Dart: Testing, Reports and Dashboards

Daniel J. Blezek
A. G. Amitha Perera

June 2, 2005

Acknowledgments

This product includes software developed by the Apache Software Foundation (http://www.apache.org/).
This product includes software developed by the Visigoth Software Society (http://www.visigoths.org/).

CHAPTER

ONE

Introduction

1.1 Dart Statement of Purpose

Dart shall aggregate data across many independent distributed build and test hosts, summarizing the software quality aspects of the project in a concise and informative fashion cross-sectionally and longitudinally.

CHAPTER

TWO

Tutorial

2.1 Quick Start

Assuming the Dart Server software has been installed, a basic Project can be created and started easily.

1. Create the Server directory and configuration

   ```
   java -classpath DartServer.jar dart.DartServer --createserver TestServer
   ```

   The `--createserver` flag creates a new Server directory and an default configuration file Server.xml.

2. Initialize the server

   ```
   java -classpath DartServer.jar dart.DartServer --initializeserver TestServer
   ```

   Initializes the Server database.

3. Create the project

   ```
   java -classpath DartServer.jar dart.DartServer  --create TestProject
   ```

   The `--create` flag requires a directory name (`TestProject`) for the Project.

4. Start the project

   ```
   java -classpath DartServer.jar dart.DartServer \
                --initialize --refresh TestServer TestProject
   ```

The `--initialize` flag instructs the server to create the database tables that Dart requires for the project, while `--refresh` copies the project resources into the TestProject directory. TestServer is the name of the Dart Server, while any projects to be started can be configured in TestServer/Server.xml and are overridden by the commandline arguments. Note that \ indicates line continuation; that is, the code above should be typed on one line.

5. View the dashboard. Point your browser to `http://localhost:8081/TestProject/Dashboard/` to view the (empty) Dashboard.

The project `TestProject` is now up and running accepting XML-RPC submissions on port `8080` and serving HTML pages on port `8081`. The ports and other configurations are covered in Section 2.4.

2.2 Submission

Dart ships with a utility called DartClient to submit results to the server. The basic use is:

```
# java -classpath DartServer.jar dart.DartClient TestProject Results.xml
```

This submits Results.xml to the `TestProject` Project on the Server running on `localhost`. Submission is only a copy, and does not provide feedback on the XML validity.

DartClient also provides other options:

```
# java -classpath DartServer.jar dart.DartClient --help
0 [main] INFO dart.DartClient  - Starting DartClient
usage: DartClient [options] Project <foox.xml> <foo2.xml> ... <fooN.xml>
 -g,--getstatus   Get Server status
 -h,--help        Print help message
 -p,--port        XML-RPC Port to connect to, 8080 is default
 -q,--shutdown    Shutdown the Server
 -r,--refresh     Refresh Project resources
 -s,--server      Server to connect to, localhost is default
```

To connect through a proxy or firewall use:

```
java -Dhttp.proxyPort=8080 -Dhttp.proxyHost=proxyhost.mydomain.org -classpath DartServer.jar dart.Dar
```

2.2.1 ctest Integration

The latest CVS version of ctest (http://www.cmake.org/) can be used to submit to a Dart Server. Instructions may be found at http://na-mic.org/Wiki/index.php/User:Andy. The settings needed in the project's CMake file are:

```
ENABLE_TESTING()
INCLUDE(Dart)
```

```
  SET (DROP_METHOD "xmlrpc")
  SET (DROP_SITE "http://myserver.org:8081")
  SET (DROP_LOCATION "TestProject")
  SET (COMPRESS_SUBMISSION ON)

  ADD_TEST(name executable arg1 arg2 ...)  # see CMake documentation
```

In `DartConfiguration.tcl`, this translates to:

```
DropSite: http://myserver.org:8081/
DropLocation: TestProject
DropMethod: xmlrpc
```

2.3 Software Installation

To be completed.

2.4 Server Setup

The Server has several command line options.

```
# java -classpath DartServer.jar dart.DartServer
usage: DartServer [options] Server.xml <Project0.xml> <Project1.xml> ...
                  <ProjectN.xml>
 -c,--create              Create a new project in the directory specified
 -d,--database            At project creation time, configure the
                          Schema.sql file for generic, Postgres, Derby
 -h,--help                Print help message
 -i,--initialize          Initialize the database from the Schema.sql file
                          in the project directory
 -j,--initializeserver    Initialize the database from the
                          ServerSchema.sql file in the dart server directory
 -k,--createserver        Create a new server in the directory specified
 -l,--logconfiguration    File to configure log4j from, defaults are used
                          if not present
 -r,--refresh             Refresh project resources
```

2.5 Server Configuration

In the `TestServer` directory, there is a file named `Server.xml`. This contains the default settings for a dart server. The sections of the Server configuration are as follows.

2.5.1 Server Info

```
<?xml version="1.0" encoding="utf-8"?>
<Server>
  <Title>TestServer</Title>
  <BaseDirectory>f:\Source\Dart\TestServer</BaseDirectory>
```

This is the XML preamble followed by the Server information: Title and BaseDirectory.

2.5.2 Ports

```
  <HTTPPort>8081</HTTPPort>
  <XMLRPCPort>8080</XMLRPCPort>
```

These are the HTTP and XML-RPC port specifications. They are required to be different.

2.5.3 Scheduler

```
  <Scheduler>
    <ThreadPoolSize>10</ThreadPoolSize>
  </Scheduler>
```

The Scheduler has a default ThreadPoolSize of 10, indicating that 10 jobs may be executed concurrent by all the Projects managed by this Server instance.

2.5.4 Database

```
  <!-- Configure the database parameters derby-->
  <Database>
    <!-- Derby database -->
    <Driver>org.apache.derby.jdbc.EmbeddedDriver</Driver>
    <URL>jdbc:derby:f:\Source\Dart\TestServer/Database/TestServer;create=true</URL>
    <ShutdownURL>jdbc:derby:f:\Source\Dart\TestServer/Database/TestServer;shutdown=true</ShutdownURL>
    <Username/>
    <Password/>
    <!-- Maximum active / idle connections, -1 is infinite -->
    <MaxActive>10</MaxActive>
    <MaxIdle>3</MaxIdle>
  </Database>
```

This section specifies the connection to the Server's database. In this example, the database is Derby. The Driver tag specifies the class implementing the JDBC connection. URL is the connection string, ShutdownURL is used to shutdown the Derby database cleanly, it may be safely left blank for other JDBC packages. The Username and Password tags specify the connection parameters. Dart uses a connection pooling mechanism for the database with two parameters: MaxActive specifies the maximum number of active connections, and MaxIdle specifies the maximum number of idle threads. If a connection is needed when MaxActive threads are already active, the connection will hang until a connection is return to the pool.

2.5.5 Servlet Manager

```
<!-- Servlet configuration -->
<ServletManager>
  <Servlet>
    <Class>dart.server.servlet.Server</Class>
    <Context>/Dart/*</Context>
    <Properties/>
  </Servlet>
</ServletManager>
```

The Servlet manager is responsible for configuring Jetty. Different Servlets can respond to different URLs. In this case, the dart.server.servlet.Server class is configured to respond to requests starting with /Dart/*.

2.6 Project Configuration

After following the directions in Section 2.1, in the `TestProject` directory, there is a file named `Project.xml`. This is a pre-configured Project configuration file containing all the settings required to run a basic Dart Project. The contents of the file are not presented in their entirety, but will be discussed section by section.

2.6.1 Project Info

```
<?xml version="1.0" encoding="utf-8"?>
<Project>
  <Title>TestProject</Title>
  <BaseDirectory>/projects/Dart/TestProject</BaseDirectory>
```

The first line is the `xml` preamble, and is required for all `xml` files. The `<Project>` tag indicates the start of the Project configuration. `<Title>` is the project title, and `<BaseDirectory>` is the absolute path name to the Project directory. If the Project is moved to a new location on the file system, `<BaseDirectory>` must be changed to reflect the new location.

2.6.2 Database Configuration

```
  <!-- Derby database -->
  <Database>
    <Driver>org.apache.derby.jdbc.EmbeddedDriver</Driver>
    <URL>jdbc:derby:/projects/Dart/TestProject/Database/TestProject;create=true</URL>
    <ShutdownURL>jdbc:derby:/projects/Dart/TestProject/Database/TestProject
                 ;shutdown=true</ShutdownURL>
    <Username/>
    <Password/>
  </Database>
```

This section configures the Database connection that the Project will use. The <Driver> tag indicates the JDBC Java class for the particular type of relational database management system (RDBMS). In the example, org.apache.derby.jdbc.EmbeddedDriver is the driver for the Derby Open Source embedded RDBMS system. Note: the <ShutdownURL> was broken across two lines for display purposes, and should not be broken in an actual configuration file.

The <URL> tag specifies the connection URL for the RDBMS. This is a RDBMS specific string. In the above example, the create=true property indicates that the driver should create the database if it does not exist. Please consult your RDBMS documentation for the proper setting for the <URL> tag. Because Derby is an embedded RDBMS, it must be properly shutdown to leave the database in a consistent state. This is specified in the <ShutdownURL> tag. If the RDBMS does not require special shutdown processing, leave this tag empty and it will be ignored.

<Username/> and <Password> tags specify the authentication settings for the RDBMS. In the case of Derby, no Username/Password is required.

2.6.3 CommandManager Configuration

```
<CommandManager>
  <Command>
    <Name>Submit</Name>
    <Class>dart.server.command.Submit</Class>
    <Properties>
      <Property name="DeleteWhenDigested">true</Property>
    </Properties>
  </Command>
</CommandManager>
```

The Dart Server provides an XML-RPC server for results to be submitted to a Project. This server operates through a Servlet configured in the ServletManager (see Section 2.6.4 below). For the CommandManager to operate, a dart.server.servlet.CommandServlet object must be added to the ServletManager. In addition, the CommandServer can be configured to respond to any query using specialized Commands. The <CommandServer> section specifies the settings for the Project specific settings.

In the instance above, the <Command> tag specifies an object that the Project will use to respond to XML-RPC calls. Commands must implement the dart.server.command.Command interface. <Name> is the object name, <Class> is the name of the class that the CommandManager instanciates and any Properties for the object are specified usingthe <Properties> tags. Any public methods of the object are exposed to XML-RPC calls.

2.6.4 ServletManager Configuration

```
<!-- Servlet configuration -->
<ServletManager>
  <Servlet>
    <Class>dart.server.servlet.Dashboard</Class>
    <Context>/Dashboard/*</Context>
    <Properties>
```

```
    </Properties>
  </Servlet>
  <Servlet>
    <Class>dart.server.servlet.CommandServlet</Class>
    <Context>/Command/</Context>
    <Properties/>
  </Servlet>
```

To generate Dashboard pages, the Server uses the Jetty Servlet engine in conjunction with the FreeMarker template engine. Stock Project Servlets are automatically configured at project creation time. User defined Servlets may be added if desired. The `<Class>` tag indicates the class of the Servlet, `<Context>` tag indicates how the Servlet is found by Jetty. By default, the Project title is stored in the Servlet's initial parameters as `''project''` and may be accessed as `getInitParameter ( ''project'' )` within the Servlet. Parameters in the `<Properties>` section are also put in the initial parameters map.

The second Servlet in the stock configuration is dart.server.servlet.CommandServlet. CommandServlet accepts XML-RPC calls and delegates them to the appropriate handler object as configured in the CommandManager.

To call an XML-RPC method, the URL needed is determined by the root project URL, *i.e.* http://localhost:8081/*ProjectName*/Command/*Command.Method*. For example, the URL to submit some results to the Dart project TestProject running on the local system is: http://localhost:8081/TestProject/Command/ and the method is `Submit.put`.

2.6.5 Task Configuration

```
<!-- Scheduled tasks.  The Schedule tag is in cron format. -->
<Task>
  <Type>dart.server.task.QueueManager</Type>
  <Schedule>0/10 * * * * ?</Schedule>
  <Properties>
    <Property name="MaxTasks">10</Property>
  </Properties>
</Task>
```

Tasks configured in the `Dart.xml` file are periodically scheduled. Tasks must implement the `dart.server.task.Task` interface. In the above example, the `dart.server.task.QueueManager` is scheduled to run every 10 seconds. The `QueueManager` class processes other Tasks that have been placed in the TaskQueue. The `Properties` tag specifies settings that are passed into the Task when it executes. For `QueueManager`, the "MaxTasks" property indicates how many queued tasks will be processed at during each execution, providing a "throttling" mechanism. The format of the `<Schedule>` tag is detailed at `http://quartz.sourceforge.net/javadoc/org/quartz/CronTrigger.html`.

Another example Task is the `SaveStatistics` Task which writes the Project statistics to the file system on a regular bases. The Task does not have any parameters, and saves the statistics in a file named `Statistics.txt` in the Project directory.

```
<Task>
  <Type>dart.server.task.SaveStatistics</Type>
```

```
  <Schedule>0 * * * * ?</Schedule>
  <Properties>
  </Properties>
</Task>
```

CHAPTER

THREE

Dart Development

3.1 Requirements

To work on Dart, you will need:

- Subversion (http://subversion.tigris.org/). Dart source code is maintained in a Subversion repository.
- Java SDK (http://java.sun.com). Version 1.4.2 or later is needed.
- Apache Ant (http://ant.apache.org/), version 1.6.2 or greater. This is a build system, similar in concept to Unix Makefiles.
- JUnit (http://www.junit.org/). Java unit testing framework. This is used to define and run regression tests on the Dart source. The JUnit jar file is included in the checkout. Drop the `junit.jar` file in `ant/lib` directory to enable JUnit to run as an ant task.
- The Dart source (see below).

The other packages required by Dart, such as Quartz and Jaxor, are available as part of the Dart source. You do not need to obtain these separately.

3.2 Obtaining the source

Obtain a copy of the source code by checking it out of the repository:

```
cd MySrc
svn co http://svn.na-mic.org:8000/svn/Dart
```

This will create a directory `MySrc/Dart` containing the current Dart source.

If you have a HTTP proxy server, you will need to specify the variables `http-proxy-exceptions`, `http-proxy-host` and `http-proxt-port` in your `/.subversion/servers` (Unix) or `c:/Documents and Settings/User/Application Data/Subversion/servers` (Windows) file. Refer to the Subversion documentation for more details.

3.3 Build the source

The most straight forward method of building is

```
cd MySrc/Dart
ant all
```

basic steps are

```
cd MySrc/Dart
ant wrap
ant compile
ant jar
ant test
```

Each of “wrap”, “compile”, “jar” and “test” are compile targets, similar to Makefile targets. The full list is:

wrap Generate the Jaxor wrapping code. This generates Java objects to wrap the SQL queries defined in `Source/Wrap`. The wrapping process can be time consuming, and so is not run automatically for every compile. Wrap must be run when any of the Jaxor sources changes.

compile Compile the `.java` files to `.class` files. This is the default target.

jar Generate `DartServer.jar` containing the compiled Dart code.

test Run regression tests, with summary output.

testverbose Run regression tests with verbose output.

clean Clean the `.class` files.

fullclean Clean the `.class` files and the `.java` files generated by “wrap” above.

doc Runs JavaDoc to generate the API documentation into `Documentation/api`.

all Does a clean compile of Dart, runs the tests and builds the jar file.

3.4 Troubleshooting

- ‘Unexpected element “setproxy” ’

  ▷ You need a newer version of Apache Ant.

- ‘package dart.server.wrap does not exist’, while compiling.

  ▷ You didn’t run “wrap” before “compile”.

- ‘org.apache.velocity.runtime.exception.ReferenceException: reference : template = FinderImpl.vm [line 45,column 36] : ${primaryKeyQuery.getMethodName()} is not a valid reference’, while wrapping.

  ▷ The wrapping process did not execute correctly. This could be due to clock skew on NFS mounted file systems, which incorrectly causes some rules to not fire.

CHAPTER

FOUR

Dart Requirements and Design

This chapter describes the requirements and design criteria for the next version of Dart.

4.1 Dart Statement of Purpose

> *Dart shall aggregate data across many independent distributed build and test hosts, summarizing the software quality aspects of the project in a concise and informative fashion cross-sectionally and longitudinally.*

4.2 User Requirements

1. A single server instance shall process multiple projects, with simple, flexible configuration and management.
2. Presentation of results shall be configurable, allowing results to persist on the dashboard for different periods. For instance, coverage information is time consuming to produce but slowly changing and ought to persist for more than one day.
3. Dashboards may be aggregated into Meta-Dashboards. For instance, Slicer depends on VTK, ITK, gsl and Tcl/Tk. The Slicer Meta-Dashboard shall present summary information from these dependencies.
4. Dart shall support submission authentication and selectively reject or expire unauthenticated submissions.
5. Dart shall provide resource management tools for disk space, bandwidth and processing time allowing both Clients and Servers to efficiently manage resources.

4.3 Design Requirements

Basic

1. The server shall contain all components required and shall not require any external packages, nor operating system applications. The server shall run as a daemon and shall include these components:
   (a) Scheduler: Dart shall include an internal scheduling system for routine systems tasks, *etc*...

(b) RDBMS: Dart shall include an embedded database to handle small Projects.

(c) Web Server: Dart shall include an embedded web server to publish dashboard pages.

(d) Web Services: Dart shall communicate using an established protocol for web services, allowing Results submissions and query of Project status from remote, homogeneous clients.

2. The server shall be extensible with user supplied components, including:

   (a) RDBMS: Dart shall use JDBC compliant drivers for all DB access allowing different database systems such as MySQL, Postgres, Oracle, *etc.*

   (b) Web Server: Apache and other web servers shall be capable of serving Dart generated pages.

   (c) Web Services: Dart shall allow the ability to communicate using external web servers such as Apache, Tomcat, *etc.*

   (d) Portal Server: If desired, a Portal server such as Jetspeed may be used to interface with Dart results database. This capability is currently unspecified.

Resource Management (Section 4.3.1)

1. Dart shall, as an option, maintain compressed XML files, using on-the-fly decompression. This will result in approximately 10:1 spacing savings for the XML.

2. Dart shall provide a policy mechanism to selectively delete or archive unnecessary Builds. An archived Build shall consume less than 10K of disk space by retaining only summary information.

Storage, Processing and Presentation Engines

1. Dart shall comprise three engines: Storage, Processing and Presentation.

2. The Storage engine shall accept submissions from clients parse the input and store results in a generic format with large data items stored in the file system, *i.e.* images, with numeric and shorter text information stored in a database system.

3. The Processing engine shall process and summarize the results organized by the Storage engine at regular intervals and upon user-defined event triggering processing actions.

4. The Presentation engine shall provide a customizable view of data: both “raw” data from the Storage engine, and from the Processing engine. In the first instance, the Presentation engine shall simply be HTML, potentially migrating to a Portal based server.

Customization

1. Dart shall provide an easy to modify template engine for summarizing results.

2. Dart shall provide a server side plug in mechanism allowing custom data aggregation and flexible reporting.

3. Dart shall make provide mechanisms for simple localization and internationalization, where appropriate.

4.3.1 Resource Management

Disk Space

The Insight toolkit is the largest Dart project to date. Currently, with compressed HTML files, one day consumes over 650M of disk space. This includes (from November 4, 2004):

- One Doxygen run (400K for XML, 13K for HTMLZ)
- One Master Update (13K for XML, 4K for HTMLZ)
- One Dashboard (12M for XML, 9K for HTMLZ)
- One BuildOverview (250K for XML, 8K for HTMLZ)
- One TestOverview (25M for XML, 2 x 29K for HTMLZ)
- One Coverage build (24M for XML, 21M for HTMLZ)
- Builds (average of 9M for XML and HTMLZ)
  - 52 Nightly Builds
  - 19 Continuous Builds
  - 22 Experimental Builds

Breaking down an example day, we have:

File	Size	Notes
Build.xml	140K	111 Warnings, average of 1.2K per warning
Configure.xml	0.8K	
Test.xml	4.3M	859 Tests, average of 5K per test
Update.xml	1.4K	1 Update, 1.4K per updated file
TestSummary.xml	215K	
All.htmlz	86K	Total for 8 HTMLZ files

The largest generator of data is test output. Errors/Warnings and Update information are rather verbose, capturing context information. In general, XML is verbose with low entropy. A 4.3M Test.xml file is 522K compressed with gzip.

Dashboard Generation Time

Bandwidth

4.3.2 Historical Data

Dart currently preserves data from previous days, it is not linked across temporal Builds on the same system. While simple, this restriction increases the difficulty of monitoring the quality of a project. To overcome this limitation, Dart shall link data in a temporal fashion.

4.3.3 Hierarchical Data

Dashboards

Tests

Builds

4.3.4 Persistence of Builds/Results

Stream Concept

4.3.5 Documentation

4.3.6 Submissions

Incremental Submission

Mechanisms

Authentication

4.3.7 Configuration

Initial Setup

Options

4.3.8 Customization

Dashboard presentation

4.3.9 Extensibility

CHAPTER

FIVE

Design

5.1 Server

The Dart server is implemented in Java. It is composed of several different services, outlined below.

5.2 DartServer

The DartServer is responsible for starting up the other services. Projects are created, configured, loaded and started by the DartServer class.

5.2.1 Command Manager

Commands to Dart are passed to the DartServer via XML-RPC. The DartServer starts up a the Apache XML-RPC server on the same port as the HTTP Server by default.

5.2.2 Scheduler

The Quartz enterprise Scheduler is initialized and passed to each Project. In turn, each Project adds Tasks to the Scheduler to be executed as needed.

5.2.3 HTTP Server

Jetty is used in an embedded mode to serve static content, and generate dynamic content.

5.3 Project

Each Project hosted on a DartServer is created by loading the Dart.xml file in the Project directory. A Project is composed of several components.

5.3.1 Database

The Database object coordinates all access to the underlying RDBMS. The Database provides Connections to other Project components as needed.

5.3.2 ResultServer

The ResultServer object is responsible for handling XML-RPC requests. During startup, a service is added to the DartServer SubmissionServer.

5.3.3 ServletManager

The ServletManager is responsible for creating the Project specific Servlets and adding them to the Server's HTTP Server. User Servlets may be added to the Project's Plugins directory.

CHAPTER

SIX

Implementation Ideas

This section captures some implementation ideas.

6.1 Server

Language Of all cross-platform languages, Java provides the most robust set of libraries suitable for Dart. Java also allows simple distribution of compiled libraries, *i.e.* jar files, as plug-ins. Potentially, a client could query the server for a list of available plug-ins downloading and installing as needed.

RDBMS There are several embeddable Java RDBMS available, two of the more interesting projects are Cloudscape, recently released from IBM, and renamed Derby on the Apache site and Hypersonic SQL (HSQLDB) project hosted on SourceForge. Dart is envisioned to have a RDBMS holding summary data; embedding a database into the server should help to make it transparent and invisible to the casual Dart user. For more scalability, the backing store could be any RDBMS with a JDBC driver. MySQL and Postgres come to mind.

Transport Though over-designed and complex, SOAP has the elements need to transmit XML files to the server from the client. Specifically, SOAP with attachments could deliver chunks of compressed XML to the server via HTTP, since most (all?) firewalls allow HTTP traffic. SOAP could also be used for Dashboard to Dashboard (D2D?) communication and remote management and monitoring of Dart servers. XML-RPC is a much simpler API, and identically suitable. XML-RPC will be considered at the same level as SOAP. Another possible use is dissemination of plug-ins for clients. The Java Messaging Service (JMS) is another possibility. JMS gives great flexibility to transport mechanisms and can operate asynchronously.

Scheduling Quartz is an open source enterprise strength scheduling system for Java. Quartz will drive scheduled events such as Dashboard roll ups, DB tasks, and archiving/deletion of old results. Quartz will replace cron.

Template Engine There are several competing Template engines for Java. Velocity is an Apache sponsored project and has some great features including close integration with other Apache projects. FreeMarker is another engine that is more sophisticated than Velocity, but not as integrated. The Template engine will be the driver to produce HTML and other reports replacing XSLT.

Jakarta The Apache Jakarta project provides several packages of immediate use.

- Digester builds objects from XML, greatly simplifying configuration from XML files. Each object is constructed as needed and automatically configured.
- CLI should provide a great command line parsing interface.
- Commons eMail provides a simple java email client.
- ORO and RegExp, two regular expression packages.

Portal Though the current Dart HTML pages serve the purpose well, adding a portal on top would allow custom portlets to be developed for specific purposes. For instance, one portlet could be configured to show a particular build over the last several days, or perhaps graph the performance of a Test or Result through time across several architectures. Dynamic generation of all the Dart results places undo burden on the server, where a Portal could dynamically generate limited data in an efficient manner. One Portal project that is interesting is Jetspeed 2, an Apache sponsored project.

Portals do add administrative overhead. It is preferable to have the ability to use Dart without a Portal, but easily being able to add the increased utility if desired.

6.2 Client

CHAPTER

SEVEN

External Packages

7.1 Packages

Dart is built upon many Open Source packages. Each of these packages has different licenses. To comply with the licenses of each of these, we have listed the packages, their licenses and copyrights in this chapter.

Apache v1.1 Apache XML-RPC, Apache CLI

Apache Version 2.0 Bean Utilities, Derby, Collections, DBCP, Digester, Pool, VFS, Jetty

BSD License Jaxor

Common Public License, v1.0 JUnit

BSD-Like license Quartz

Freemarker License Freemarker

7.2 Apache License, Version 1.1

```
The Apache Software License, Version 1.1


Copyright (c) 2001 The Apache Software Foundation.  All rights
reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
```

notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.

3. The end-user documentation included with the redistribution,
   if any, must include the following acknowledgment:
      "This product includes software developed by the
       Apache Software Foundation (http://www.apache.org/)."
   Alternately, this acknowledgment may appear in the software itself,
   if and wherever such third-party acknowledgments normally appear.

4. The names "XML-RPC" and "Apache Software Foundation" must
   not be used to endorse or promote products derived from this
   software without prior written permission. For written
   permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache",
   nor may "Apache" appear in their name, without prior written
   permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
==

This software consists of voluntary contributions made by many
individuals on behalf of the Apache Software Foundation. For more
information on the Apache Software Foundation, please see
<http://www.apache.org/>.

7.3 Apache License, Version 2.0

Apache License

Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions

to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices

stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don’t include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

7.4 BSD License

Copyright (c) <YEAR>, <OWNER>
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above
* copyright notice, this list of conditions and the following
* disclaimer in the documentation and/or other materials provided
* with the distribution.

* Neither the name of the <ORGANIZATION> nor the names of its
* contributors may be used to endorse or promote products derived
* from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7.5 Freemarker License

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.

2. The end-user documentation included with the redistribution, if any, must
   include the following acknowlegement:
     "This product includes software developed by the
     Visigoth Software Society (http://www.visigoths.org/)."
   Alternately, this acknowlegement may appear in the software itself, if and
   wherever such third-party acknowlegements normally appear.

3. Neither the name "FreeMarker", "Visigoth", nor any of the names of the
   project contributors may be used to endorse or promote products derived
   from this software without prior written permission. For written
   permission, please contact visigoths@visigoths.org.

4. Products derived from this software may not be called "FreeMarker" or
   "Visigoth" nor may "FreeMarker" or "Visigoth" appear in their names
   without prior written permission of the Visigoth Software Society.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
VISIGOTH SOFTWARE SOCIETY OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7.6 Quartz License

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.