

# Introduction

## Table of contents

Introduction. . . . .	1
Internals. . . . .	1

### Introduction

Hawron is designed for building a custom Offline-Processing-Application with Cocoon. Cocoon is a very powerfull XML-Publishing-Framework, the main usage is and will be in a Servlet-Environment. But the capabilities for working with XML in an Offline-Environment are nevertheless unique. You get all the XML-technology in one framework. The [Forrest-Project](#) is such a project, which uses Cocoon.

Hawron tries to be a GUI-interface for user, to edit,generating with Cocoon and publish the results. The first experiments where, to put OpenOffice and Cocoon together for building OpenOffice-driven websites. The application grows up and after some redesign, there is more place for other components, editors, viewers, action and things I have never thought about.

You can extend Hawron in different ways:

- write a plugin for your special XML-Editor, Viewer or Action
- write a Component, which replace a given Components
- add your native(non-java) XML-Editor, viewer
- switch off components you don't need

Not all is possible in an easy way at the moment.

Hawron has switched to Apache-[Avalon](#) as underlying infrastructure. This process is not finished and not all parts will be (or could be) changed to. Hawron's locale-implementation is such a point.

Some former build-in components are rolled out as plugins now, maybe there will follow more.

### Internals

After changing to Apache-[Avalon](#) the ApplicationContainer is one of the core-components (but not an Avalon-component). The ApplicationContainer holds all components together and manage the lifecycle of the Components. At the moment all Components (and some Objects which are not defined by an interface) are listed in the "roles.conf" in the "conf" directory. For supporting plugins the ApplicationContainer uses the PluginManager, which report all founded plugins back to the Container and to the PluginReciver (every class can registrate as a PluginReciver to the PluginManager and will get all plugins for interfaces the PluginReciver await). PluginComponets override the default implementation of components. The ApplicationContainer and the PluginManager are the base underlying infrastructure.

The GUI-environment is held by the ApplicationFrame, which holds the ProjectView. The ProjectView is a container for ProjectViewComponents, like ContextView, ProcessView and the ServletView (JettyPlugin). There are some components around this, like ApplicationSetup and the MenuBar (which will be changed later) and the Toolbar, where other components can interact with.