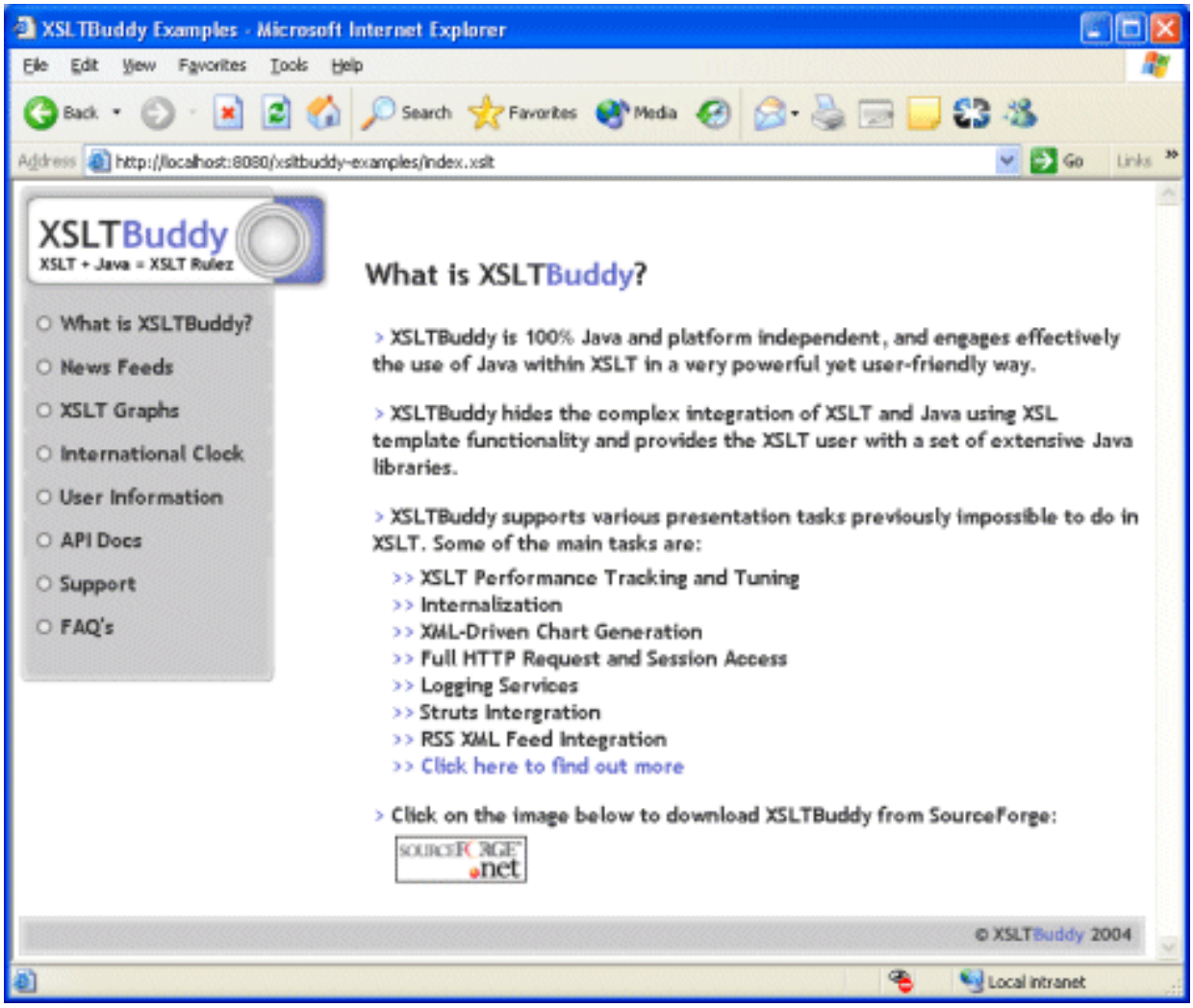
This section is still under construction. Please visit the other examples contained in the quickstart guide.

2.3.2. XSLTBuddy Examples

Use XSLTBuddy Examples as a quick start to see what XSLTBuddy is capable of.

This example uses Apache Tomcat as the servlet container.

- Downloaded the latest version of xsltbuddy-examples (available here)
- Extract the downloaded file, and copy xsltbuddy-examples.war to TOMCAT/webapps
- Run TOMCAT
- Open http://localhost:8080/xsltbuddy-examples in web browser
- You should see something like this:



- Use the navigation on the left to peruse through the examples. Note that the 'News Feed' example only works if you got a Internet Connection.

2.3.3. XSLTBuddy PetStore Demo

XSLTBuddy PetStore demo is a great example on how a real web application can work using XSLTBuddy. It also exemplifies how well XSLTBuddy integrates with Struts.

This demo uses Apache Tomcat as the servlet container, and Struts as the MVC Model 2 framework.

- Downloaded the latest version of xsltbuddy-petstore (available here)
- Extract the downloaded file, and copy xsltbuddy-petstore.war to TOMCAT/webapps
- Run TOMCAT
- Open http://localhost:8080/xsltbuddy-petstore in web browser
- You should see something like this:

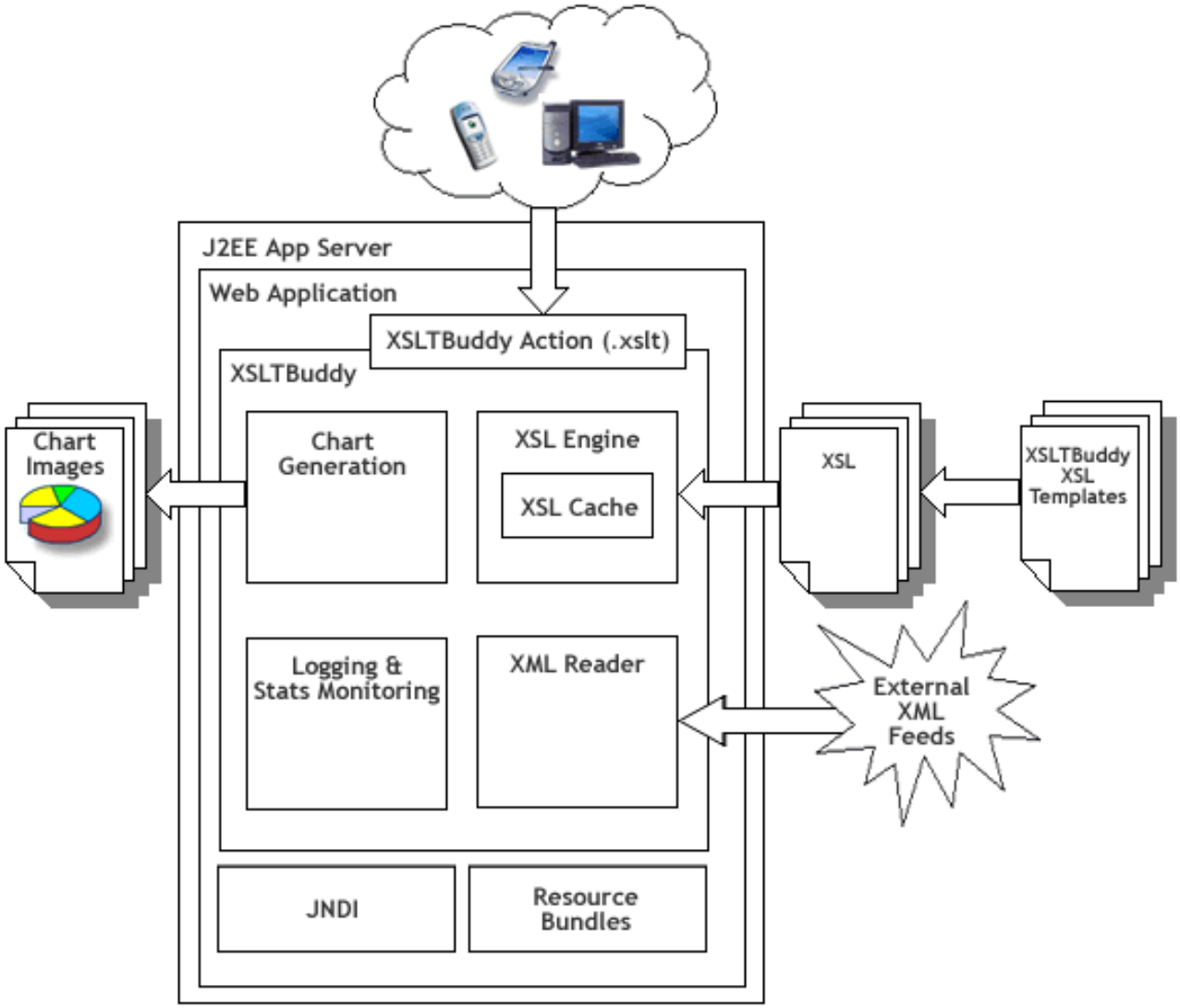
- Browse through the petstore catalog, add your favourites pets to the cart, and progress through the checkout process.

3. XSLTBuddy Framework

This section gives a brief architectural overview on how XSLTBuddy works and what features it provides.

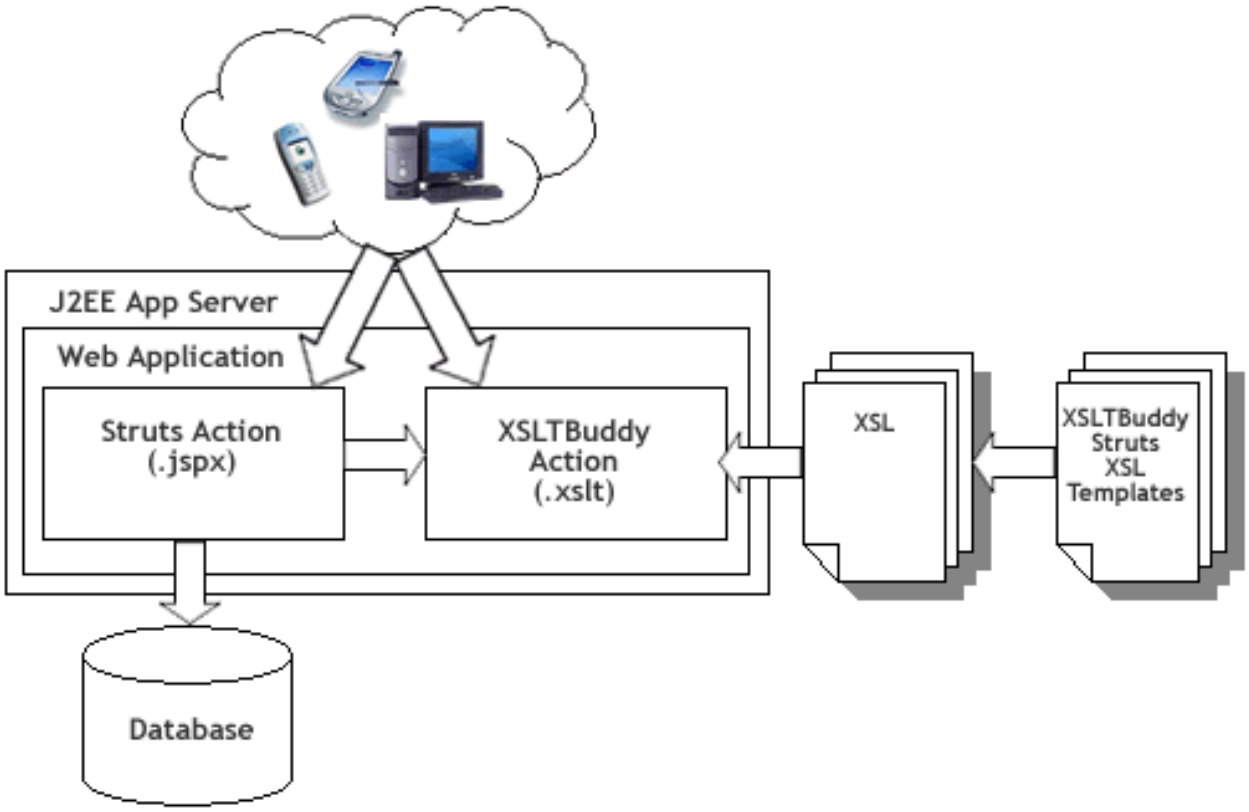
3.1. Overview

A (very) high-level view of the XSLTBuddy architecture:



This diagram shows XSLTBuddy using a XSL driven approach in displaying xml data to the end user.

XSLTBuddy also works great alongside Struts, as shown in the figure below.


J2EE App Server
Web Application
Struts Action
(.jspx)
XSLTBuddy
Action
(.xslt)
XSL
XSLTBuddy
Struts
XSL
Templates
Database

3.2. XML/XSL Driven Applications

Separating content from presentation is one of XML's major features. But eventually you need to style that XML into something presentable. That's where XSL (Extensible Stylesheet Language) comes in - XSL transforms XML into any conceivable format (html, wml, pdf, plain text, etc).

The problem arises when a developer wants to integrate XML and XSL within their application. They would have to use a mixture of servlets, jsp's, taglibs and also write a XSL transformation engine - but no longer is that the case! With XSLTBuddy you no longer need to write code to present your data using XML and XSL. A little web configuration, create your XSL file using XSLTBuddy's XSL templates and your done!

XSLTBuddy also enhances the capabilities of XSL. All presentation tasks can be dealt with within your XSL file. No more crummy jsp scripplets or messy servlets. XSLTBuddy offers logging, statistics monitoring, xml feed integration, chart generation, formatting, internationalization, Struts integration, http request handling, and much much more.

The big added bonus of all this is that the developer can 100% leave the presentation layer to the designer, and concentrate on the real issue - the business logic.

XSLTBuddy also allows XSL files to sit outside the web application and be accessed via HTTP (they're just a web resource that can be sitting anywhere). The presentation layer is then physically separated from the business and database layer, which makes a strong n-tiered application model.

3.3. Features

XSLT Performance Tracking and Tuning

XSLTBuddy gives the user the ability to track and log how long a given section of XSLT code will take to perform. This then allows the user to better analyse and performance tune the XSLT code.

JAMon is used to gather the XSLT statistics.

Internationalization

Internationalization is the process of designing a system so that it can be adapted to various languages without engineering changes.

XSLTBuddy grants the user internationalization capabilities within XSLT itself, by allowing direct access to resource bundles.

XML-Driven Chart Generation

Generating graphical reports from a XML feed is made easy by using XSLTBuddy.

There is no more need for expensive xml to java translation, and then generating the chart through JSP technology. By directly rendering the chart from XML within the XSLT code, this adds flexibility and future proof to the application.

JFreeChart is used to perform chart image generation.

Full HTTP Request and Session Access

XSLTBuddy gives access to essential user information from within the XSLT code.

How would you display different content using XSLT depending if the user is logged on or not? With XSLTBuddy you can do this very easily by doing a <xsl:if> condition around the 'user' session object.

Logging Services

XSLTBuddy provides essential logging facilities to the XSLT user.

Have you have had difficulties in debugging XSLT code? This can be remedied using XSLTBuddy by inserting simple log/debug/error statements within the XSLT code.

Log4J is used by XSLTBuddy for it's logging facilities.

XML Feed Integration

XSLTBuddy allows the user to gain direct access to XML feeds directly from the XSLT

code.

The user can access any given XML feed, translate and transform that into presentation content in 2 to 3 easy steps.

Struts Integration (Coming Soon)

This brand new feature will enable the user to easily integrate Struts straight into the XSLT code.

4. Configuration

The configuration covered in this section is only required when building web applications.

Configuring XSLTBuddy only requires simple modifications to $WEB_APPLICATION/WEB-INF/web.xml.

Example web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <display-name>XSLTBuddy Example Configuration</display-name>
  <description>Exemplifies an example of how XSLTBuddy can be
configured</description>

  <!-- XSLTBuddy servlet -->
  <servlet>
    <servlet-name>XSLTBuddy</servlet-name>
    <servlet-class>net.sf.xsltbuddy.servlet.ActionServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <!-- XSLTBuddy Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>XSLTBuddy</servlet-name>
    <url-pattern>*.xslt</url-pattern>
  </servlet-mapping>
</web-app>
```

By default, XSLTBuddy will map a given .xslt user request to the corresponding XML stylesheet file living under the $WEB_APPLICATION/xsl directory.

For example, the following url 'http://localhost:8080/myapp/index.xslt' would render the results produced by the $WEB_APPLICATION/xsl/index.xsl file.

But for very valid reasons you might want to store your XML stylesheets somewhere different. Even outside your application, on a server somewhere on the other side of the world. With XSLTBuddy you can physically separate the application's presentation and

business layers.

To specify the location of the XML stylesheets, you will need to add the <init-param> XSL_BASE under the XSLTBuddy servlet configuration. It can either be a relative mapping (e.g. /xsl) or a absolute mapping (e.g. http://localhost:8080/xsl - under TOMCAT you would deploy the XML stylesheets under TOMCAT/webapps/ROOT/xsl).

To import the XSLTBuddy templates to your XML stylesheet you will have to include the following xsl:import before you define any templates, like so:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="html" encoding="UTF-8"/>

<xsl:include href="xsltbuddy.xsl"/> <!-- XSLTBuddy XSL Include -->

<xsl:template match="/">

...

</xsl:template>

</xsl:stylesheet>
```

As you may notice, the href attribute has a relative path to xsltbuddy.xsl, and this is because when XSLTBuddy initializes it copies over the XSLTBuddy templates over to system's temp directory (under TOMCAT it's ../temp/) by default.

You can, if you wish to do so, specify which directory the templates need to be copied over to by specifying the <init-param> XSL_TEMPLATE_DIR under the XSLTBuddy servlet configuration.

Example XSLTBuddy servlet configuration:

```
<servlet>
  <servlet-name>XSLTBuddy</servlet-name>
  <servlet-class>net.sf.xsltbuddy.servlet.ActionServlet</servlet-class>
  <init-param>
    <param-name>XSL_BASE</param-name>
    <param-value>http://www.mysite.com/commonxsl</param-value>
  </init-param>
  <init-param>
    <param-name>XSL_TEMPLATE_DIR</param-name>
    <param-value>c:/temp/</param-value>
  </init-param>
```

```
  <load-on-startup>1</load-on-startup>
</servlet>
```

Chart Generation Configuration

If chart generation is a requirement, then make sure to include the following in $WEB_APPLICATION/WEB-INF/web.xml:

```
<!-- Display Chart servlet -->
<servlet>
  <servlet-name>DisplayChart</servlet-name>
  <servlet-class>org.jfree.chart.servlet.DisplayChart</servlet-class>
</servlet>
...
<!-- Display Chart mapping -->
<servlet-mapping>
  <servlet-name>DisplayChart</servlet-name>
  <url-pattern>/servlet/DisplayChart</url-pattern>
</servlet-mapping>
```

5. Posting XML into a XSLTBuddy Action

To post XML input to be transformed by the XML stylesheet, you can post it either as request parameter or a request attribute under the name of 'xml'. For example, the following HTML code would invoke the test.xsl XML stylesheet and post the XML '<Test>This is a test</Test>' to be transformed.

```
<html>
<head>
<title>XSLTBuddy Test</title>
</head>
<body>
<form name="testform" action="test.xslt" method="post">
<input type="text" name="xml" value="<Test>This is a test</Test>" size="40"/>
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

If invoking a XSLTBuddy action from a servlet or a Struts action class, then the easiest way of posting XML to be transformed is to set the XML content as a request attribute, as shown in the following code snippet:

```
String xmlContent = "<Test>This is a test</Test>";
// Set XML as request attribute
request.setAttribute("xml", xmlContent);
```

Uniquely in XSLTBuddy you don't have to provide XML to perform a XSLTBuddy XML transformation. When doing normal XSL transformations (i.e. without the help of XSLTBuddy) the user must provide the XML as input.

With XSLTBuddy the XML can be given as the input, derived within the XML stylesheet, or not provided at all.

6. XSLTBuddy and XSL

To be able to use XSLTBuddy templates within your XML stylesheets, all that is required is to have the following within your XML stylesheet:

<xsl:include href="xsltbuddy.xsl"/>

Here's an example XML stylesheet which uses XSLTBuddy:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="html" encoding="UTF-8"/>

<xsl:include href="xsltbuddy.xsl"/> <!-- XSLTBuddy XSL Include -->

<xsl:template match="/">

<html>
<head>
<title>XSLTBuddy Simple Example</title>

<xsl:call-template name="basetag"/> <!-- XSLTBuddy HTML Base Tag Template -->

</head>

<body>

Today is:
<xsl:call-template name="formatDate"> <!-- XSLTBuddy Date Formatting Template -->
  <xsl:with-param name="value" select="$today"/> <!-- $today variable constructed by XSLTBuddy -->
  <xsl:with-param name="pattern" select="'dd-MM-yyyy'"/>
</xsl:call-template>

</body>
</html>

</xsl:template>

</xsl:stylesheet>
```

7. Template Reference

This template reference is quite useful if you wan't to find out everything you need to know about a given XSLTBuddy template.

7.1. Logging

Template Name: log

Purpose: Will log the given message according to the log4j configuration. Also accepts different log levels to be outputted.

Parameter Name	Definition	Required	Possible Values	Default
msg	Message to log	X		
level	Log level		info, debug, and error	info

Example XSL snippet

```
...
<xsl:variable name="var1" select="'This is'"/>
<xsl:variable name="var2" select="' a test!'"/>
...
<xsl:call-template name="log">
  <xsl:with-param name="msg" select="'Running concatenation test'"/>
  <xsl:with-param name="level" select="'debug'"/>
</xsl:call-template>

<xsl:call-template name="log">
  <xsl:with-param name="msg" select="concat($var1, $var2)"/>
</xsl:call-template>

<xsl:call-template name="log">
  <xsl:with-param name="msg" select="'Test run successful'"/>
  <xsl:with-param name="level" select="'debug'"/>
</xsl:call-template>
...
```

7.2. Statistics Monitoring

Template Name: initmonitor
Purpose: Will commence monitoring activity.

Parameter Name	Definition	Required	Possible Values	Default
id	Monitor Identifier			XSLTBuddy Monitor

Template Name: logstats
Purpose: Will cease monitoring the given monitor identifier, and log the monitor statistics according to the log4j configuration.

Parameter Name	Definition	Required	Possible Values	Default
monitorid	Monitor Identifier	X		

Example XSL snippet

```
...
<xsl:variable name="monitorid">
  <xsl:call-template name="initmonitor"/>
</xsl:variable>
...
<xsl:call-template name="logstats">
  <xsl:with-param name="monitorid" select="$monitorid"/>
</xsl:call-template>
...
```

7.3. String, Date, and Number Formatting

String Formatting Templates

Template Name: captitalize
Purpose: Capitalizes given text.

Parameter Name	Definition	Required	Possible Values	Default
text	Text to capitalize.	X		

Template Name: wordWrap
Purpose: Word wraps given text to given width.

Parameter Name	Definition	Required	Possible Values	Default
text	Text to word wrap.	X		
width	Width to word wrap to.	X		

Template Name: flexibleWordWrap
Purpose: Word wraps given text, but will not cut words up.

Parameter Name	Definition	Required	Possible Values	Default
text	Text to word wrap.	X		
width	Width to word wrap to.	X		

Template Name: numberOfMatches
Purpose: Counts the number of occurences a given substring fits into the text given.

Parameter Name	Definition	Required	Possible Values	Default
text	Text to find occurrences.	X		
substring	Substring to count number of matches.	X		

Template Name: replace
Purpose: Replaces all pattern matches in given text with given replacement.

Parameter Name	Definition	Required	Possible Values	Default
text	Text to be transformed.	X		
pattern	Pattern to be applied.	X		
replacement	Replacement text.	X		

Template Name: replaceFirst
Purpose: Replaces the first pattern matche in given text with given replacement.

Parameter Name	Definition	Required	Possible Values	Default
text	Text to be transformed.	X		
pattern	Pattern to be applied.	X		
replacement	Replacement text.	X		

Template Name: reverse
Purpose: Reverses given text.

Parameter Name	Definition	Required	Possible Values	Default
text	Text to be reversed.	X		

Template Name: leftjustify
Purpose: Left-justify given text according to given width.

Parameter Name	Definition	Required	Possible Values	Default
text	Text to be left justified.	X		
text	Width to be applied.	X		

Template Name: rightjustify
Purpose: Right-justify given text according to given width.

Parameter Name	Definition	Required	Possible Values	Default

text	Text to be rightjustified.	X		
text	Width to be applied.	X		

Template Name: tohtml

Purpose: Replaces all line breaks with
 and & with & in given text .

Parameter Name	Definition	Required	Possible Values	Default
text	Text to be rightjustified.	X		

Template Name: towml

Purpose: Replaces all line breaks with
 , all & with &, and all currency symbols with correct unicode sequence in given text .

Parameter Name	Definition	Required	Possible Values	Default
text	Text to be rightjustified.	X		

Template Name: toMixedCase

Purpose: Normalizes given text (e.g. 'jOHn SMItH' -> 'John Smith').

Parameter Name	Definition	Required	Possible Values	Default
text	Text to be rightjustified.	X		

Number Formatting Templates

Template Name: toCurrency

Purpose: Return value as currency (e.g. '5' -> '£5.00').

Parameter Name	Definition	Required	Possible Values	Default
value	Value to be converted.	X		

Template Name: formatNumber

Purpose: Format number with given pattern.

Parameter	Definition	Required	Possible	Default

Name			Values	
value	Value to be converted.	X		
value	Pattern to be applied.	X		

Date Formatting Templates

Template Name: formatDate
Purpose: Format date with given pattern.

Parameter Name	Definition	Required	Possible Values	Default
value	Value to be converted.	X		
value	Pattern to be applied.	X		

Template Name: dateformat
Purpose: Convert date into XML Node, and apply templates (this will invoke <xsl:template match="formatted-date">). Locale can be given to convert date into a specific locale.

Parameter Name	Definition	Required	Possible Values	Default
date	Date to be converted.	X		
locale	Locale to be applied.			System Locale

Example XSL snippet

```
...
<!-- String Examples -->
...
<xsl:call-template name="capitalize"><xsl:with-param name="text" select="'this is a test!'"/></xsl:call-template>
...
<xsl:call-template name="wordWrap">
  <xsl:with-param name="text" select="'this is a test!'"/>
  <xsl:with-param name="width" select="4"/>
</xsl:call-template>
```

```
...
<xsl:call-template name="flexibleWordWrap">
  <xsl:with-param name="text" select="'this is a test!'"/>
  <xsl:with-param name="width" select="2"/>
</xsl:call-template>
...
<xsl:call-template name="numberOfMatches">
  <xsl:with-param name="text" select="'this is a test!'"/>
  <xsl:with-param name="width" select="'s'"/>
</xsl:call-template>
...
<xsl:call-template name="replace">
  <xsl:with-param name="text" select="'this is a test!'"/>
  <xsl:with-param name="pattern" select="'test'"/>
  <xsl:with-param name="pattern" select="'replace test'"/>
</xsl:call-template>
...
<xsl:call-template name="replaceFirst">
  <xsl:with-param name="text" select="'this is a test which will test replacement!'"/>
  <xsl:with-param name="pattern" select="'test'"/>
  <xsl:with-param name="pattern" select="'replace test'"/>
</xsl:call-template>
...
<xsl:call-template name="reverse"><xsl:with-param name="text"
select="'trap'"/></xsl:call-template>
...
<xsl:call-template name="leftjustify">
  <xsl:with-param name="text" select="' This is a test!'"/>
  <xsl:with-param name="text" select="8"/>
</xsl:call-template>
...
<xsl:call-template name="rightjustify">
  <xsl:with-param name="text" select="'This is a test!'"/>
```

```
  <xsl:with-param name="text" select="100"/>
</xsl:call-template>
...
<xsl:call-template name="tohtml">
  <xsl:with-param name="text">
  This is
  a test!
  </xsl:with-param>
</xsl:call-template>
...
<xsl:call-template name="towml">
  <xsl:with-param name="text">
  This is
  a test!
  </xsl:with-param>
</xsl:call-template>
...
<xsl:call-template name="toMixedCase"><xsl:with-param name="text" select="'JOHn
smITH'"/></xsl:call-template>
...
<!-- Number Examples -->
<xsl:call-template name="toCurrency"><xsl:with-param name="value"
select="5"/></xsl:call-template>
...
<xsl:call-template name="formatNumber">
  <xsl:with-param name="value" select="5"/>
  <xsl:with-param name="pattern" select="'#0.00'"/>
</xsl:call-template>
...
<!-- Date Examples -->
<xsl:call-template name="formatDate">
  <xsl:with-param name="date" select="$today"/>
  <xsl:with-param name="pattern" select="'HH:mm:ss dd-MM-yyyy'"/>
</xsl:call-template>
```

```
...
<xsl:call-template name="dateformat">
  <xsl:with-param name="date" select="$today"/>
  <xsl:with-param name="locale" select="'en'"/>
</xsl:call-template>
...
<xsl:template match="formatted-date">
  Time: <xsl:value-of select="hour"/>:<xsl:value-of select="minutes"/>:<xsl:value-of
select="seconds"/><br/>
  Date: <xsl:value-of select="day-of-week"/>, <xsl:value-of
select="month"/> <xsl:value-of select="day-of-month"/>, <xsl:value-of select="year"/>
</xsl:template>
...
```

7.4. Internationalisation

Template Name: resource

Purpose: Will get the resource for the given key in the given locale. The resource bundle can also be specified.

Parameter Name	Definition	Required	Possible Values	Default
key	Resource key	X		
locale	Locale			System Locale
bundle	Resource Bundle			xsltbuddy_resources

Example XSL snippet

```
...
<xsl:call-template name="resource">
  <xsl:with-param name="key" select="'error_msg'"></xsl:with-param>
  <xsl:with-param name="locale" select="'en'"></xsl:with-param>
</xsl:call-template>
...
```

7.5. HTTP Utils

Template Name: encode
Purpose: Encodes given url.

Parameter Name	Definition	Required	Possible Values	Default
url	URL to encode	X		

Template Name: decode
Purpose: Decodes given url.

Parameter Name	Definition	Required	Possible Values	Default
url	URL to encode	X		

Template Name: getSessionID
Purpose: Get the user's session id.

No Parameters Needed.

Template Name: getClientIP
Purpose: Get the user's IP Address.

No Parameters Needed.

Template Name: getHostName
Purpose: Do DNS Lookup on given IP Address.

Parameter Name	Definition	Required	Possible Values	Default
ip	IP Address	X		

Template Name: getParameterNames
Purpose: Get the request's parameter names.

Parameter Name	Definition	Required	Possible Values	Default
tag	XML Root to be applied	X		
childtag	Child XML Tag to be applied.			

Template Name: getHeaderNames
Purpose: Get the request's header names.

Parameter Name	Definition	Required	Possible Values	Default
tag	XML Root to be applied	X		

childtag	Child XML Tag to be applied.			

Template Name: getCookies
Purpose: Get this user's cookies.

Parameter Name	Definition	Required	Possible Values	Default
tag	XML Root to be applied	X		
childtag	Child XML Tag to be applied.			

Template Name: getParam
Purpose: Get the specified request parameter.

Parameter Name	Definition	Required	Possible Values	Default
param	Parameter Name	X		

Template Name: getHeader
Purpose: Get the specified request header.

Param Name	Definition	Required	Possible Values	Default
header	Header Name	X		

Template Name: getCookie
Purpose: Get the specified user cookie.

Param Name	Definition	Required	Possible Values	Default
position	Position in cookie list	X		

Template Name: addattribute
Purpose: Set the given attribute.

Param Name	Definition	Required	Possible Values	Default
param	Name	X		
value	Value	X		

Template Name: getattribute
Purpose: Get the requested attribute from request scope.

Param Name	Definition	Required	Possible	Default

			Values	
param	Name	X		

Template Name: findattribute
Purpose: Find the requested attribute in either request, session, or application scope.

Param Name	Definition	Required	Possible Values	Default
param	Name	X		

Template Name: xmlfeed
Purpose: Retrive the given xml feed.

Parameter Name	Definition	Required	Possible Values	Default
url	XML Feed URL	X		

Template Name: include
Purpose: Performs a static include.

Parameter Name	Definition	Required	Possible Values	Default
url	URL to include	X		

Template Name: testconnection
Purpose: Tests to see if the given URL is accessible

Parameter Name	Definition	Required	Possible Values	Default
url	URL to test the connection with.	X		

Example XSL snippet

```
...
<xsl:call-template name="addattribute">
  <xsl:with-param name="param" select="'myattribute'"/>
  <xsl:with-param name="value" select="'This is a test'"/>
</xsl:call-template>
...
<xsl:call-template name="getattribute">
  <xsl:with-param name="param" select="'myattribute'"/>
</xsl:call-template>
```

```
...
<xsl:call-template name="findattribute">
  <xsl:with-param name="param" select="'myattribute'"/>
</xsl:call-template>
...
<xsl:call-template name="xmlfeed">
  <xsl:with-param name="url"
select="'http://sports.espn.go.com/espn/rss/news'"></xsl:with-param>
</xsl:call-template>
...
<xsl:template match="rss">
  <table>
  <xsl:for-each select="./channel/item">
    <xsl:variable name="link" select="./link"/>
    <tr bgcolor="#9999CC"><td><a target="blank" class="newslink"
href="{$link}"><b> <xsl:value-of select="./title"/></b></a></td></tr>
    <tr><td> </td></tr>
    <tr><td><xsl:value-of disable-output-escaping="yes" select="./description"/></td></tr>
    <tr><td> </td></tr>
    <tr><td> </td></tr>
  </xsl:for-each>
  </table>
</xsl:template>
...
<!-- Look at the next XSL example for more examples of the other HTTP Utils
templates -->
```

Example XSL (taken from XSLTBuddy Examples App)

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="UTF-8"/>
<xsl:include href="xsltbuddy.xsl"/>
<xsl:template match="/">
```

```
<html>
<head>
<title>XSLTBuddy Examples</title>
<link rel="stylesheet" href="../style/xsltbuddy.css" type="text/css"/>
<script language="javascript" src="../js/xsltbuddy.js"></script>
</head>
<body bgcolor="#FFFFFF" leftmargin="4" topmargin="0" marginwidth="0"
marginheight="0">

<!-- MIDDLE -->
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr height="100%">

<xsl:call-template name="include">
  <xsl:with-param name="url" select="'/nav.html'"/>
</xsl:call-template>

<!-- MAIN -->
<td valign="top">

  <xsl:variable name="clientip">
  <xsl:call-template name="getClientIP"/>
  </xsl:variable>

  <xsl:variable name="sessionid">
  <xsl:call-template name="getSessionID"/>
  </xsl:variable>


  <xsl:variable name="hostname">
  <xsl:call-template name="getHostName">
    <xsl:with-param name="ip" select="$clientip"/>
  </xsl:call-template>
  </xsl:variable>

  <table>
  <tr><td><img src="../images/spacer.gif" width="1" height="20"/></td></tr>
  <tr>
```

```
  <td colspan="2" align="center"><font color="blue"><b>Request Info</b></font></td>
  </tr>

  <tr>
  <td align="right"><b>Session ID: </b></td><td><xsl:value-of
select="$sessionid"/></td>
  </tr>
  <tr>
  <td align="right"><b>Client IP: </b></td><td><xsl:value-of select="$clientip"/></td>
  </tr>
  <tr>
  <td align="right"><b>Client Host Name: </b></td><td><xsl:value-of
select="$hostname"/></td>
  </tr>

  <tr>
  <td colspan="2"> </td>
  </tr>

  <tr>
  <td colspan="2" align="center"><font color="blue"><b>Request
Headers</b></font></td>
  </tr>


  <xsl:call-template name="getHeaderNames">
  <xsl:with-param name="tag" select="'headers'"/>
  <xsl:with-param name="childtag" select="'header'"/>
  </xsl:call-template>
  <xsl:call-template name="applyTemplateResult"/>

  <tr>
  <td colspan="2"> </td>
  </tr>

  <tr>
  <td colspan="2" align="center"><font color="blue"><b>Request
Parameters</b></font></td>
  </tr>
```

```
  <xsl:call-template name="getParameterNames">
  <xsl:with-param name="tag" select="'parameters'"/>
  <xsl:with-param name="childtag" select="'parameter'"/>
  </xsl:call-template>
  <xsl:call-template name="applyTemplateResult"/>

  <tr>
  <td colspan="2"> </td>
  </tr>

  <tr>
  <td colspan="2" align="center"><font color="blue"><b>Cookies</b></font></td>
  </tr>

  <xsl:call-template name="getCookies">
  <xsl:with-param name="tag" select="'cookies'"/>
  <xsl:with-param name="childtag" select="'cookie'"/>
  </xsl:call-template>
  <xsl:call-template name="applyTemplateResult"/>


  <tr><td><img src="../images/spacer.gif" width="1" height="30"/></td></tr>
  </table>

</td>

</tr>
</table>
<!-- END OF MIDDLE -->

<xsl:call-template name="include">
  <xsl:with-param name="url" select="'/footer.html'"/>
</xsl:call-template>

</body>
</html>

</xsl:template>

<xsl:template match="headers">
```

```
  <xsl:for-each select="./header">
    <xsl:variable name="value">
      <xsl:call-template name="getHeader">
        <xsl:with-param name="header" select="."/>
      </xsl:call-template>
    </xsl:variable>
    <tr>
    <td align="right"><b><xsl:value-of select="."/>: </b></td><td><xsl:value-of
select="$value"/></td>
    </tr>
  </xsl:for-each>
</xsl:template>

<xsl:template match="parameters">
  <xsl:if test="count(./parameter) = 0">
  <tr>
  <td colspan="2" align="center">No Parameters</td>
  </tr>
  </xsl:if>
  <xsl:for-each select="./parameter">
    <xsl:variable name="value">
      <xsl:call-template name="getParam">
        <xsl:with-param name="param" select="."/>
      </xsl:call-template>
    </xsl:variable>
    <tr>
    <td align="right"><b><xsl:value-of select="."/>: </b></td><td><xsl:value-of
select="$value"/></td>
    </tr>
  </xsl:for-each>
</xsl:template>

<xsl:template match="cookies">
  <xsl:if test="count(./cookie) = 0">
  <tr>
```

```
  <td colspan="2" align="center">No Cookies</td>
  </tr>
  </xsl:if>
  <xsl:for-each select="./cookie">
    <xsl:variable name="value">
    <xsl:call-template name="getCookie">
      <xsl:with-param name="position" select="position()"/>
     </xsl:call-template>
    </xsl:variable>
    <tr>
    <td align="right"><b><xsl:value-of select="."/>: </b></td><td><xsl:value-of
select="$value"/></td>
    </tr>
  </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

7.6. JNDI (Java Naming Directory)

Template Name: addJNDIResource
Purpose: Will add a given resource to the JNDI.

Parameter Name	Definition	Required	Possible Values	Default
key	JNDI key	X		
value	Resource to add	X		

Template Name: getJNDIResource
Purpose: Will get a given resource from the JNDI.

Parameter Name	Definition	Required	Possible Values	Default
key	JNDI key	X		

Example XSL snippet

```
...
<xsl:call-template name="addJNDIResource">
  <xsl:with-param name="key" select="'test'"/>
  <xsl:with-param name="value" select="'This is a test!'"/>
</xsl:call-template>
...
<xsl:call-template name="getJNDIResource">
  <xsl:with-param name="key" select="'test'"/>
</xsl:call-template>
...
```

7.7. Chart Generation

Template Name: initdataset
Purpose: Initializes the chart data set.

Parameter Name	Definition	Required	Possible Values	Default
charttype	Chart Type		barchart,piechart,xychart	barchart

Template Name: initxyseries
Purpose: Initializes XY Series data set.

Parameter Name	Definition	Required	Possible Values	Default
id	XY series identifier.	X		

Template Name: newdataentry
Purpose: Initializes new data entry.

Parameter Name	Definition	Required	Possible Values	Default
id	Data entry identifier	X		
statkey	Statistic key	X		
statvalue	Statistic value	X		

Template Name: newxyentry
Purpose: Initializes new XY data entry.

Parameter Name	Definition	Required	Possible Values	Default
position	Data entry position	X		
statvalue	Statistic value	X		

Template Name: newxydataentry
Purpose: Specify a new XY data entry.

Parameter Name	Definition	Required	Possible Values	Default
xyseries	XY series	X		

Template Name: renderchart
Purpose: Renders the chart.

Parameter Name	Definition	Required	Possible Values	Default
title	Chart Title	X		

width	Chart Image Width	X		
height	Chart Image Height	X		
showlegend	Show Legend Flag	X		

Example XSL snippet

```
...
<!-- Bar Chart Example -->
<xsl:call-template name="initdataset">
  <xsl:with-param name="charttype" select="'barchart'"/>
</xsl:call-template>

<xsl:call-template name="newdataentry">
  <xsl:with-param name="id" select="'2002/2003'"/>
  <xsl:with-param name="statvalue" select="'15345'"/>
  <xsl:with-param name="statkey" select="'Sales'"/>
</xsl:call-template>

<xsl:call-template name="newdataentry">
  <xsl:with-param name="id" select="'2003/2004'"/>
  <xsl:with-param name="statvalue" select="'17833'"/>
  <xsl:with-param name="statkey" select="'Sales'"/>
</xsl:call-template>

<xsl:variable name="graphurl">
  <xsl:call-template name="renderchart">
    <xsl:with-param name="title" select="'Sales'"/>
    <xsl:with-param name="width" select="500"/>
    <xsl:with-param name="height" select="300"/>
    <xsl:with-param name="showlegend" select="boolean('true')"/>
  </xsl:call-template>
</xsl:variable>
<img src="{$graphurl}" width="500" height="300" border="0"/>
...
<!-- Pie Chart Example-->
```

```
<xsl:call-template name="initdataset">
  <xsl:with-param name="charttype" select="'piechart'"/>
</xsl:call-template>

<xsl:call-template name="newdataentry">
  <xsl:with-param name="id" select="'Sales'"/>
  <xsl:with-param name="statvalue" select="'15345'"/>
  <xsl:with-param name="statkey" select="'2002/2003'"/>
</xsl:call-template>

<xsl:call-template name="newdataentry">
  <xsl:with-param name="id" select="'Sales'"/>
  <xsl:with-param name="statvalue" select="'17833'"/>
  <xsl:with-param name="statkey" select="'2003/2004'"/>
</xsl:call-template>

<xsl:variable name="graphurl">
  <xsl:call-template name="renderchart">
    <xsl:with-param name="title" select="'Sales'"/>
    <xsl:with-param name="width" select="500"/>
    <xsl:with-param name="height" select="300"/>
    <xsl:with-param name="showlegend" select="boolean('true')"/>
  </xsl:call-template>
</xsl:variable>
<img src="{$graphurl}" width="500" height="300" border="0"/>
...
<!-- XY Chart Example -->
<xsl:call-template name="initdataset">
  <xsl:with-param name="charttype" select="'xychart'"/>
</xsl:call-template>

<xsl:call-template name="initxyseries">
  <xsl:with-param name="id" select="'2002/2003 Sales'"/>
</xsl:call-template>

<xsl:call-template name="newxyentry">
  <xsl:with-param name="position" select="1"/>
```

```
  <xsl:with-param name="statvalue" select="3700"/>
</xsl:call-template>

<xsl:call-template name="newxyentry">
  <xsl:with-param name="position" select="2"/>
  <xsl:with-param name="statvalue" select="3812"/>
</xsl:call-template>

<xsl:call-template name="newxyentry">
  <xsl:with-param name="position" select="3"/>
  <xsl:with-param name="statvalue" select="3123"/>
</xsl:call-template>

<xsl:call-template name="newxyentry">
  <xsl:with-param name="position" select="4"/>
  <xsl:with-param name="statvalue" select="4710"/>
</xsl:call-template>

<xsl:variable name="graphurl">
  <xsl:call-template name="renderchart">
    <xsl:with-param name="title" select="'Sales'"/>
    <xsl:with-param name="width" select="500"/>
    <xsl:with-param name="height" select="300"/>
    <xsl:with-param name="showlegend" select="boolean('true')"/>
  </xsl:call-template>
</xsl:variable>
<img src="{$graphurl}" width="500" height="300" border="0"/>
...
```

7.8. Struts Tags

Template Name: basetag
Purpose: Renders HTML base tag.

Parameter Name	Definition	Required	Possible Values	Default
servername	Server Name (e.g. http://www.mysite.co.uk/myapp/)	X		
target	Target			

Template Name: valueof
Purpose: Renders a value attribute with the value of the bean.

Parameter Name	Definition	Required	Possible Values	Default
scope	Scope		request,session	application
name	Bean Name	X		
property	Property Name			
default	Default Value			
format	Format			
formatKey	Format Key			
localeKey	Locale Key			
bundle	Bundle Name			
filter	Filter Flag			
ignore	Ignore Flag			

Template Name: beanwrite
Purpose: Outputs the value of the bean.

Parameter Name	Definition	Required	Possible Values	Default
scope	Scope		request,session	application
name	Bean Name	X		
property	Property Name			
default	Default Value			
format	Format			
formatKey	Format Key			
localeKey	Locale Key			
bundle	Bundle Name			
filter	Filter Flag			
ignore	Ignore Flag			

Template Name: selected

Purpose: Check if option tag should be selected or not.

Parameter Name	Definition	Required	Possible Values	Default
beanvalue	Bean Value	X		
optionvalue	Value of the option tag			
default	Default Flasg		true,false	

Template Name: buttontag

Purpose: Renders HTML button tag (e.g. <input type="button" name="mybutton" value="submit"/>)

Parameter Name	Definition	Required	Possible Values	Default
value	Value			
property"	Property name	X		
accesskey"	Access key			
tabindex"	Tab index			
style"	Style			
styleClass"	Style class			
styleId"	Style id			
title"	Title			
alt"	Alt			
onclick"	On mouse click			
ondblclick"	On mouse double-click			
onmouseover"	On mouse over			
onmouseout"	On mouse out			
onmousemove"	On mouse move			
onmousedown"	On mouse down			
onmouseup"	On mouse up			
onkeydown"	On key down			
onkeyup"	On key up			
onkeypress"	On key press			
onselect"	On select			
onchange"	On change			
onblur"	On blur			
onfocus"	On focus			
disabled"	Disabled flag			

readonly"	Read only Flag			

Template Name: errors
Purpose: Renders errors tag.

Parameter Name	Definition	Required	Possible Values	Default
property	Property Name			
bundle	Bundle Name			
locale	Locale			
name	Bean Name			

Example XSL snippet

```
...
<xsl:call-template name="basetag"/>
...
<xsl:call-template name="errors"/>
...
<xsl:variable name="myflag">
<xsl:call-template name="beanwrite">
   <xsl:with-param name="name" select="'myForm'"/>
   <xsl:with-param name="property" select="'flag'"/>
</xsl:call-template>
</xsl:variable>

<select name="flag">
  <option value="">Select</option>
  <option value="true">
    <xsl:call-template name="selected">
       <xsl:with-param name="beanvalue" select="$myflag"/>
       <xsl:with-param name="optionvalue" select="'true'"/>
    </xsl:call-template>
    True
  </option>
  <option value="false">
    <xsl:call-template name="selected">
       <xsl:with-param name="beanvalue" select="$myflag"/>
```

```
        <xsl:with-param name="optionvalue" select="'false'"/>
    </xsl:call-template>
    False
  </option>
</select>
...
<input type="text" name="name">
  <xsl:call-template name="valueof">
    <xsl:with-param name="name" select="'myForm'"/>
    <xsl:with-param name="property" select="'name'"/>
  </xsl:call-template>
</input>
...
<xsl:call-template name="buttontag">
  <xsl-with name="property" select="'mybutton'" />
  <xsl-with name="value" select="'submit'" />
</xsl:call-template>
...
```

8. FAQ's

This section walks through some Frequently Asked Questions about XSLTBuddy.

8.1. Why use XSLTBuddy?

XSLTBuddy gives you the best of XSLT and Java.

XSLT enables users with limited programming and web page design experience to easily create and modify templates to define how data is displayed to the end user.

Java provides true extensibility and reusability to the XSLT-based application by offering functionality currently impossible to implement solely in XSLT.

8.2. Why use Java and XML technologies?

Java and XML are based on industry standards, and share many features that are ideal for building web-based enterprise applications, like platform-independence, extensibility, reusability, and internationalization support.

Java and XML shows how to build real-world applications in which both the code and the data are truly portable.

8.3. XSLTBuddy vs. JSP

1. XSLTBuddy is XML driven. JSP aren't XML aware.

2. JSP files must sit within the application realm. Using XSLTBuddy, the presentation and business layer can be physically separated, because the XML stylesheets can be accessed anywhere - within or outside the application realm.

3. JSP technology requires java development skills. XSLTBuddy uses XSL templates, which only requires XSL skills which can be learnt easily and applied by less technical personnel (e.g. designers).

4. XSLTBuddy provides functions that are a lot more flexible, extensible, and object-oriented than TagLibs.

TagLib technology has the following deficiencies:

> TagLibs are also not XML aware

> TagLibs don't accept objects as parameters

> TagLibs have a limit on the number of parameters they can accept

> TagLibs don't support arrays or collections

> TagLibs can't be configured or changed on the fly

> TagLib functionality can't be changed on the fly

8.4. Do you need to know Java to use XSLTBuddy?

No. Even though XSLTBuddy is based on Java technology, by using XSL templates to make the Java functionality accessible, it hides the complex XSL and Java integration from the XSL user.

Of course to run XSLTBuddy in a web application or in a standalone application, someone technically aware of Java technologies will have to initially setup that environment.

9. Finally ...

XSLTBuddy is easy to use. All that's required is to learn how to use the XSL templates provided by XSLTBuddy. If you are new to XSL, then you will need to get acquainted with it.

It's strongly recommended that you use a XSL editor like XML Spy to write your XML stylesheets. A good XSL and XSLT reference can be found here along with some good XSLT tutorials.