εχTEX

Developer's Guide

Version 1.7

Gerd Neugebauer

Michael Niedermair

With contribtions of Bastien Roucaries

This document describes some basic steps to develop and test εχTEX. It is meant for newcomers to the project or people who want to evaluate εχTEX by inspecting the sources.

Gerd Neugebauer
Im Lerchelsböhl 5
64521 Groß-Gerau (Germany)
gene@gerd-neugebauer.de

Michael Niedermair m.g.n@gmx.de
Bastien Roucaries roucaries.bastien@gmail.com

Contents

1. Introduction

εχTeX aims at providing a high-quality typesetting system. The development of εχTeX has been inspired by the experiences with TeX. The focus lies on an open design and a high degree of configurability. Thus εχTeX should be a good base for further development.

On the other hand we have to take care not to leave the current user base of TeX behind. pdfTeX has taught us that a migration path from TeX has a positive value in it. In the mean time the majority of TeX users applies in fact pdfTeX.

To provide a backward compatibility of εχTeX with TeX one special configuration is provided. Thus backward compatibility is just a matter of configuration.

1.1. Audience

This document is meant for developers and those interested in the sources and development processes of εχTeX. It should contain all information for getting started quickly.

1.2. Mailing Lists

If you are ready to try εχTeX you might as well want to join a mailing list to get in contact with the community. The following mailing lists might be of interest:

extex@dante.de
: A general mailing list about εχTeX. It has low traffic and is mainly in German. Subscribe and unsubscribe via the Web form `http://www.dante.de/listman/extex`.

extex-eng@dante.de
: A general mailing list about εχTeX. It has currently very low traffic and is in English. Subscribe and unsubscribe via the Web form `http://www.dante.de/listman/extex-eng`.

extex-devel@dante.de
: A mailing list for the exchange of the developers of εχTeX. It has low traffic and is partly in German. Subscribe and unsubscribe via the Web form `http://www.dante.de/listman/extex-devel`.

extex-cvs@list.berlios.de
: A mailing list for automatic notification about changes in the CVS repository of

ε_χTEX. It is not meant to post mails on this list. This list is not archived. Subscribe and unsubscribe via the Web form https://lists.berlios.de/mailman/listman/extex-cvs. You need to be logged in at Berlios when registering.

extex-bugs@list.berlios.de
A mailing list for bug messages of ε_χTEX. Subscribe and unsubscribe via the Web form http://lists.berlios.de/mailman/listman/extex-bugs.

1.3. Organizational Agreements

The developers of ε_χTEX have agreed on some rule for cooperation. Those rules are documented here.

1.3.1. Language

The official project language for ε_χTEX is English in the US dialect. The sources are documented in this language and the major documents are written in this language.

Since some of the developers are German this language might slip in during intensive discussions.

1.3.2. Maintainers of Files

Each file has a single maintainer – even if there are several authors. The maintainer has to be informed and has to agree on any changes in the file. The maintainership is usually indicated in the Java sources with the help of te tag `@author`. The first author is always the maintainer.

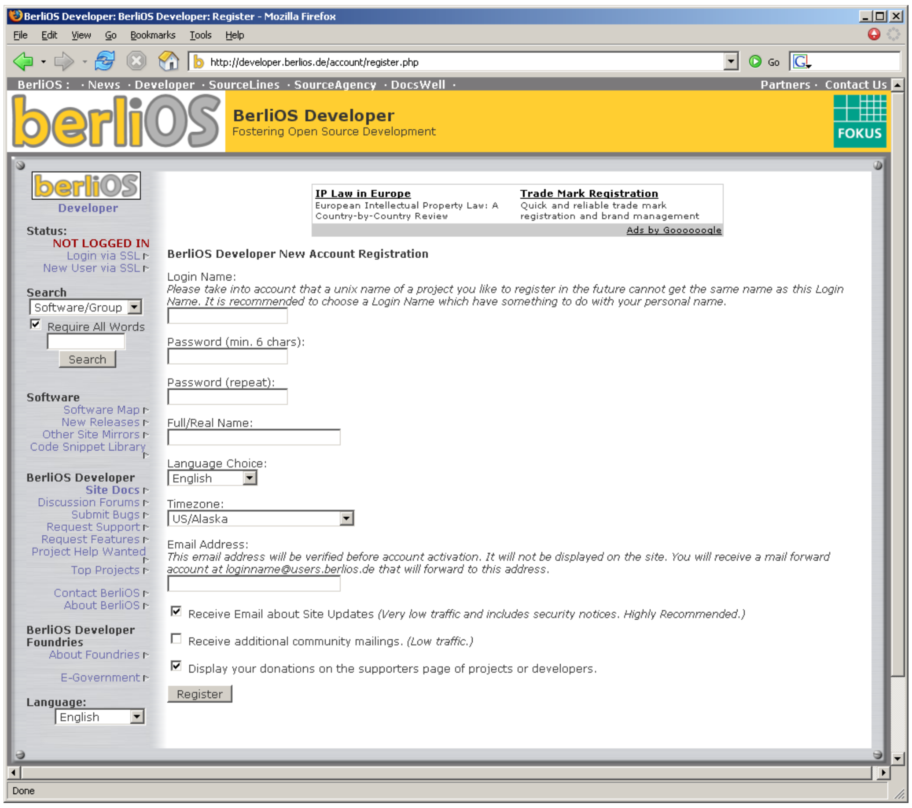
Changes to a file can be carried out by the maintainer or delegated to somebody else. The maintainer can change if both the old and new maintainer agree in this.

2. Prerequisites

2.1. User Account at Berlios

To commit changes to the repository you have to be enlisted as a developer for ε_{χ}TEX. A first requirement for this is an account at Berlios – the hosting site. If you just want to read the sources then you can use anonymous access.

To register at Berios use the page http://developer.berlios.de/ and select the item on the upper left side. You will find yourself in the registration page as shown in figure 2.1. You will find your way through easily.



BerliOS Developer New Account Registration

Login Name:
Please take into account that a unix name of a project you like to register in the future cannot get the same name as this Login Name. It is recommended to choose a Login Name which have something to do with your personal name.

Password (min. 6 chars):

Password (repeat):

Full/Real Name:

Language Choice:
English

Timezone:
US/Alaska

Email Address:
This email address will be verified before account activation. It will not be displayed on the site. You will receive a mail forward account at loginname@users.berlios.de that will forward to this address.

☑ Receive Email about Site Updates *(Very low traffic and includes security notices. Highly Recommended.)*

☐ Receive additional community mailings. *(Low traffic.)*

☑ Display your donations on the supporters page of projects or developers.

Register

Figure 2.1.: New Account at Berlios

When you have an account at Berlios you might be added to the developers list of ε_{χ}TEX. This as to be done by one of the admins of ε_{χ}TEX.

2.2. Java

You need to have Java 1.4.2 or later installed on your system. You can get Java for a several systems directly from java.sun.com. Download and install it according to the installation instructions for your environment.

To check that you have an appropriate Java on your path you can use the command `java` with the argument `-version`. This can be seen in the following listing:

```
# java -version
java version "1.4.2_09"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_09-b03)
Java HotSpot(TM) Client VM (build 1.4.2_09-b03, mixed mode)
#
```

Free Java implementations are currently not supported. They might work, but the last time we checked it, GCJ didn't suffice. We would be happy to have someone working on a compatibility layer for a free Java implementation.

2.3. TEXMF

If you want to use more than the pure ε_χTEX engine, fonts and macros can be inherited from a texmf tree. ε_χTEX itself does not contain a full texmf tree. It comes just with some rudimentary files necessary for testing. Thus you should have installed a texmf tree, e.g. from a TEXLive installation. This can be found on the Comprehensive TEX Archive Network (CTAN).

There is no need to install the texmf tree in a special place. You have to tell ε_χTEX anyhow where it can be found. It is even possible to work with several texmf trees.

One requirement for the texmf trees is that they have a file database (`ls-R`). ε_χTEX can be configured to work without it, but then ε_χTEX is deadly slow. Thus you do not really want to try this alternative.

To use your texmf tree you should create a configuration file in your home directory. On a Unix system the home directory is stored the environment variable `$HOME`. On a Windows system the home directory is usually located under `C:\WINDOWS\Profiles\`. The configuration file must have the name `.extex` (a little intelligence test under Windows;-). It contains one line of the following form

```
texmf.path=/usr/lib/texmf
```

The value should point to the location of the texmf tree. If you have several texmf trees which need to be used you can put them into this attribute by separating them by a platform-specific separator. This separator is a colon (`:`) under Unix and a semicolon (`;`) under Windows.

2.4. CVS Client

You need a CVS client installed. In the simplest case this is the client build into the IDE Eclipse, or the command line version of cvs.

2.5. A Command Line Interpreter

For several tasks it is convenient to have a command line interpreter at hand. On Unix this can be the (bourne, bash,...) shell. On Windows we recommend the Cygwin suite which contains the bash.

2.6. Ant

Ant is used throughout the whole buld system. For mpst purposes it is sufficient to live with the files distributed. Nevertheless it might be necessary to have ant installed and on the path for special tasks.

2.7. Perl

Perl is used to create the wae site. Thus it has to be installed for this purpose. For the usual development it is not necesary to have Perl installed.

3. The Development Environment

There is no mandatory IDE for the development of εχTeX. Nevertheless in practice you can get good support if you stick to the development environment widely used within the εχTeX community. This is based on the Eclipse IDE.

3.1. Eclipse

Eclipse is a free IDE for Java and other programming languages. It also provides a framework for the development of own programs. But this is not needed for the εχTeX core. Currently the version 3.1 of Eclipse is used within the εχTeX development team.

eclipse
3.1
(c) Copyright Eclipse contributors and others. 2000, 2005. All rights reserved. Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S., other countries, or both. Eclipse is a trademark of the Eclipse Foundation, Inc.

Figure 3.1.: Eclipse

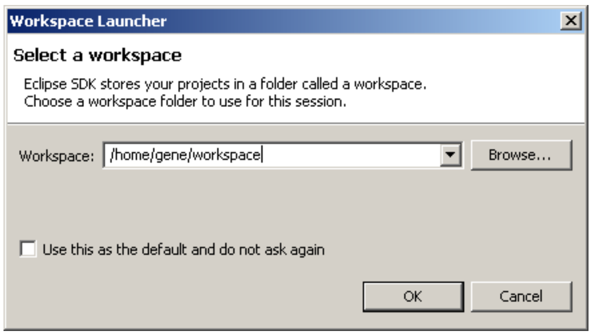
3.1.1. Eclipse Installation

Eclipse can be downloaded for free from `http://www.eclipse.org`. There you can get a file appropriate for your operating system containing the software development kit (SDK). For instance

eclipse-SDK-3.1-win32.zip
: for any decent Windows platform.

eclipse-SDK-3.1-linux-gtk.zip
: for Linux on Intel x86 with GTK.

Download the appropriate file and unpack it in the installation directory. A new subdirectory `eclipse` will be created containing all files of Eclipse. You are done with the basic installation. You can start the `eclipse` executable found in the just installed directory.


Workspace Launcher

Select a workspace

Eclipse SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /home/gene/workspace

Use this as the default and do not ask again

Figure 3.2.: Eclipse Workspace

When Eclipse starts you first see the splash screen shown in figure 3.1. Then Eclipse requests a workspace – as shown in figure 3.2. The workspace is a directory where the projects live and where your preferences are stored. If you have chosen the workspace directory carefully, you can turn on the check mark in this dialog to be not asked again.

Finally you end up in the welcome window of Eclipse shown in figure 3.3. Take some time and read the introductory material found there.

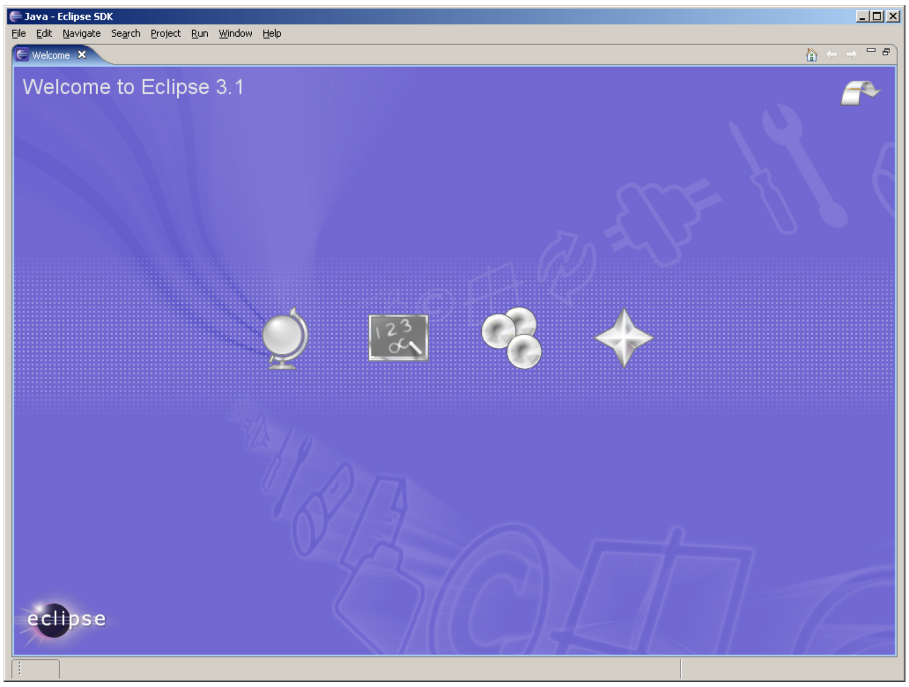


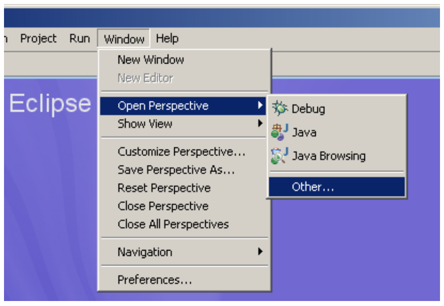
Figure 3.3.: Eclipse Initial Window

The following sections describe some of the configurations which should be performed in order to work with Eclipse on εχTeX.

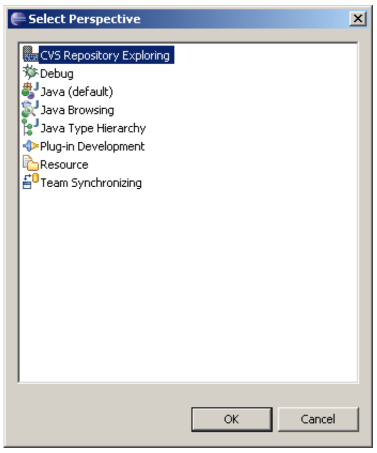
3.1.2. Downloading the Sources

Now we are ready to create a project for the sources of εχTeX. Everything needed can be found in the CVs repository of εχTeX hosted by Berlios.. Thus we start to get things onto the local host. For this purpose we need to open a new perspective in Eclipse. A perspective is a collection of windows which are usually meant for a common task.

A new perspective can be opened via the window Window → Open Perspective → Other... which can be seen in figure 3.4(a). This menu item opens a dialog box which offers some perspectives for opening. Currently we need a "CVS Exploring" perspective. This perspective is meant for inspecting CVS repositories and manipulation. Thus this perspective is selected (see figure 3.4(b)) and the dialog is completed with the OK button.



(a) Open Perspective



(b) Selecting "CVS Exploring Perspective"

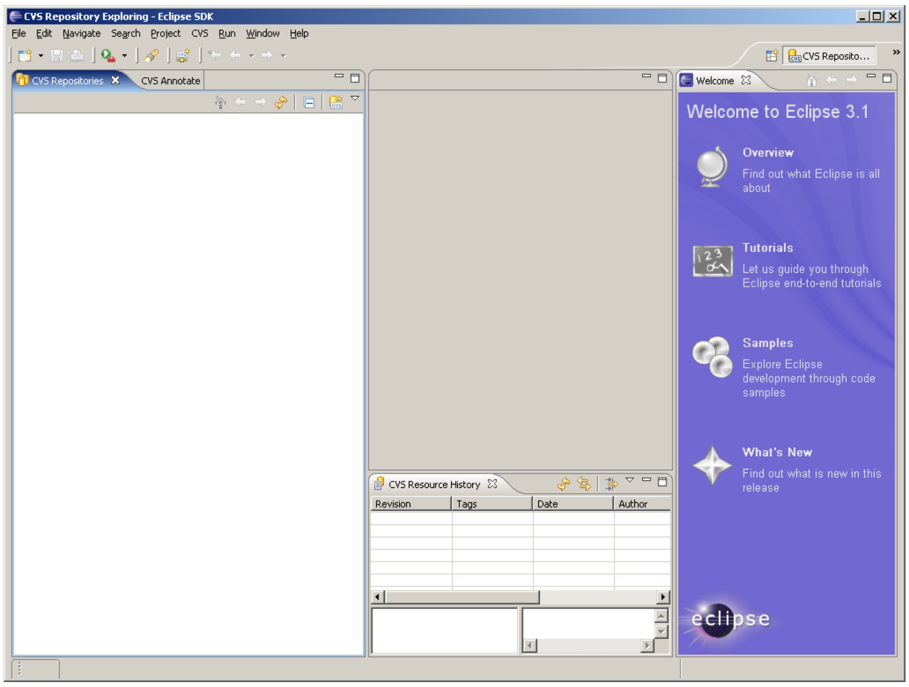
Figure 3.4.: Switching to a Perspective

Now a CVS exploring perspective is opened (see figure 3.5). You see a lot of windows and icons there. The tab "CVS Repositories" on the left side shows all repository locations currently known. This list is empty since we have not added any CVS locations yet.

To add a new repository location press the left mouse button on this tab and select New → Repository Location... (see figure 3.6(a)). This brings up the dialog shown in figure 3.6(b). Here you can enter the coordinates of the εχTeX CVS repository.

Note that you have to enter your account at Berlios and its password into the appropriate fields. If you do not have an account you can use the account name `anonymous` without any password to get reading access to the sources.

For this step you need online access to the internet. When the form is submitted with the OK button, the accessibility of the repository location is checked. Upon success the



Welcome to Eclipse 3.1

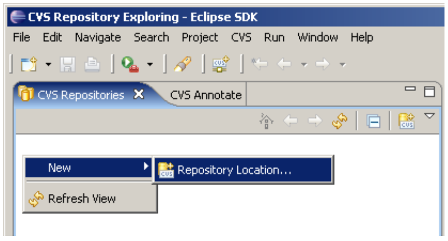
Overview
Find out what Eclipse is all about

Tutorials
Let us guide you through Eclipse end-to-end tutorials

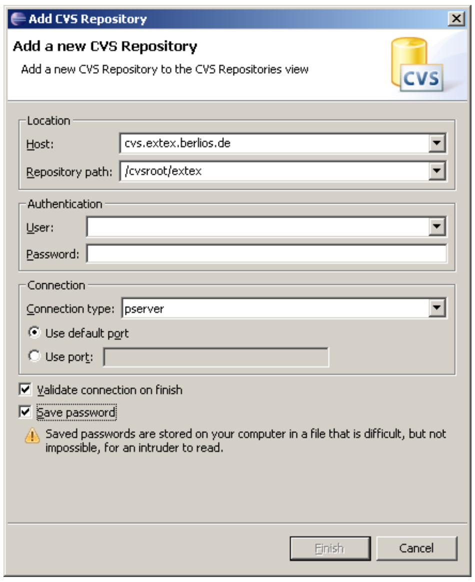
Samples
Explore Eclipse development through code samples

What's New
Find out what is new in this release

Figure 3.5.: CVS Exploring Perspective


CVS Repository Exploring - Eclipse SDK
File Edit Navigate Search Project CVS Run Window Help
CVS Repositories
CVS Annotate
New
Repository Location...
Refresh View

(a) New Repository Location



Add a new CVS Repository

Add a new CVS Repository to the CVS Repositories view

Location

Host: cvs.extex.berlios.de

Repository path: /cvsroot/extex

Authentication

User:

Password:

Connection

Connection type: pserver

(●) Use default port

( ) Use port:

[✓] Validate connection on finish

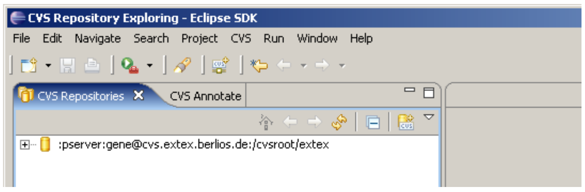
[✓] Save password

Saved passwords are stored on your computer in a file that is difficult, but not impossible, for an intruder to read.

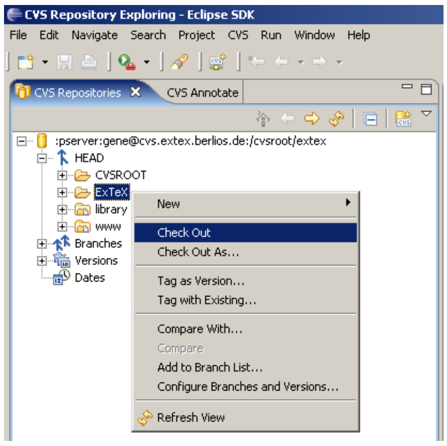
(b) The Coordinates of the ε_{χ}TEX CVS Repository

Figure 3.6.: Adding the ε_{χ}TEX CVS Repository

new repository location is added to the list of repository locations as can be seen in figure 3.7(a).


CVS Repository Exploring - Eclipse SDK
File Edit Navigate Search Project CVS Run Window Help
CVS Repositories
CVS Annotate
:pserver:gene@cvs.extex.berlios.de:/cvsroot/extex

(a) The ε_{χ}TEX Repository Listed



(b) Selecting to check-out of ε_{χ}TEX

Figure 3.7.: Checking-out of ε_{χ}TEX

The next step consists of the check-out of the sources into an Eclipse project. To accomplish this you have to open the repository location and the HEAD within. Right-click the item `ExTeX` in the list (see figure 3.7(a)) and select Checkout in the appearing context menu (see figure 3.7(b)). This will instruct Exclipe to create a new project in the workspace and fill it with the files from the repository.

Eclipse shows a progress bar during the check-out (see figure reffig:eclipse-checkout). This operation may take some time – we have been really busy creating files. When the checkout is finished you will find the project `ExTeX` in Eclipse containing the files within. The appearance of the Package View with those files is shown in figure 3.8(b).

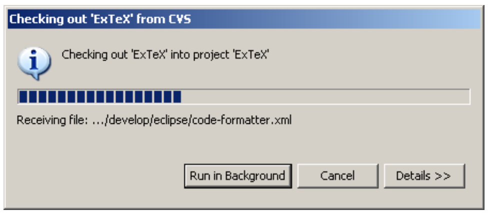
3.1.3. Configuring Eclipse

Eclipse can be configured in a wide range. In the following sections some configuration options are proposed for the seamless development of ε_{χ}TEX. The configuration is performed via the preferences dialog. This dialog can be opened via Window → Preferences...

This menu brings up a dialog with many tabs which can be used to adjust the behaviour of Eclipse in many ways. The first step described below consists of the adaption of the appearance of the text editors. In the tree view on the left side of the dialog select General → Editors → Text Editors as shown in figure 3.9(b).

Now you can adjust some values on the right side of the dialog. Set the tag width to 8. Check the item Show print margin. Adjust the print margin to 80. And finally change the print color to red. The settings are stored in the workspace by accepting the settings with the OK button.

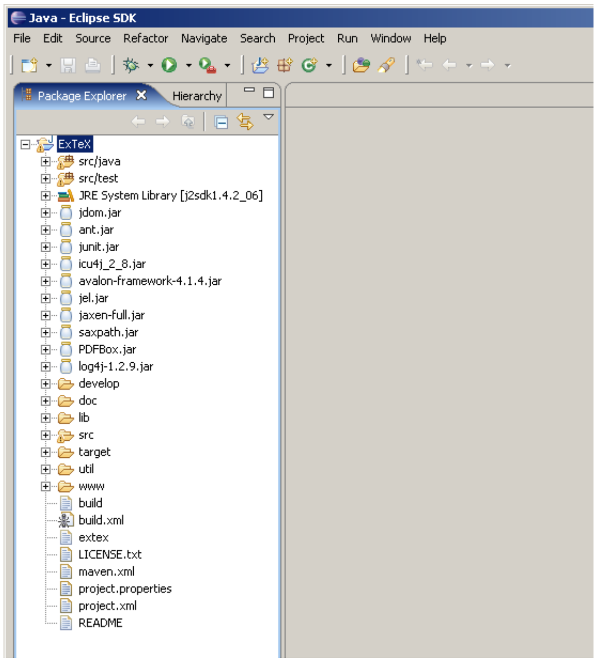
The rational is that the tabs should be used in the traditional sense of eight chracters wide. In fact this is just a fallback. Usually tabs should be avoided where possible. The


Checking out 'ExTeX' from CVS

Checking out 'ExTeX' into project 'ExTeX'

Receiving file: .../develop/eclipse/code-formatter.xml

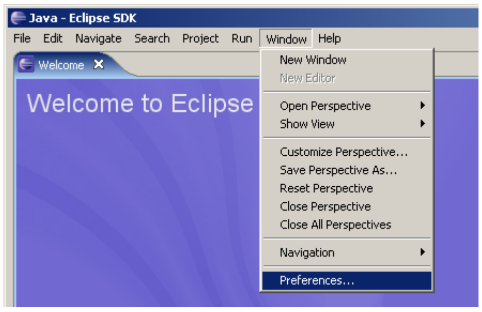
(a) The Checkout Progress Bar


Java - Eclipse SDK

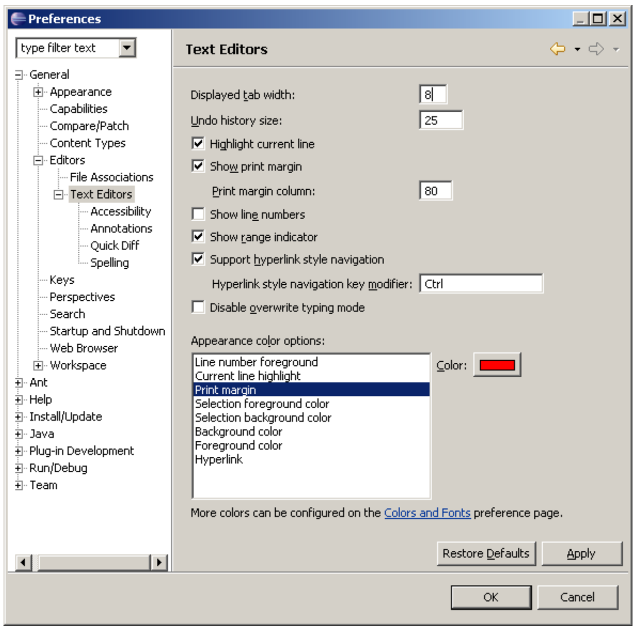
- ExTeX
  - src/java
  - src/test
  - JRE System Library [j2sdk1.4.2_06]
  - jdom.jar
  - ant.jar
  - junit.jar
  - icu4j_2_8.jar
  - avalon-framework-4.1.4.jar
  - jel.jar
  - jaxen-full.jar
  - saxpath.jar
  - PDFBox.jar
  - log4j-1.2.9.jar
  - develop
  - doc
  - lib
  - src
  - target
  - util
  - www
  - build
  - build.xml
  - extex
  - LICENSE.txt
  - maven.xml
  - project.properties
  - project.xml
  - README

(b) The ε_χTEX Project in the Package View

Figure 3.8.: Checking out ε_χTEX from the Repository


Java - Eclipse SDK

(a) Eclipse Preferences


Preferences

Text Editors

Displayed tab width: 8

Undo history size: 25

[✓] Highlight current line

[✓] Show print margin

Print margin column: 80

[ ] Show line numbers

[✓] Show range indicator

[✓] Support hyperlink style navigation

Hyperlink style navigation key modifier: Ctrl

[ ] Disable overwrite typing mode

Appearance color options:

Line number foreground
Current line highlight
Print margin
Selection foreground color
Selection background color
Background color
Foreground color
Hyperlink

Color: 🟥

More colors can be configured on the Colors and Fonts preference page.

(b) Eclipse Print Margin

Figure 3.9.: Some simple Settings

print margin of 80 is a weak rule. Try to limit yourself to this width. Sometimes it is not reasonable. Thus the checkstyle rules allow some more characters before complaining.

The following sections describe some more of the configuration options. You should really consider to follow the instructions to make maximal use of the configurations provided with εχTEX.

3.1.4. Code Templates

Code templates provide a convenient way of filling in a frame for the documentation whenever some code is generated by Eclipse. The εχTEX repository contains in the file `develop/eclipse/codetemplates.xml` some definitions of code templates. To import those definitions use the preferences (see figure 3.9(a)). Here select the item Java → Code Style → Code Templates. The button Import... can leads to a file selector where the file `develop/eclipse/codetemplates.xml` should be entered.

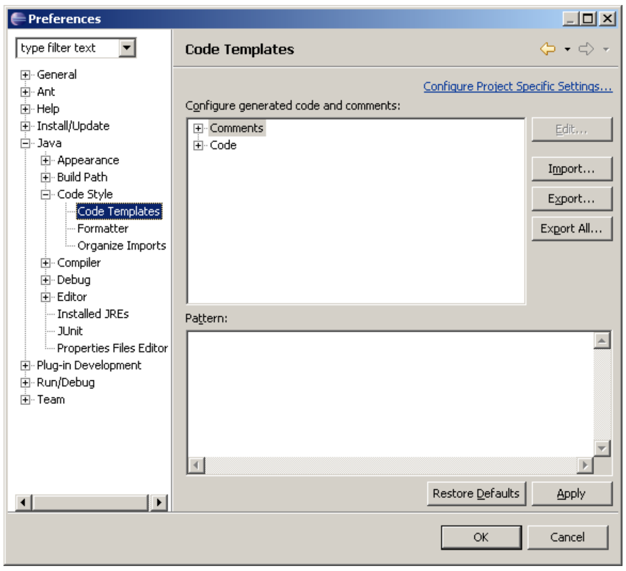
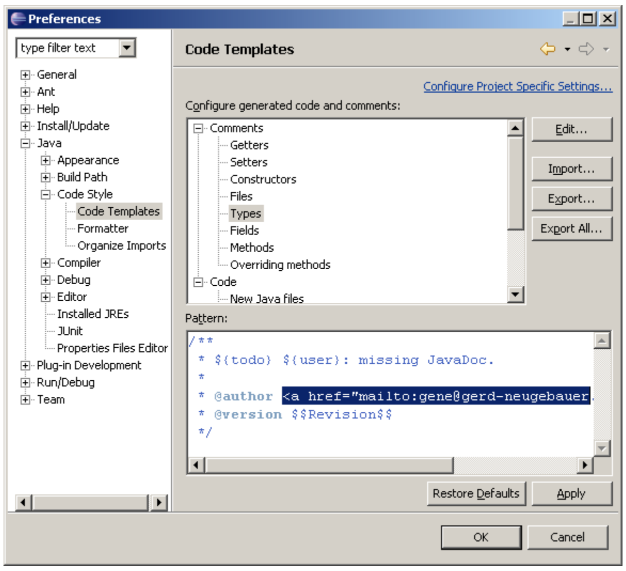


Figure 3.10.: Eclipse Preferences

After the code templates have been loaded a minor adaption is required. The entry under the key Comments → Types containes hard-wired a name and email address of the author. Here the own name and email address should be entered (see figure fig:eclipse-template-author).

3.1.5. The Code Formatter

Eclipse comes has a code formatter which can be invoked easily. This code formatter can be configured for different needs. A configuration for εχTEX is contained in the repository under `develop/eclipse/formatter.xml`. In Eclipse the preference page can be found under the key Java → Code Style → Formatter. ere you can use the button



Configure generated code and comments:

- Comments
  - Getters
  - Setters
  - Constructors
  - Files
  - Types
  - Fields
  - Methods
  - Overriding methods
- Code
  - New Java files

Pattern:

```
/**
 * ${todo} ${user}: missing JavaDoc.
 *
 * @author <a href="mailto:gene@gerd-neugebauer.
 * @version $$Revision$$
 */
```

Figure 3.11.: Code Template Author Name

Import... and select the configuration file. Now the profile “gene” is loaded and can be selected. This is shown in figure 3.12.

The formatter for Ant files has distinct parameters which should be adapted. The Prference page can be found under the key Ant → Editor → Formatter. The values should be adjusted as shown in figure 3.13.

3.1.6. Checkstyle

Checkstyle is a tool for checking the adherence of Java source code to certain rules. The rules can be freely configured. The ε_{χ}TEX repository contains a set of rules for checkstyle.

Checkstyle comes in a command line version and as a plug-in for Eclipse. This plugin has to be installed first.

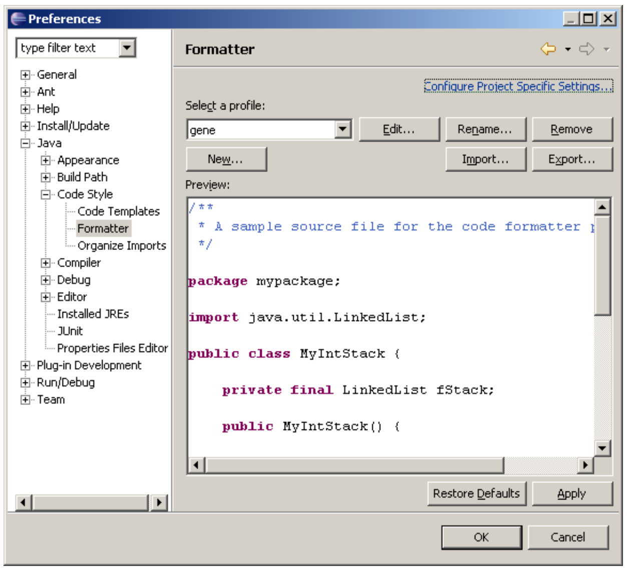
Install Checkstyle plugin

To install the checkstyle plugin over the update wizard, use Help → Software Update → Find and Install → Search for new features to install → Next → New Remote Site and input the name ’checkstyle’ and the URL `http://eclipse-cs.sourceforge.net/update` (see figure 3.14).

Configure Checkstyle

To set the configuration use Window → Preferences → Checkstyle . Create a new configuration and set the values in figure 3.15.

The configuration is stored in `develop/eclipse/extex_checkstyle.xml`.

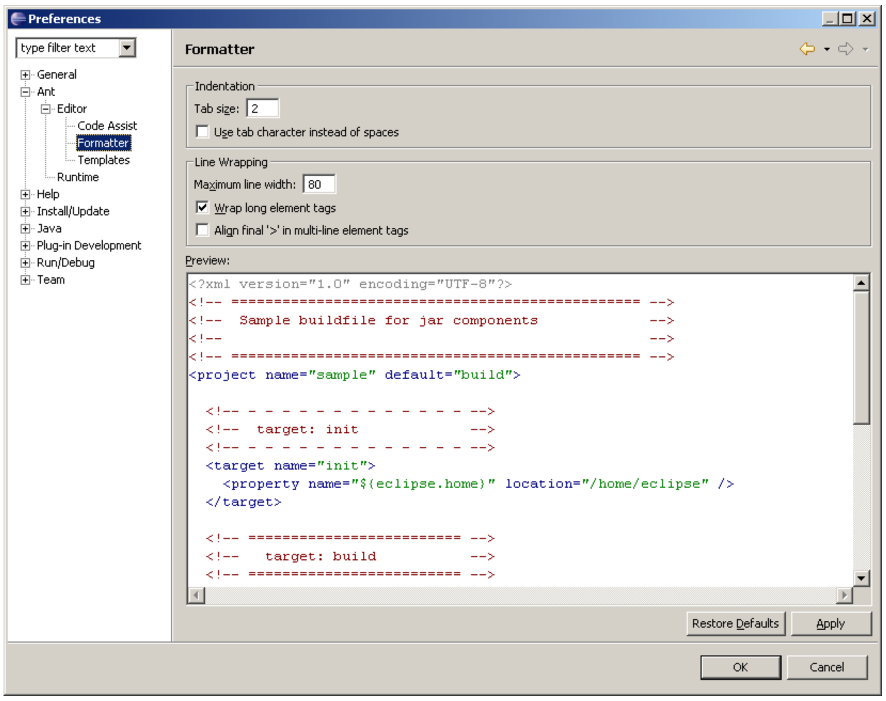


Select a profile: gene

Preview:

```
/**
 * A sample source file for the code formatter
 */

package mypackage;

import java.util.LinkedList;

public class MyIntStack {

    private final LinkedList fStack;

    public MyIntStack() {
```

Figure 3.12.: Settings for the Code Formatter



Indentation

Tab size: 2

☐ Use tab character instead of spaces

Line Wrapping

Maximum line width: 80

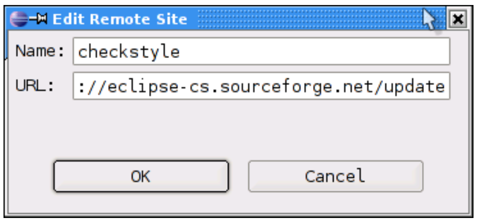
☑ Wrap long element tags

☐ Align final '>' in multi-line element tags

Preview:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ================================================ -->
<!--  Sample buildfile for jar components             -->
<!--                                                  -->
<!-- ================================================ -->
<project name="sample" default="build">

  <!-- - - - - - - - - - - - - - - - - - -->
  <!--  target: init                    -->
  <!-- - - - - - - - - - - - - - - - - - -->
  <target name="init">
    <property name="${eclipse.home}" location="/home/eclipse" />
  </target>

  <!-- ========================= -->
  <!--   target: build           -->
  <!-- ========================= -->
```

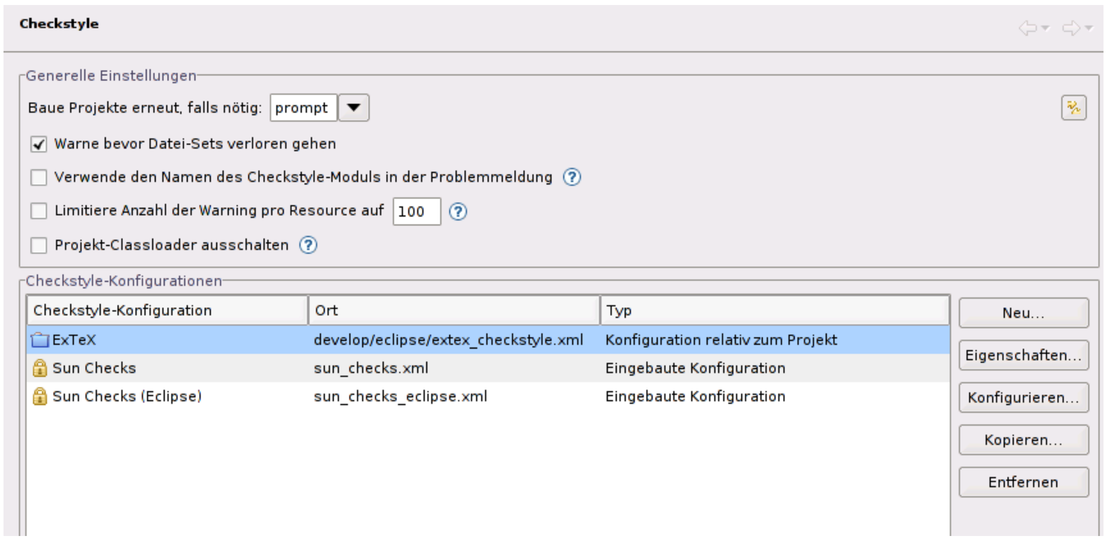
Figure 3.13.: Settings for the Ant Formatter



Name: checkstyle

URL: ://eclipse-cs.sourceforge.net/update

Figure 3.14.: Checkstyle URL



Baue Projekte erneut, falls nötig: prompt

[✓] Warne bevor Datei-Sets verloren gehen

[ ] Verwende den Namen des Checkstyle-Moduls in der Problemmeldung

[ ] Limitiere Anzahl der Warning pro Resource auf 100

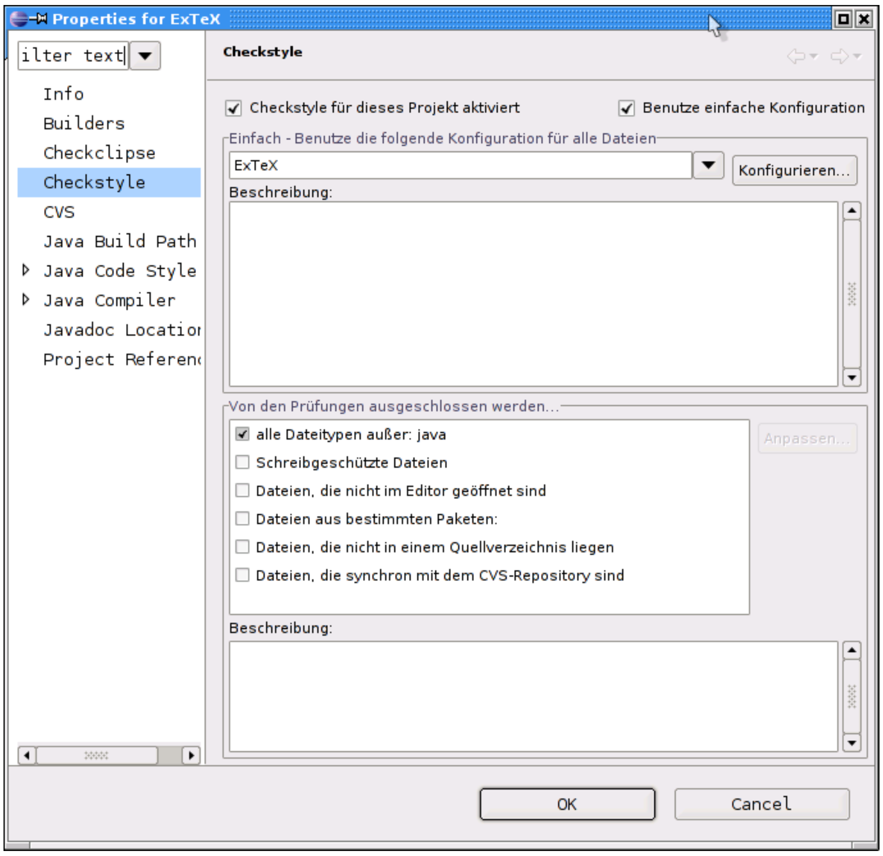
[ ] Projekt-Classloader ausschalten

Checkstyle-Konfiguration	Ort	Typ
ExTeX	develop/eclipse/extex_checkstyle.xml	Konfiguration relativ zum Projekt
Sun Checks	sun_checks.xml	Eingebaute Konfiguration
Sun Checks (Eclipse)	sun_checks_eclipse.xml	Eingebaute Konfiguration

Figure 3.15.: Checkstyle configuration

Enable Checkstyle

To enable the checks set in Project → Properties → Chekcstyle the configuration to *ExTeX* (see figure 3.16).



Checkstyle

✓ Checkstyle für dieses Projekt aktiviert
✓ Benutze einfache Konfiguration

Einfach - Benutze die folgende Konfiguration für alle Dateien
ExTeX
Beschreibung:

Von den Prüfungen ausgeschlossen werden...
✓ alle Dateitypen außer: java
☐ Schreibgeschützte Dateien
☐ Dateien, die nicht im Editor geöffnet sind
☐ Dateien aus bestimmten Paketen:
☐ Dateien, die nicht in einem Quellverzeichnis liegen
☐ Dateien, die synchron mit dem CVS-Repository sind
Beschreibung:

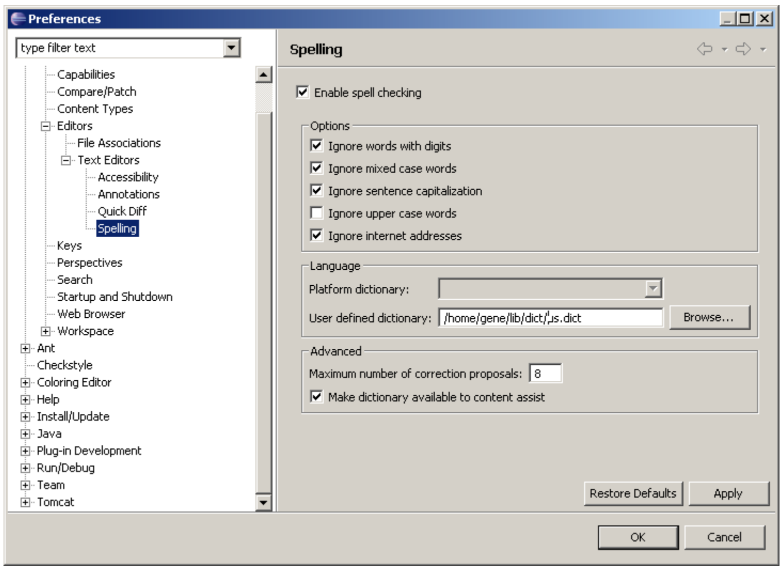
Figure 3.16.: enable Checkstyle

3.1.7. Spelling

Since English is not the native language of each developer it is a good idea to enable the spell checking of the source code. This feature is provided by Eclipse. In figure reffig:eclipse-spelling you can seen the preference page where you can activate the spell checking and provide a dictionary.

A dictionary can be got from SCOWL (`http://wordlist.sourceforge.net/`). You might want to use the US dictionary of medium size. Since this contains enough words to fit but not too much obscure words which hide typos.

After the spell checking is activated potential typos are marked in the editor with yellow lines. Correction proposals can be requested with the quick fix shortcut Ctrl-1.



Spelling

- [✓] Enable spell checking

Options
- [✓] Ignore words with digits
- [✓] Ignore mixed case words
- [✓] Ignore sentence capitalization
- [ ] Ignore upper case words
- [✓] Ignore internet addresses

Language
- Platform dictionary:
- User defined dictionary: /home/gene/lib/dict/us.dict Browse...

Advanced
- Maximum number of correction proposals: 8
- [✓] Make dictionary available to content assist

Figure 3.17.: Spelling Preferences

3.1.8. Compiling εχTEX

Any source file in Eclipse is compiled automatically when the file is saved. Thus it is usually not necessary to compile things manually. If you feel the need to recompile everything you can achieve this by selecting Project → Clean... while the item Project → Build Automatically is checked (see figure 3.18).

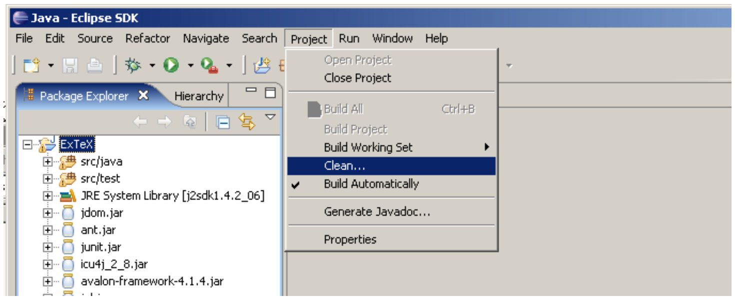


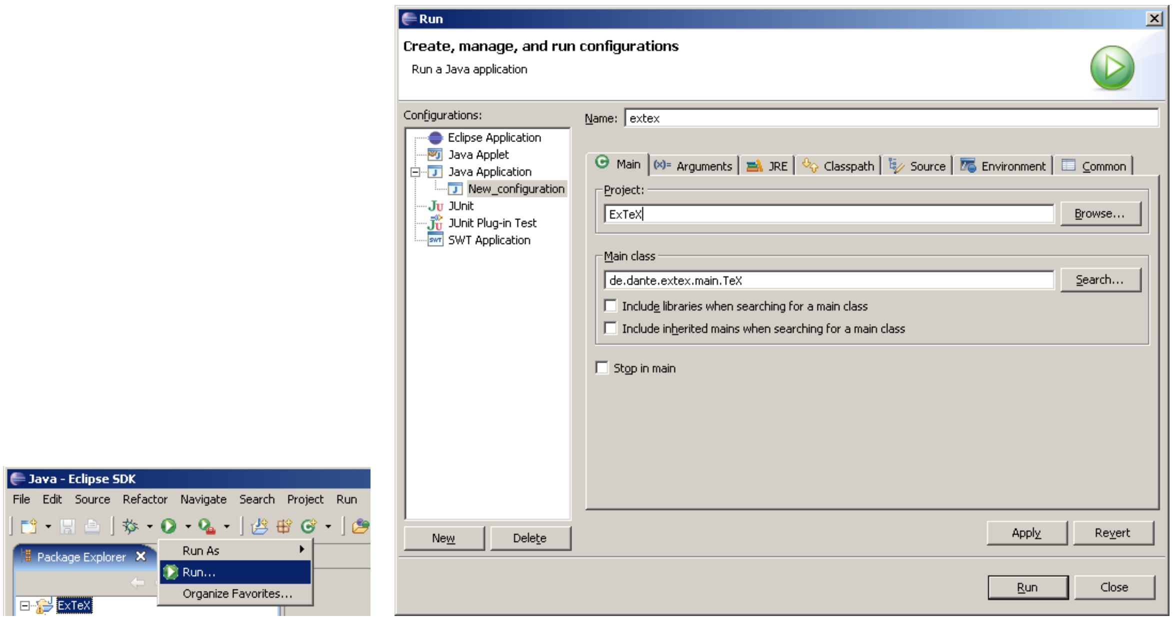
Figure 3.18.: Recompiling a Project

Another recompilation can be triggered via the Ant task `compile`.

3.1.9. Running εχTEX

εχTEX can be run from within Eclipse. We will describe here the execution of the compiled sources from a workspace. The executio of an external program would be an alterative. But this is only of minor relevance for a developer.

To run εχTEX on some input file you have to create a run profile. The profile is kept and can be used the next time again. To create a run profile select the toolbar item Run... in the Java perspective (see figure 3.19(a)). In the appearing dialog select Java Application and press the button New. Now you can fill in the tabs as seen in figure 3.19(b). Enter a name, the project and the main class. The main class to use is `de.dante.extex.main.TeX`.



Name: extex
Project: ExTeX
Main class: de.dante.extex.main.TeX
☐ Include libraries when searching for a main class
☐ Include inherited mains when searching for a main class
☐ Stop in main

(a) The Run Menu (b) Creating a run configuration

Figure 3.19.: Checking out εχTEX from the Repository

On the Arguments tab you can enter the arguments for the invocation of εχTEX. These are the same arguments which can also be used on the command line. Usually here the input file is given (see figure 3.20.

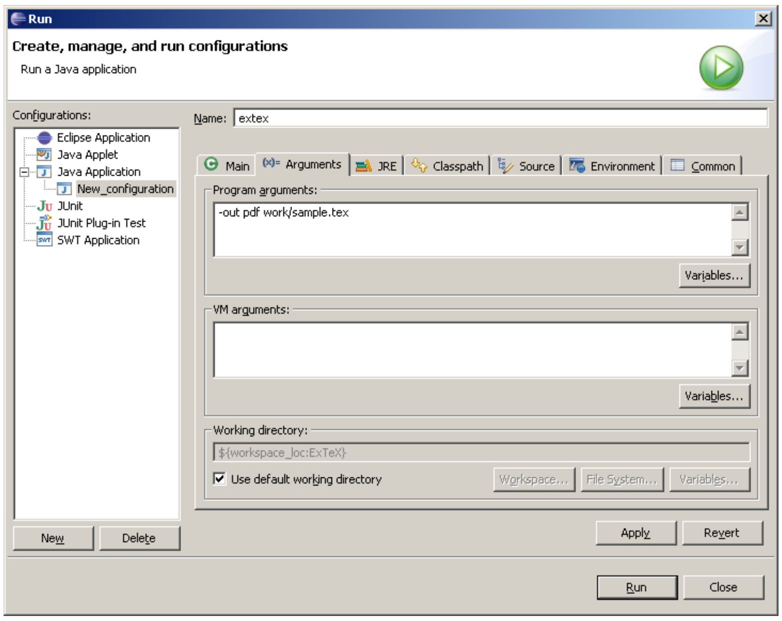
The Run button submits the command. A Console view is opened which can be used to interact with the the program – like in the command line interpreter.

3.1.10. Committing Changes

Eclipse ships with a CVS plugin which hides the details of the underlying version control system. Thus things are quite simple for the newcomers. On the other hand they are differnt from the procedure on the command line or other tools whic mimic the command line (like WinCVS or TortoiseCVS).

The metaphor used in Eclipse is the synchronisation of the workspace with the repository. In the course of this syncronisation changes in the workspace files are committed to the repository, changes from the repository are updated into the workspace, and conflicts can be resolved. The conflict resolution – also known as merging – is the demanding task. Thus it has to be performed by a human.

To start the synchronisation select in the Package Manager or Navigator view the topmost εχTEX node and activate in the context menu (right mouse button) the entry


Run

Create, manage, and run configurations

Run a Java application

Name: extex

Program arguments: -out pdf work/sample.tex

VM arguments:

Working directory: ${workspace_loc:ExTeX}

☑ Use default working directory

Figure 3.20.: Arguments for Running ε_{χ}TeX

Team → Synchronize with Repository (see figure 3.21).

To be completed.

3.1.11. Running Ant from within Eclipse

To use Ant from within Eclipse you have to open the Ant view. This can be acomplished via the menu Window → Show View → Ant (see figure 3.22(a)).

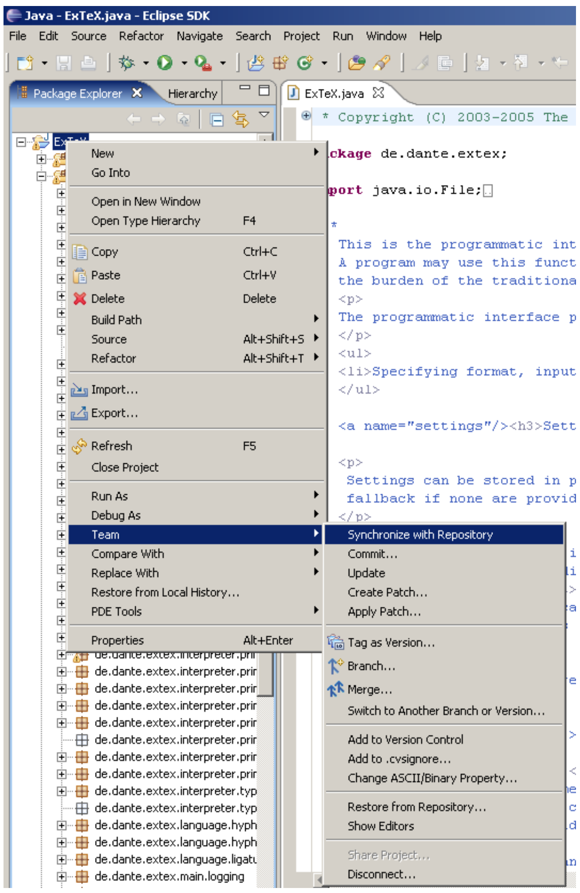
In this view use the leftmost tool to add an Ant file. In the file selector choose the file `ExTeX/build.xml`. The Ant file is added to the (previously empty) list. It can be open to show the Ant target available (see figure 3.22(b)).

A double click on a target starts it's execution. The output is shown in a Console view.

A description of the targets can be found in section 3.2.3.

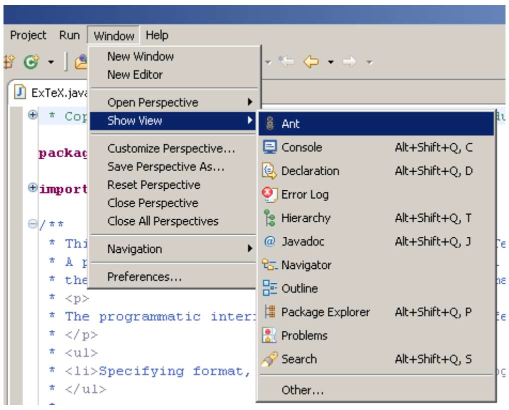
3.1.12. Creating Javadoc

To create the Javadoc HTML description of the sources you can use the Ant target `javadoc`. See sections 3.1.11 and 3.2.3. The result can be found in the directory `target/javadoc`.



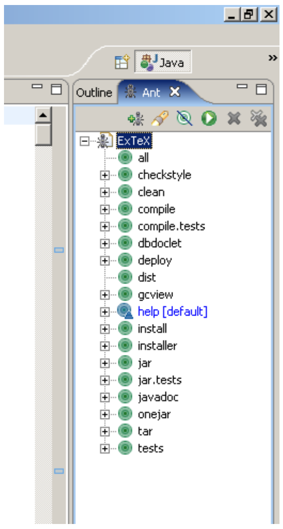
```
 * Copyright (C) 2003-2005 The
ckage de.dante.extex;
port java.io.File;
 This is the programmatic int
 A program may use this funct
 the burden of the traditiona
 <p>
 The programmatic interface p
 </p>
 <ul>
 <li>Specifying format, input
 </ul>
 <a name="settings"/><h3>Sett
 <p>
  Settings can be stored in p
  fallback if none are provid
 </p>
```

Figure 3.21.: Starting Synchronization



```
* Cop
packag
import
/**
 * Thi
 * A p
 * the
 * <p>
 * The programmatic inter
 * </p>
 * <ul>
 * <li>Specifying format,
 * </ul>
```

(a) Opening an Ant View



(b) The Ant View for ε_χTeX

Figure 3.22.: Ant in Eclipse

3.1.13. Creating the Installer

To create the inastaller you can use the Ant target `installer`. See sections 3.1.11 and 3.2.3. The result can be found in the file `target/ExTeX-setup.jar`.

3.2. Command Line Use

3.2.1. Downloading the Sources

The sources of εχTEX are stored in a RCS repository. To access this repository you need access to the internet and RCS installed in some way.

The coordinates of the repository are:

Connection type:	`pserver`
User:	`anonymous`
Host:	`cvs.extex.berlios.de`
Location:	`/cvsroot/extex`
Module:	`ExTeX`

These coordinates allow you anonymous access to the sources with reading permissions only.

You need to download the sources of εχTEX. On the command line this can be done with the following commands:

```
# cvs -d:pserver:anonymous@cvs.extex.berlios.de:/cvsroot/extex login
# cvs -z3 -d:pserver:anonymous@cvs.extex.berlios.de:/cvsroot/extex co ExTeX
```

If you want to participate in the development and are enlisted at Berlios you should use your account and password instead of the anonymous account.

3.2.2. Checkstyle

Checkstyle is a source code checker. εχTEX should show a homogeneous appearance of the sources. Thus certain rules should be followed. Some of the rules are checked by the following command:

```
# build checkstyle
```

The result can be found in the file `target/checkstyle.txt`.

3.2.3. Ant

Apache Ant (`http://ant.apache.org`) is a build system for Java. It can be considered state of the art for Java programs to come with Ant scripts. εχTEX supports Ant by providing a `build.xml` file for various tasks.

The files needed for running Ant are included in the ε_χTEX repository. Thus no additional installation is required. Just some setting need to be performed before Ant can be used.

An environment variable `JAVA_HOME` should be defined which points to the JDK. The following jars should be placed on the environment variable `CLASSPATH`:

- `$JAVA_HOME/lib/tools.jar`
- `$JAVA_HOME/lib/classes.zip`
- and all jars found in `develop/lib`

The Ant can be invoked like in

```
#  $JAVA_HOME/bin/java -Dant.home=$ANT_HOME org.apache.tools.ant.Main compile
Buildfile: build.xml

compile:

BUILD SUCCESSFUL
Total time: 1 second
#
```

These steps are performed by the shell script `build` in the ε_χTEX directory. Thus you can achieve the same effect – without any preparations except setting `JAVA_HOME` – with the following command:

```
#  build compile
Buildfile: build.xml

compile:

BUILD SUCCESSFUL
Total time: 1 second
#
```

The Ant configuration can be found in the file `build.xml` in the ε_χTEX main directory. This configuration contains at least the following targets:

all This target builds nearly everything.

compile Tis target compiles all Java files of the sources into the directora `target/classes`. Note, that the test classess are not compiled. See also section 3.2.4.

jar This target arranges that the file `target/extex.jar` is created. It contains the compiled sources of ε_χTEX.

javadoc This target creates the Javadoc HTML files in the directory `target/javadoc`. See also section 3.2.7.

checkstyle This target applies checkstyle and creates a report in `target/checkstyle.txt`.

tests This target aplies all JUnit test cases. See also section 3.2.6.

installer This target creates the installer with the graphical user interface. The result is placed in the file `target/ExTeX-setup.jar`. See also section 3.2.8.

clean This target deletes some generated files.

3.2.4. Compiling εχTEX

Compiling εχTEXform the command line can be accomplished with the help of the build script. The build script is a wrapper around Ant. It can be invoked with the following command:

```
#  build compile
Buildfile: build.xml

compile:

BUILD SUCCESSFUL
Total time: 1 second
#
```

The generated files are placed in the sub-directory `target/classes`. Thus if this directory and the jars in `lib` are on the class path then εχTEX can be run immediately.

3.2.5. Running εχTEX

εχTEX can be run with the help of the εχTEX script in the main directory or by a direct invocation of Java. The start script is provided for Unix under the name `extex` and for Windows under the name `extex.bat`.

```
#  extex work/empty.tex
This is ExTeX, Version 0.0 (ExTeX mode)
(work/empty)
No pages of output.
Transcript written on ./empty.log.
#
```

For the usual purposes these scripts can be used as a plug-in replacement for TEX. See the User's Guide for the command line options.

To run εχTEX from the command line prepare the class path – i.e. the environment variable `CLASSPATH` – to contain all libraries found in the directory `lib`. In addition the directory `target/classes` have to be on the class path. Then you can invoke εχTEX like in the following example:

```
#  java de.dante.extex.Main.TeX work/empty
This is ExTeX, Version 0.0 (ExTeX mode)
(work/empty)
No pages of output.
Transcript written on ./empty.log.
#
```

The command line arguments are the same as for `extex` mentioned above.

3.2.6. JUnit

JUnit is the state of the art concerning test automation for Java programs. Thus ε_χTEX provides some test cases in the form of JUnit classes.

All test can be run from the command line with the build script:

```
#  build tests
Buildfile: build.xml

compile:

jar:
      [jar] Building jar: /home/gene/src/ExTeX/target/lib/extex.jar

compile.tests:
     [copy] Copying 148 files to /home/gene/src/ExTeX/target/classes

jar.tests:
      [jar] Building jar: /home/gene/src/ExTeX/target/lib/testsuite.jar

tests:
    [mkdir] Created dir: /home/gene/src/ExTeX/tmp/tests
    [junit] Running de.dante.etex.CurrentgrouplevelTest
    [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 1,623 sec
...

BUILD SUCCESSFUL
Total time: 3 minutes 54 seconds
#
```

This invocation runas all JUnit test cases found in the directory `src/test`. The results can be found in the directory `target/tests` with one file per test class.

To run single cases or a selected set of test cases you can use the parameter `cases`. This parameter is added to the command line arguments with the prefix `-D`. The value follows after an equals sign. The value is a pattern to select the test case files to be used. The pattern `**` denotes an arbitrary deep directory hierarchy. `*` denotes an arbitrary sequence of characters. Note that the pattern should end in `Test.java`.

```
#  build tests -Dcases=**/RelaxTest.java
Buildfile: build.xml

compile:

jar:

compile.tests:

jar.tests:

tests:
   [delete] Deleting directory /home/gene/src/ExTeX/target/tests
    [mkdir] Created dir: /home/gene/src/ExTeX/target/tests
    [junit] Running de.dante.extex.interpreter.primitives.RelaxTest
    [junit] Tests run: 5, Failures: 0, Errors: 0, Time elapsed: 1,062 sec

BUILD SUCCESSFUL
Total time: 5 seconds
#
```

Details on testing and test cases can be found in section 5.

3.2.7. Creating Javadoc

Creating the Javadoc HTML pages can best be complished with the help of the build script. Here the target `javadoc` does everythign necessary:

```
#  build javadoc
Buildfile: build.xml

javadoc:
  [javadoc] Generating Javadoc
  [javadoc] Javadoc execution
  [javadoc] Loading source files for package de.dante.extex...
  [javadoc] Loading source files for package de.dante.extex.color...
  [javadoc] Loading source files for package de.dante.extex.color.model...
...

BUILD SUCCESSFUL
Total time: 2 minutes 20 seconds
#
```

As the result of this invocation the Javadoc HTML pages are stored in the subdirectory `target/javadoc`.

3.2.8. Creating the Installer

The installer can be build with the help of the build script. The invocation looks as follows:

```
#  build installer
Buildfile: build.xml

compile:

jar:
      [jar] Building jar: /home/gene/src/ExTeX/target/lib/extex.jar

installer:
   [izpack] Adding resource : IzPack.uninstaller ...
   [izpack] Setting the installer informations ...
   [izpack] Setting the GUI preferences ...
...
   [izpack] Writing Packs ...
   [izpack] Writing Pack #0 : Core
   [izpack] Writing Pack #1 : Libraries
   [izpack] Writing Pack #2 : User Settings
   [izpack] Writing Pack #3 : Documentation
   [izpack] Writing Pack #4 : Fonts
   [izpack] Writing Pack #5 : Sources

BUILD SUCCESSFUL
Total time: 1 minute 30 seconds
#
```

After the work is complete the installer can be found in the file `ExTeX-setup.jar` in the directory `target`. The use of the installer is described in the User's Manual.

Alternatively the installer can also be created with the Ant task `installer`. Using this method can be applied from the command line and from within Eclipse.

Note that the installer is automatically created once a day and provided in the snapshot directory of the ε_{χ}TEX Web site.

3.3. Use with Emacs and JDEE

JDEE is the extension of Emcas for the development with Java. ε_{χ}TEX contais soem support files for use in this context.

To be completed.

3.4. Modelling: Jude

Jude is a UML modeller written in Java. It is distributed in a community edition for free use. Currently the version 1.6.2 is available from `http://www.esm.jp/jude-web/index.html`.

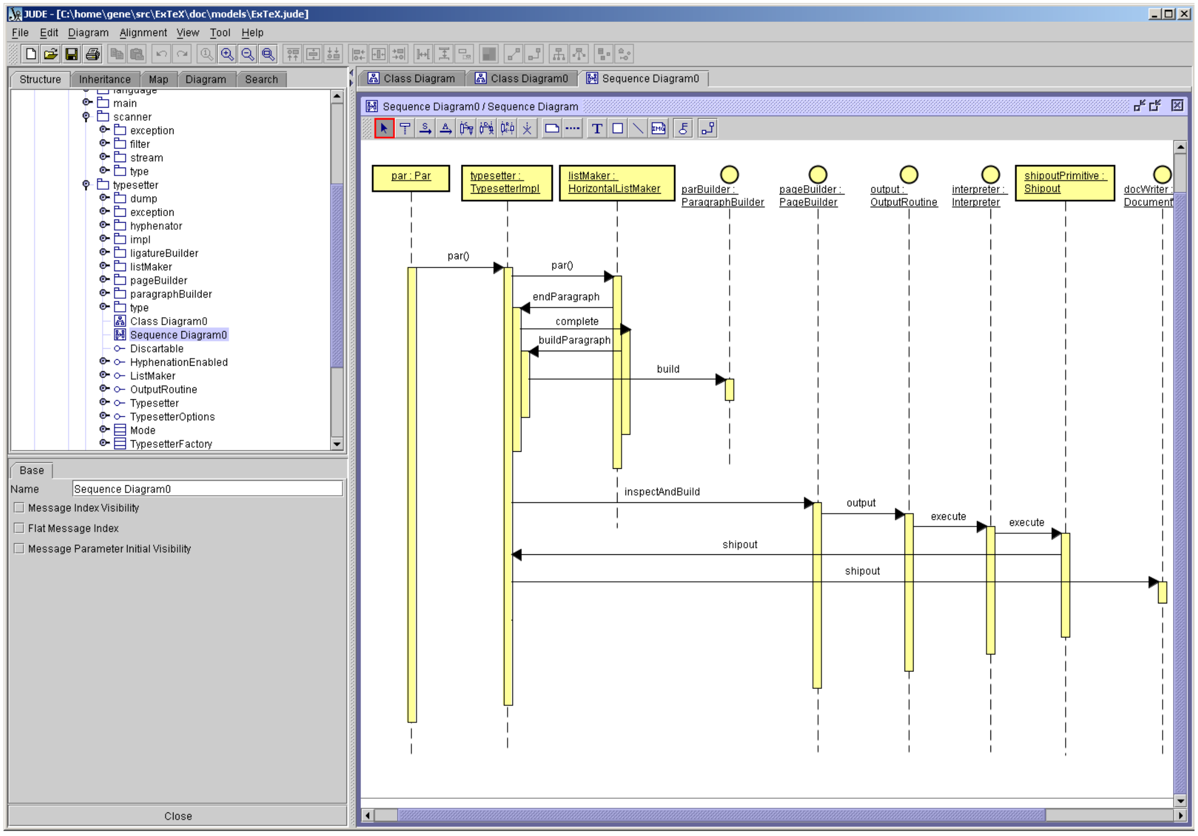


Figure 3.23.: Jude

Jude should be used for any situations where UML diagrams are needed. A screenshot of Jude can be seen in figure 3.23.

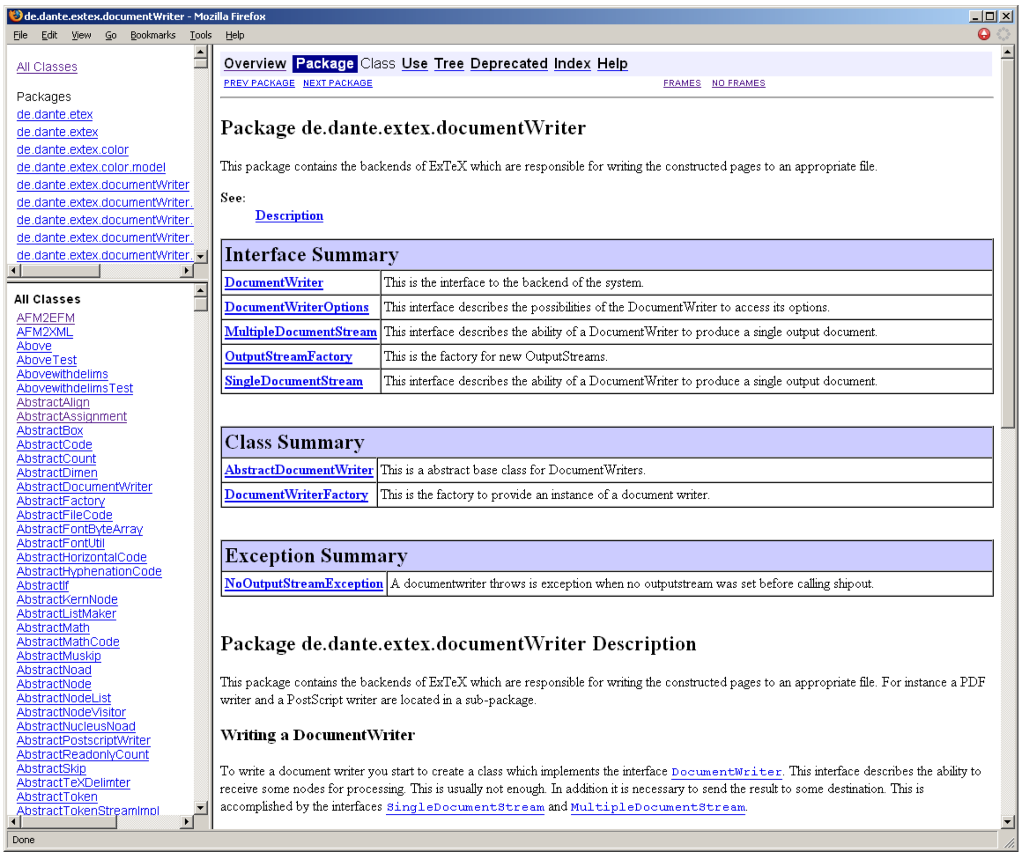
Models for ε_χTEX should be placed in the directory `doc/models`.

4. Source Code Documentation

The source code has to be documented. TEX shows us a good example of a proper documenatation. Donald Knuth has invented the Web system to keep together the documentation and the source code. The source code and documentation are extracted from a common file. In the Java world the Javadoc system has been invented for a similar purpose.

4.1. Javadoc

The Javadoc conventions for comments make it possible to extract the relevant part of the documentation and generate several outut formats from it. The primary output format is HTML.



Package de.dante.extex.documentWriter

This package contains the backends of ExTeX which are responsible for writing the constructed pages to an appropriate file.

See:
Description

Interface Summary	
DocumentWriter	This is the interface to the backend of the system.
DocumentWriterOptions	This interface describes the possibilities of the DocumentWriter to access its options.
MultipleDocumentStream	This interface describes the ability of a DocumentWriter to produce a single output document.
OutputStreamFactory	This is the factory for new OutputStreams.
SingleDocumentStream	This interface describes the ability of a DocumentWriter to produce a single output document.

Class Summary	
AbstractDocumentWriter	This is a abstract base class for DocumentWriters.
DocumentWriterFactory	This is the factory to provide an instance of a document writer.

Exception Summary	
NoOutputStreamException	A documentwriter throws is exception when no outputstream was set before calling shipout.

Package de.dante.extex.documentWriter Description

This package contains the backends of ExTeX which are responsible for writing the constructed pages to an appropriate file. For instance a PDF writer and a PostScript writer are located in a sub-package.

Writing a DocumentWriter

To write a document writer you start to create a class which implements the interface `DocumentWriter`. This interface describes the ability to receive some nodes for processing. This is usually not enough. In addition it is necessary to send the result to some destination. This is accomplished by the interfaces `SingleDocumentStream` and `MultipleDocumentStream`.

Figure 4.1.: Javadoc in the Browser

4.2. Documentation of Primitives

The documentation of the primitives is contained in the Javadoc comments of the implementing Java classes. A script is used to extract the information from the sources for the User's Manual. To make this happen, the documentation meant for the manual has to be marked and formatted specially.

To be completed.

5. Quality Assurance and Unit Tests

Quality assurance and testing play an important rôle in software development. Automated regression tests help to guarantee that funtionality is preserved across releases.

5.1. Deficiencies of the Trip Test

Donald Knuth has provided the trip test for TEX. This test is not suitable for ε_χTEX for several reasons:

- The trip test compares the log file and the dvi output. ε_χTEX does not guarantee identical log files.
- And ε_χTEX may produce more than dvi.
- The trip test covers only part of the functionality of TEX. It is interesting to test other features too.
- The trip test contains tests for failures as well. In those cases the bahaviour of ε_χTEX might be different.

As a consequence ε_χTEX comes with an own set of test cases.

5.2. Anatomy of a JUnit Test Class

To be completed.

```
package de.dante.extex.interpreter.primitives;

import de.dante.test.ExTeXLauncher;

/**
 * This is a test suite for the primitive <tt>\relax</tt>.
 *
 * @author <a href="mailto:gene@gerd-neugebauer.de">Gerd Neugebauer</a>
 * @version $Revision: 1.1 $
 */
public class RelaxTest extends ExTeXLauncher {

    /**
     * Method for running the tests standalone.
     *
     * @param args command line parameter
     */
    public static void main(final String[] args) {

        junit.textui.TestRunner.run(RelaxTest.class);
    }

    /**
     * Constructor for RelaxTest.
     *
     * @param arg the name
     */
    public RelaxTest(final String arg) {

        super(arg);
    }

    /**
     * Test case checking that a pure \relax has no effect.
     * @throws Exception in case of an error
     */
    public void test1() throws Exception {

        runCode(//--- input code ---
                "\\relax",
                //--- log message ---
                "",
                //--- output channel ---
                "");
    }
}
```

5.3. Creating Test Cases for the Interpreter

To be completed.

6. The Source Tree Organization

In this section the description of the directory hierarchy is contained. This structure is oriented on the structuring proposed by Maven (`http://maven.apache.org`).

6.1. The Toplevel Directory

The toplevel directory of an εχTEX project contains certain files and sub-directories.

develop
: The sub-directory `develop` contains bits and pieces needed for development only.

doc
: The sub-directory `doc` contains documentation – papers written by the εχTEX Group and material collected from elsewhere.

lib
: The sub-directory `lib` contains libraries which need to be present for the final executable to run.

src
: The sub-directory `src` contains the sources.

target
: The sub-directory `target` contains the generated files. This directory is not present in the CVS archive.

tmp
: The sub-directory `tmp` may contain intermediary files. This directory is not present in the CVS archive.

util
: The sub-directory `util` contains some utilities for development. They are not included into the installer.

work
: The sub-directory `work` may contain working files of single developers. It is not shared via the repositiory.

www
: The sub-directory `www` contains the sources for the Web pages of εχTEX.

6.2. `develop`: Development Support

eclipse

lib

6.3. `doc`: Documentation

DevelopersGuide

Library

Publications

UsersGuide

models

notes

6.4. `lib`: Third-Party Libraries

This directory contains libraries needed for $\varepsilon_{\mathcal{X}}$TEX to run.

6.5. `src`: The Sources

font

java

javadoc

text

6.6. `util`: Utilities

This directory contains vairous utitlies and scripts.

Installer

6.7. `work`: User's Working Area

Any user may have some files in the project area, Those files should not be committed to the Repository. For this purpose the directory `work` is reserved.

6.8. `www`: The Web Site

src

7. Design Details

This chapter contains some explanations, tips & tricks. It might be helpful to read them when you are concerned with the related topics.

7.1. Writing New Primitives

The core primitives of ε_χTEX are written in Java and bound to control sequences or active characters. In this section we will explain how to write new primitives in Java.

7.1.1. Executable Code

Executable primitives are those primitives which can be invoked in a left-hand-side context of the expansion. This is the case whenever the next top-level macro is treated. You can consider for example the treatment of the macro `\def` as such a case:

```
\def\abc{123}
```

In this example `\def` is an executable primitive.

Executable code has to implement the interface Code. Doing this directly is not hard. Nevertheless the abstract base class AbstractCode is provided which contains default implementations for all methods already. Thus only the interesting methods have to be overwritten.

In the simplest case only a constructor with one String argument and the method `execute()` has to be defined. Such an empty frame can be seen in the following example:

```
package my.package;

import de.dante.extex.interpreter.context.Context;
import de.dante.extex.interpreter.Flags;
import de.dante.extex.interpreter.TokenSource;
import de.dante.extex.interpreter.exception.InterpreterException;
import de.dante.extex.interpreter.type.AbstractCode;
import de.dante.extex.typesetter.Typesetter;

class MyPrimitive extends AbstractCode {

  public MyPrimitive(final String name) {
    super(name);
    // initialization code -- if required
  }
```

```
    public boolean execute(final Flags prefix,
                           final Context context,
                           final TokenSource source,
                           final Typesetter typesetter
                          ) throws InterpreterException {
      // implement the execution behaviour here
      return true;
    }
  }
```

In the method `execute()` you have access to other components. This can be utilized to implement the desired functionality. The following parameters are provided:

Flags **`prefix`** This parameter gives access to prefix arguments like `\immediate` or `\global`. For this purpose the class Flags provides appropriate getters. You can even modify the flags passed to the method. Usually you should invoke `prefix.clear()` somewhere in your implementation when the prefix is not needed any more. If this method is omitted then the prefix is passed on to the next execution. This can be desirable if you want to implement a prefix primitive yourself.

Context **`context`** The context provides reading and writing access to the data stored in the processor. This information is the memory. It is written to file when dumping a format. Refer to the documentation of the interface Context for details.

TokenSource **`source`** The source provides access to the token stream. It can be used to get the next tokens if required. For example when implementing a primitive like `\def` it is necessary to read the next tokens as arguments: the macro name, the parameter pattern, and the expansion text. The token source can also be used to push tokens to the input stream to be read back in later. This feature is used when implementing expandable primitives.

Typesetter **`typesetter`** The typesetter is the component which collects nodes and finally sends them to the document writer. With access to this component it is possible to produce some output to the paper.

The return value indicates how to deal with prefix flags. The usual behaviour is to return *true*. This indicates that the flags should be cleared afterwards. For those primitives which modify the prefix flags the return value *false* must be used.

7.1.2. Registering New Macros

The primary way to register new macros is in the configuration file used by ε_χTEX. For example the default file is located in the package `config` and named `extex.xml`. There you can find lines like the following one:

```
    <define name="def"
            class="de.dante.extex.interpreter.primitives.macro.Def"/>
```

To add another primitive to ε_χTEX you should make a copy of this configuration file under a different name and add a line like the one shown above:

```
    <define name="myPrim"
            class="my.package.MyPrimitive"/>
```

Now you can invoke ε_χTEX on the command line with the parameter `-configuration` or add a line `extex.config` to your `.extex` file pointing ε_χTEX to your new configuration file:

```
    extex -configuration config/myExTeX.xml
```

This is enough. In the instance of ε_χTEX with these settings the new macro `\myPrim` is defined and points to your code for execution.

7.1.3. Registering New Macros Dynamically

One extension provided with ε_χTEX contains a dynamic definition of new macros. Those macros are defined at runtime. The assignment of the Java code to the macro name can be controlled with the help of a primitive. Check out whether the macro `\javadef` is defined in one of the configuration files provided and consult the documentation.

7.1.4. Exceptions

The implementing Java code for new primitives can signal abnormal situations with the help of exceptions. The exceptions used should be derived from InterpreterException. `RuntimeExceptions` and `Errors` or derived classes should not be used.

ε_χTEX provides means for externalzing strings. Thus it should be made easy to translate the messages to other languages. For this purpose the class Localizer is provided. See the documentation of this class for details.

7.1.5. Assignments

Assignments are a special kind of executable code. TEX defines that the parameter `\globaldef` is evaluated and the macro `\afterassignment` has some effect. To ease the development of assignments the abstract base class AbstractAssignment is provided. This class defines the method `execute()` appropriately. The only task left is to overwrite the method `assign()` to perform the assignment.

```
   package my.package;

   import de.dante.extex.interpreter.contect.Context;
   import de.dante.extex.interpreter.Flags;
```

```
import de.dante.extex.interpreter.TokenSource;
import de.dante.extex.interpreter.exception.InterpreterException;
import de.dante.extex.interpreter.type.AbstractAssignment;
import de.dante.extex.typesetter.Typesetter;

class MyAssign extends AbstractAssignment {

  public MyAssign(final String name) {
    super(name);
    // initialization code -- if required
  }

  public void assign(final Flags prefix,
                     final Context context,
                     final TokenSource source,
                     final Typesetter typesetter
                    ) throws InterpreterException {
    // implement the assignment here
  }
}
```

The arguments of the method `assign()` are the same as the arguments of `execute()` described above. In contrast to the remarks made there it is not necessary to return something. The clearing of the flags is done in the abstract class automatically.

7.1.6. Expandable Code

Some macros are expandable. This means that they can be used on the right-hand-side of an invocation as well. This feature is expressed by the interface ExpandableCode. Since Java does not allow multiple inheritance no abstract base class is provided.

To implement an expandable primitive it is sufficient to declare the interface for the class and implement the method `expand()`. This is sketched in the following example:

```
package my.package;

import de.dante.extex.interpreter.contect.Context;
import de.dante.extex.interpreter.Flags;
import de.dante.extex.interpreter.TokenSource;
import de.dante.extex.interpreter.exception.InterpreterException;
import de.dante.extex.interpreter.type.AbstractCode;
import de.dante.extex.interpreter.type.ExpandableCode;
import de.dante.extex.typesetter.Typesetter;

class MyExpandable extends AbstractCode implements ExpandableCode {

  public MyExpandable(final String name) {
    super(name);
```

```
        // initialization code -- if required
      }

      public boolean execute(final Flags prefix,
                             final Context context,
                             final TokenSource source,
                             final Typesetter typesetter
                            ) throws InterpreterException {
        // implement the execution behaviour here
        return true;
      }

      public void evaluate(final Flags prefix,
                           final Context context,
                           final TokenSource source,
                           final Typesetter typesetter
                          ) throws InterpreterException {
        // implement the evaluation behaviour here
      }
    }
```

The parameters of `evaluate()` are ikdentical to those of `execute()`. But note, that the expected behaviour of `evaluate()` is that it does *not* modify the context or the typesetter but exclusively modifies the token source. Usually it reads some tokens and piuts back its result to the token stream.

7.1.7. Conditionals – Also Called Ifs

Conditionals are special because they modify the flow of control. In the macro programming language of TEX this may lead to a mode where tokens are absorbed at high speed. In this mode is necessary to identify conitionals to honor matching pairs of start and end tokens.

All neccesary actions are performed by the abstract base class AbstractIf. The only thing to do is to implement the method `conditional()` which computes whether the then or the else branch should be considered relevant. This is shown in the following example:

```
    package my.package;

    import de.dante.extex.interpreter.primitives.conditional.AbstractIf;
    import de.dante.extex.interpreter.contect.Context;
    import de.dante.extex.interpreter.Flags;
    import de.dante.extex.interpreter.TokenSource;
    import de.dante.extex.interpreter.exception.InterpreterException;
    import de.dante.extex.typesetter.Typesetter;
```

```
  class MyIf extends AbstractIf {

    public MyIf(final String name) {
      super(name);
      // initialization code -- if required
    }

    public boolean conditional(final Flags prefix,
                               final Context context,
                               final TokenSource source,
                               final Typesetter typesetter
                              ) throws InterpreterException {
      // implement the evaluation of the conditional here
  return result;
    }
  }
```

The parameters are the same as the parameters for `execute()` described above.

Note that any conditional is expandable automatically. Thus it should not modify the context or the typesetter.

7.1.8. Interaction With Other Macros

Several primitives of ε_χTeX are implemented generically. Let us consider for example the macro `\the`. This primitive simply gathers the next token and delegates the task of providing an appropriate definition for `\the` to the definition of this token.

The ability to be usable after `\the` is expressed with the help of the interface Theable. Thus it is enough for a primitive to implement this interface if it needs to be usable after `\the`.

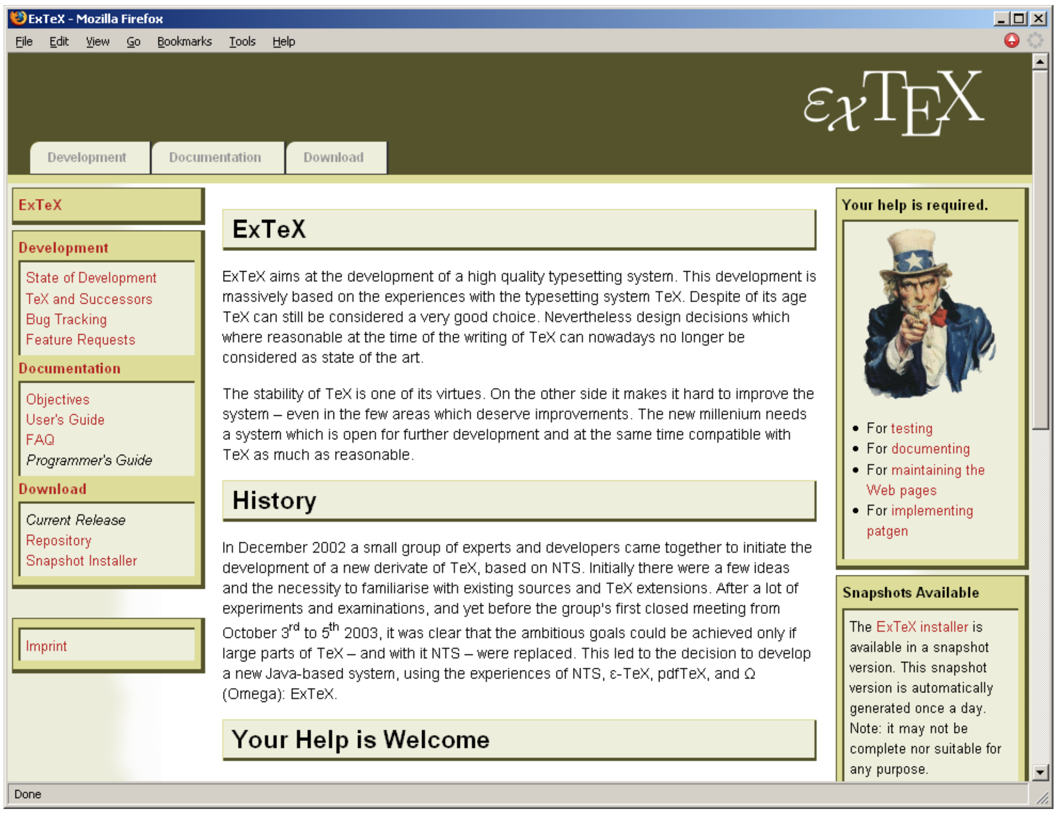
The following list contains some macros of TeX and the related interfaces:

`\advance`	Advanceable
`\box`	Boxable
`\count`	CountConvertible
`\dimen`	DimenConvertible
`\divide`	Dividable
`\font`	FontConvertible
`\multiply`	Multiplyable
`\show`	Showable
`\showthe`	Theable
`\the`	Theable

8. The Web Pages

8.1. Overview

ε_{χ}TEX has a domain of its own. This domain `www.extex.org` has been registered by DANTE e.V. In this location the official Web pages (see figure 8.1) are provided.



ExTeX

ExTeX aims at the development of a high quality typesetting system. This development is massively based on the experiences with the typesetting system TeX. Despite of its age TeX can still be considered a very good choice. Nevertheless design decisions which where reasonable at the time of the writing of TeX can nowadays no longer be considered as state of the art.

The stability of TeX is one of its virtues. On the other side it makes it hard to improve the system – even in the few areas which deserve improvements. The new millenium needs a system which is open for further development and at the same time compatible with TeX as much as reasonable.

History

In December 2002 a small group of experts and developers came together to initiate the development of a new derivate of TeX, based on NTS. Initially there were a few ideas and the necessity to familiarise with existing sources and TeX extensions. After a lot of experiments and examinations, and yet before the group's first closed meeting from October 3rd to 5th 2003, it was clear that the ambitious goals could be achieved only if large parts of TeX – and with it NTS – were replaced. This led to the decision to develop a new Java-based system, using the experiences of NTS, ε-TeX, pdfTeX, and Ω (Omega): ExTeX.

Your Help is Welcome

Your help is required.

- For testing
- For documenting
- For maintaining the Web pages
- For implementing patgen

Snapshots Available

The ExTeX installer is available in a snapshot version. This snapshot version is automatically generated once a day. Note: it may not be complete nor suitable for any purpose.

Figure 8.1.: www.extex.org

The Web pages are build with a simple generator for a Web site written in Perl. It has been made for ε_{χ}TEX. The aim is the ease of maintainance for normal content of pages. They are stored as simple HTML files and augmented automatically upon publication.

The layout is separated form the content and stored in several files. This makes it very easy to adapt the appearance without touching the contents.

The sources are kept in the subdirectory `src`. The generated files are put into the subdirectory `www`. Both locations can be configured.

To generate the Web site run the following command, where the current directory is the directory `www`:

```
#  make
```

This command creates a complete directory hierarchy with all necessary sub-directories in `../target/www`. An exception are the directories named `CVS`. Those directories are ignored.

The files starting with `.` or ending in `~` or in `.bak` are also ignored. The files not ending in `.html` are copied into the destination tree. The files ending in `.html` are processed as follows: Text is inserted before the `</head>` tag from the file `.headEnd`. Text is inserted after the `<body>` tag from the file `.bodyStart`. Text is inserted before the `</body>` tag from the file `.bodyEnd`.

The text to be inserted is sought in the current directory and in case of failure upwards in the super-directories until it is found. In the inserted files the following entities and tags are replaced:

`&top;`
: this is the relative path to the top directory.

`&year;`
: this is the current year when generating.

`&month;`
: this is the current month when generating.

`&day;`
: this is the current day when generating.

`<tabs/>`
: this is replaced by the contents of the file .tabs.

`<navigation/>`
: this is replaced by the contents of the file .navigation.

`<info/>`
: this is replaced by the contents of the file .info.

Note, that even so it looks like XML processing, currently the processing is based on string manipulation. Thus tricks possible with XML might not work here.

8.2. Layout

The current layout has the scheme shown in figure 8.2.

The Header contains the right aligned Logo only. It is the same on all pages. The Tab Bar shows the topmost navigation items with the Tab metophor. The Navigation

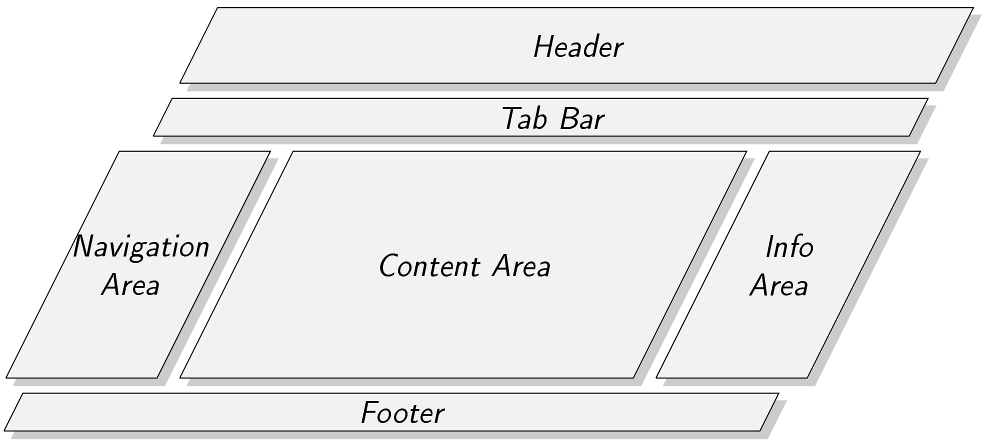


Figure 8.2.: Layout of the Web pages

Area shows all navigation items. It is the same on all pages. The Info Area shows some info items specific for the current navigation item.

The Content Area contains the contents of the page. This is maintained by the site authors. The Footer contains a simple copyrigt note.

8.3. Automatic Generation

The web pages are generated automatically every night. This task is performed with the help of a cron job on shell.berlios.de under the account gene. In the course of this generation the current sources are checked out from te CVS repository

Thus the normal user simply has to edit the pages in the area `www/src` and check them into the CVS repository. The rest happens automagically.

9. Licenses for ε_χTEX

The project goal of ε_χTEX is to provide a typesetting library facility. The global licence of the heart of ε_χTEX is therefore the LGPL. But in order to keep open linking facility against all the different licenses we must be choose carefully third party program licenses, i.e. jar files.

9.1. Acceptable Licenses

The licence that we accept and why:

- LGPL
  for obvious reasons.

- modified BSD licence (http://www.xfree86.org/3.3.6/COPYRIGHT2.html#5)
  We can always convert this licence to LGPL.

- Apache 2.0 licence
  Unfortunately linking with GPL2 programs is in gray area. Interpretation differs between FSF and Apache but GPLv3 will be compatible.

- Public domain
  (obvious)

- GPL with explicit linking clause that don't enforce viral clause of GPL and insure clear encapsulation of GPL license. It is the GPL with this special exception like this:

  > As a special exception, if other files instantiate templates or use macros or inline functions from this file, or you compile this file and link it with other works to produce a work based on this file, this file does not by itself cause the resulting work to be covered by the GNU General Public License. However the source code for this file must still be made available in accordance with section (3) of the GNU General Public License.

  or like this:

[0] Author: Bastien Roucaries

> Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the G NU General Public License cover the whole combination.
>
> As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

9.2. Non-Acceptable Licenses

The licence that *don't* accept and why:

- GPL license:
  We would loose the free linking facility.
- Apache 1.1 license:
  It forbids linking with GPL software. Check if an upgrade to Apache 2.0 exists.
- Eclipse public license:
  It forbids linking with GPL software and has nasty side effects for proprietary software. Perhaps side effects exist with Apache 2.0 due to patents issue.
- Original BSD license and Apache 1.0
  Have a boring clause that forbids linking with a lot of different licenses – particularly GPL – and non carefully written proprietary software.

A. Licenses

A.1. GNU Free Documentation License