

# **Exemplar Developer's Handbook**

**Conor McDermottroe**

---

# **Exemplar Developer's Handbook**

Conor McDermottroe

Copyright © 2005, 2006, 2007 Conor McDermottroe

---

---

---

# Table of Contents

Introduction .....	vi
1. Project Goals & Philosophy .....	1
2. Overall Architecture .....	2
Module Structure .....	2
Input Module .....	2
Model Module .....	2
Output Module .....	2
UI Module .....	2
Data Flow Diagram .....	2
3. Input Modules .....	3
Input Module characteristics .....	3
The <i>DTD</i> Input Module .....	3
How To: Create A New Input Module .....	3
4. Model Module .....	4
Features .....	4
Limitations .....	4
5. Output Modules .....	5
Output Module Characteristics .....	5
How To: Create A New Output Module .....	5
6. User Interface Modules .....	6
<i>UI</i> module characteristics .....	6
How To: Create A New <i>UI</i> Module .....	6
Logging Subsystem .....	6
Using The Logging System .....	6
The Logging System For <i>UI</i> Creators .....	6
Localised Messages .....	6
How To: Add A New Message To The Program .....	6
Options Processing .....	6
How To: Use Options .....	6
How To: Add A New Option .....	6
7. Debugging And Error Handling Features .....	7
Exception Handling .....	7
How To: Create Your Own Exceptions .....	7
Assertion Mechanism .....	7
8. Testing Framework .....	8
9. Build System .....	9
10. How To Get Help .....	10
A. Glossary .....	11
B. Bibliography .....	12

---

## List of Figures

2.1. Overall Architecture .....	2
2.2. Data Flow .....	2

---

# Introduction

This book is designed as a higher-level companion to the *API* documentation which can be produced directly from the source. In some parts of this documentation it is assumed that you have read the *API* documentation or at least have it available. A copy of the *API* documentation should be available at the location from which you obtained this book.

The purpose of this book is to provide prospective and current developers with an overview of how Exemplar is put together. Some low-level implementation details will be discussed here but, for the most part, documentation for low-level details is documented in the code (either with JavaDoc or the code itself).

As with all documentation this book may lag behind the code. If in doubt, trust the code or get help (see How To Get Help for details). If you submit a patch for the code it would be appreciated if you submit a corresponding patch for this documentation. The source for this documentation is available with the source for the program.

---

# Chapter 1. Project Goals & Philosophy

Exemplar is a program created to make it easier to work with XML. It began life as a program to generate the smallest possible parser for a given vocabulary of XML. While it still retains the capability to do this, it is now a more general program intended to support conversion of specifications of XML vocabularies (DTDs, Schemas, etc) into useful code.

Exemplar is designed to support plug-in modules to implement input translators, output producers and user/application interfaces. This should make it easier to add features to Exemplar and/or integrate it with third-party products. This is described later in Overall Architecture.

---

# Chapter 2. Overall Architecture

## Module Structure

When the program has started and is configured, there are four structural elements which cooperate to convert the input to the output. This arrangement can be seen in the diagram "Overall Architecture".

**Figure 2.1. Overall Architecture**

## Input Module

The input module reads and parses the input file(s) and creates an instance of `XMLDocumentType` which is the Model Module. All of the input modules are defined under the package `com.mcdermottroe.exemplar.input` and every input module implements the interface `InputModule`.

## Model Module

The model module is not a module *per se* but is rather a data structure providing a standard representation of a DTD or schema to allow a standard interface for both the input and output modules to work on. All of the classes which make up the model are defined in the package `com.mcdermottroe.exemplar.model` and are wrapped in an instance of `XMLDocumentType` when being passed between input and output.

## Output Module

The output module transforms the model into the relevant output. This can either be one or more files or a Java object, depending on the exact output module used. All output modules are defined in sub-packages of the package `com.mcdermottroe.exemplar.output` and all are subclasses of `XMLParserGenerator`.

## UI Module

The user interface module provides control options to the user or calling process and provides localisation and options configuration facilities to the rest of the program. All user interface modules are defined under the `com.mcdermottroe.exemplar.ui` package.

## Data Flow Diagram

The "Data Flow" diagram shows the flow of data through the program.

**Figure 2.2. Data Flow**

---

# Chapter 3. Input Modules

## Input Module characteristics

Describe input module interface and layout here.

## The *DTD* Input Module

Describe the *DTD* input module in detail here.

## How To: Create A New Input Module

Describe the construction of an input module for a fictional simplistic *DTD* format here.

---

# Chapter 4. Model Module

## Features

Describe the features, both used and unused, that the current model is capable of handling.

## Limitations

Describe the limitations of the current model.

---

# Chapter 5. Output Modules

## Output Module Characteristics

Describe output module interface and layout here.

## How To: Create A New Output Module

Describe the creation of a *DTD* output module here.

---

# Chapter 6. User Interface Modules

## *UI* module characteristics

Describe the *UI* module interface and layout here.

## How To: Create A New *UI* Module

Show how to make a new *UI* module.

## Logging Subsystem

### Using The Logging System

To use the logging system, you need to do two things:

### The Logging System For UI Creators

Describe the way in which logging is done.

## Localised Messages

Describe the way messages are done in the program.

## How To: Add A New Message To The Program

Show the methods.

## Options Processing

Describe how options processing is done.

## How To: Use Options

Show `get()`

## How To: Add A New Option

Show how to introduce a new option

---

# Chapter 7. Debugging And Error Handling Features

## Exception Handling

Describe how the exception handling works in the program.

### How To: Create Your Own Exceptions

All exceptions in Exemplar are subclasses of `com.mcdermottroe.exemplar.Exception` which allows for consistent error handling throughout the application. The easiest way to create your own exception is to copy one of the existing exceptions and change the name. All exceptions must implement the four constructors `Exception()`, `Exception(String)`, `Exception(String, Throwable)`, and `Exception(Throwable)` so that the exception chaining facilities of Java can be used.

### Assertion Mechanism

Describe the assertion mechanism here. Refer to [Meyer97] where appropriate.

---

# Chapter 8. Testing Framework

Describe the testing framework here.

---

# Chapter 9. Build System

Describe the build system here.

---

# Chapter 10. How To Get Help

Contact details here.

---

# Appendix A. Glossary

The reader should be familiar with most, if not all, of the following terms. They are included here only for completeness.

## Glossary

Application Program Interface	The set of classes, methods and fields in a program or library which may be used by developers or users to interact with it.
Document Type Definition	A formal description of a vocabulary of XML or SGML which defines the permissible structure of all documents conforming to that DTD.
User Interface	The portion of the program with which the user interacts. See User Interface Modules for how user interfaces are implemented in Exemplar.
Extensible Markup Language	A standard which defines a way in which markup languages can be written. See [Bray00] for details.

---

# Appendix B. Bibliography

This bibliography contains both references to other texts mentioned above and to ones which would be useful to many developers working on Exemplar. All of the books below are well worth reading.

## Bibliography

### Books

- [Aho86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley.
- [Gamma94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- [Gosling00] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. 2000. *The Java Language Specification*. Second Edition. Sun Microsystems.
- [Meyer97] Bertrand Meyer. 1997. *Object Oriented Software Construction*. Second Edition. Prentice-Hall.

### Specifications

- [Bray00] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. 6th October 2000. *Extensible Markup Language (XML) 1.0*. Second Edition. W3C. <http://www.w3.org/TR/2000/REC-xml-20001006>.

### Software Documentation

- [Ant] *Apache Ant Documentation*. The Apache Ant Project.