improve
FOUNDATIONS

Introduction

5.3

improve foundations

Introduction

Revision History		
Revision 1.0	12/08/2010	F. Esnault
Creation of document for first open source release		
Revision 1.0.1	28/02/2011	F. Esnault
Minor changes before release		
Revision 5.3	25/01/2012	F. Esnault
Changes about IF version 5.3		

1. Introduction

1.1. Document presentation

This document aims to introduce you to the Improve Foundations project and its open source part in particular.

Improve Foundations (http://www.improve-foundations.com/) has been created in 2003 by a French company called Improve that has been involved into many Java/JEE projects in different companies and organizations since the early beginnings of the Java technology. The general purpose of the Improve Foundations project is to make easier the development of business applications with Java and open source frameworks, libraries and tools.

After having launched some open source projects in the GUI field such as Struts Layout (http://struts.improve-technologies.com/) and Rialto (http://rialto.improve-technologies.com/), Improve, that is now Open Wide Technologies (http://technologies.openwide.fr), has decided to bring the core of Improve Foundations to the open source community.

This document presents you the main ideas and principle of the project before you go deeper into the technical documentation.

1.2. Scope

The present document deals with Improve Foundations Open Edition version 5.3.

Please note that:

- versions before 5.2 were not open source,
- there is a larger Improve Foundations Corporate Edition including more extensions, tools and professional support.

1.3. For whom is this document intended?

This document is intended for persons wishing to understand the main technical principles of Improve Foundations.

1.4. Preliminary requirements

Architecture, XML, object-oriented and Java programming skills are required to understand this document.

1.5. To go further

This document only gives an overview of the solution. Its reading may be further completed by:

- IF-Core technical documentation, which provides the concrete details to use the core functions,
- technical documentation of every extension, which presents its contributions and how to use it,
- several guides provided with Improve Foundations Corporate Edition.

2. Project purposes

As the introduction mentioned it, the aim of Improve Foundations is to make easier the development of business applications with Java and open source frameworks, libraries and tools.

2.1. Issues

Implementing a project often requires a sound match of application architecture and technical architecture. Success, however, requires above all a thorough grasp of its technological complexity.

It is increasingly rare to find the foundations of an IT project based on a single, unique product. Projects are more usually based on a collection of technologies from various sources. Such diversity accordingly impacts the level of skills required to carry out projects of this kind.

It is therefore advisable to manage complexity by integrating all these technologies within a technical foundation as a form of tool box offering an architecture, technical services and development methodology ready for use.

Many companies have first made the choice to create their own technical foundations, choosing open source elements, defining some rules and maintaining some glue code or specialized APIs. Nevertheless, after a few years, many of these companies are facing difficulties:

- Choices that were made do not seem perennial, they are challenged every 2 years (if not less)
- The team did a lot of development work but little assistance to projects or conversely, the 'framework' team is so involved in the projects there is no time left to maintain the foundation code properly
- The technical foundation was formed by highly skilled experts and is full of good ideas, unfortunately these people have left the company leaving a whole set that is difficult to maintain
- When the applications are developed by external software companies it is difficult to make them use the foundation without considering some extra cost and a form of disengagement in case of technical problems
- The cost of the 'framework' team is very high. It is difficult to justify such a cost in a company which is not directly involved in software business.

2.2. Objectives and philosophy

Improve Foundations has been conceived with the idea to be the equivalent of a company technical foundation without the drawbacks previously described. It has already been used by several companies that also contribute to its evolutions.

Improve Foundations does not necessary bring very innovative ideas or specific powerful features. Indeed, its purpose is to keep things simple, clear and easy to use by any developper whatever his or her level is. Many software solutions are conceived by people having high technical skills but what they create is too often a piece of genius that cannot be properly used by beginners or basic developpers. However, these profiles are commonly involved in IT projects simply because their salary is lower. One could argue that only good developpers with high salaries should develop but the real world is not turning this way. That's why Improve Foundations tends to facilitate the work of these basic developpers in order to ensure quality, homogeneity and ease of maintenance.

Finally, a last general and important question could be: *why use Improve Foundations whereas there are many well known frameworks available for software development?*

The answer is that Improve Foundations is not that sort of library that helps you on a specific topic. We even think it should not be called a 'framework' but a technical 'foundation' or 'base' to make the distinction clear. Indeed, while classic open source frameworks bring many possibilities without taking any decision, Improve Foundations is coming with some strong closed choices in order to make things simpler, quicker and more efficient. Of course, theses choices will not enable to cover all cases but we think if they can bring simplicity on 80 % of them, this is a significant advantage. Why charge 100 % of the work with a complexity that is only useful on a few specific cases ?

For those who think Improve Foundations is very closed to the Spring Framework, we must admit that it is true on some aspects but that it is still like comparing apples and oranges. Spring is a powerful IoC framework that tends to expand on other issues, Improve Foundations is a more global solution that considers Spring as a way among others to implement some useful mechanisms. In other words, you can of course choose Spring and Hibernate and any other framework to build your application, find which versions are compatible with each other, define the architecture, decide the way you lay out the code, explain which coding rules you want to apply, etc... with Improve Foundations, the main parts of this work are already done.

Improve Foundations is not another project to have fun with technology. It is there to bring rationality and efficiency to business IT projects. If this is what you need, then go on reading!

3. Technical overview

3.1. General principles

The following general principles are adopted by Improve Foundations:

- Functional (business) developments are carried out in Java.
- An application will be hosted in a web application (WebApp).
- The technical architecture does not impose any graphic layer which could be a light client or fat client in type.
- The technical foundation leaves developers responsible only for strictly functional aspects. It provides the tools and services enabling developers to focus on the business-specific aspects.
  - Surface controls are carried out on the client and server side.
  - More complex business rule controls (inter-field) are carried out on the server side.
- Applications may be divided into a variable number of layers.
- Applications comply with Service Oriented Architecture (SOA) concepts.
  - Systematic use of interfaces for service-type components, i.e., clearly defined service APIs
  - An application may be described by the list of services provided (Service Name, Interface, Implementation, Layer)
- Transaction management
  - A user interaction must not lock in resources beyond the HTTP request or another type of request. To do so, the Improve Foundations base includes a transaction management mechanism.
- Context management
  - The technical foundation provides a set of services to manage various contexts.

3.2. Layered architecture

Improve Foundations introduces a notion of 'Layer' which is basically a component container with an architectural purpose.

Therefore, an application built on top of Improve Foundations is composed of several layers.

The usual model includes these layers :

- the Physical layer representing data storage (typically RDBMS),
- the Enterprise layer, which contains a set of services to manipulate data (DAO),
- the Application layer, which contains a set of services representing the business logic of the application (business rules and checks, transactions...),

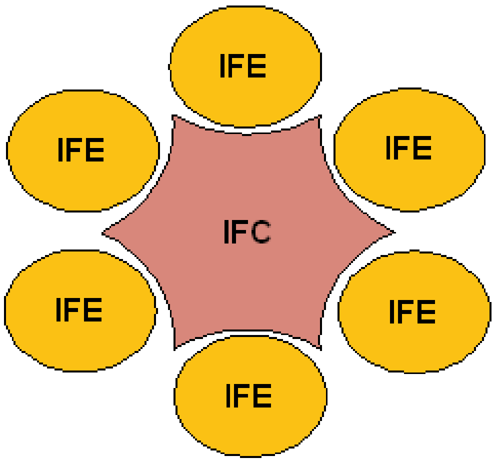
- the Client layer, which is commonly the implementation of GUI (presentation and navigation aspects).

Improve Foundations introduces also a 'Foundation layer' which represents the technical services available in the application.

This model is the default one and is the more common. It is however possible to define other layers if necessary.

3.3. Project organization

The foundation comprises a core, « Improve Foundations Core » and extensions which name is prefixed by 'IFE'.



Improve Foundations Open Edition contains the following modules:

- *IF-Core*
- *IFE-Hibernate* , for ORM style of data access using the Hibernate open source framework,
- *IFE-Web* , for general web mechanisms,
- *IFE-Struts* , to use the Struts framework as a solution for the Client layer,
- *IFE-StrutsLayout* , to make easier JSP coding using a heigh level taglib.
- *IFE-JSF* , as an alternate solution to implement the Client layer.
- *IFE-Errors* , a addon to the core for error management.

Therefore, the Open Edition is a good way to start with Improve Foundations concepts and API. It enables you to build a typical application with well known ORM and MVC solutions. However, if your needs go further, access to the Corporate Edition is recommended.

Indeed, the Corporate Edition brings many elements around this core that are suitable for business applications in a professional context: *IFE-Audit, IFE-Authentication, IFE-Authorization, IFE-Batch, IFE-Converter, IFE-CSV, IFE-EasyWeb, IFE-Flex, IFE-FTP, IFE-GWT, IFE-JasperReports, IFE-JDBC, IFE-JMS, IFE-JPA, IFE-LDAP,* *IFE-Mail, IFE-MessAdmin, IFE-MultiProcess, IFE-PDF, IFE-PortletBridge, IFE-RemotingSOAP, IFE-RemotingREST, IFE-RemotingRMI, IFE-RialtoGWT, IFE-RTFTemplate, IFE-Security, IFE-WebHelp, IFE-WebServices, IFE-XMLMapping, IFE-Saas, IFE-SaasStruts, EasyProjects, EasyForms, EasyStrutsLayout, EasyServices, extensive guides for architecture, design and development activities, and professional support*. Do not hesitate to contact Open Wide Technologies at *technologies@openwide.fr* if you need more information about the Corporate Edition.

3.4. IF-Core Features

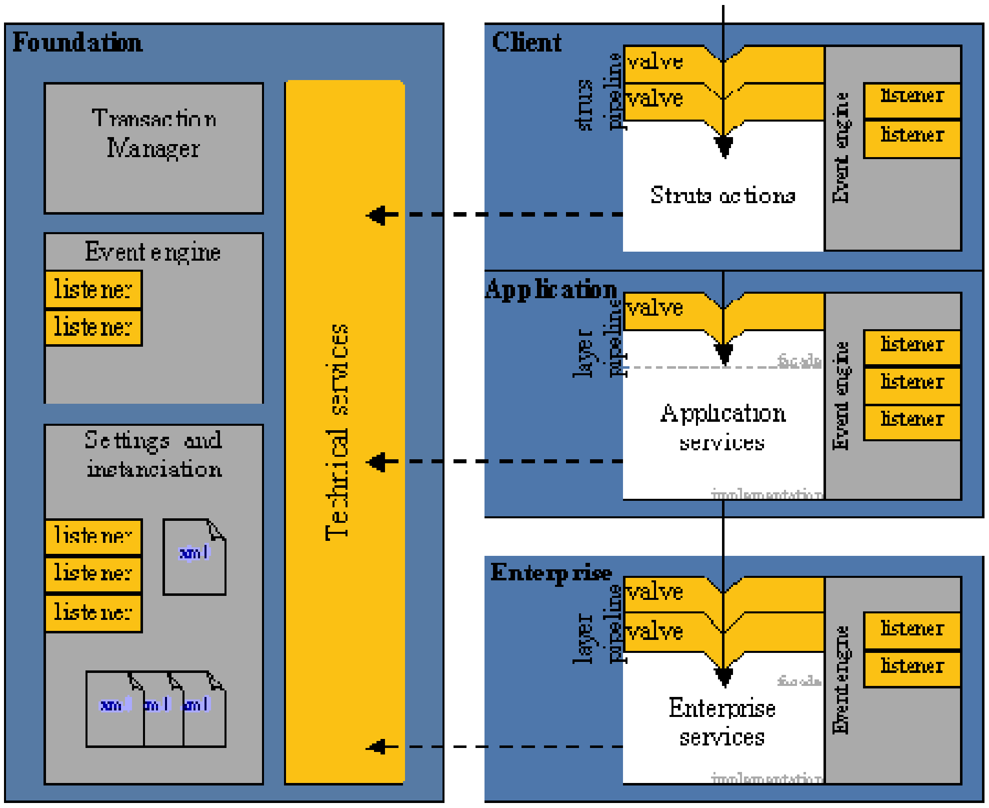


Figure 3.1. General foundation architecture (overview) – the foundation is displayed in grey; foundation extensions are in yellow and functional developments are in white.

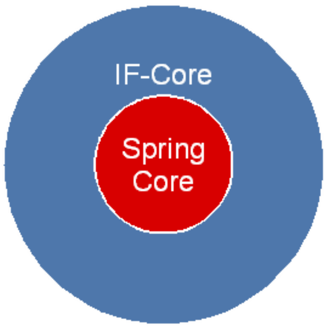
Improve Foundations Core brings the folowing notions and mechanisms:

- the *layer* concept, which is a container for components configured in XML,
- the management of execution *environments* to centralize and externalize variables to set,
- the *services* , that are made of simple Java classes and interfaces,
- *valves* and *listeners* , that enables to trigger behaviors without polluting the code of business services,
- *connections and transactions* , that are handled independently of the technical solution (there is even an implicit JTA mode that is ready to be run with no additional configuration),
- *dependency injection* , that is an available function (though not a central idea),

- a *thread context* management,
- a notion of *'Connected Services'* to make easier DAO coding,
- a way to implement easily distributed architectures using a *remoting* mechanism.

3.5. Spring Framework Integration

Version 5 of Improve Foundations is the fruit of a re-engineering of the technical foundation with the possibilities offered by the Spring framework. This choice enables the use of a certain number of proven Spring mechanisms and increases the potential of the foundation benefiting from the power of this framework and the contributions of this community.



IF-Core considers Spring Core as a low level component. It remains the only API directly seen by users, but Spring Core is the underlying engine.

IF-Core therefore, in concrete terms, comprises 2 parts: the main core (IFC) and the engine. In version 5, implementation of the available engine is IFC-Spring. It should be noted that IFC itself has no dependency on Spring.

This division was made to enable the use of other implementations in the future (particularly other IoC containers could replace it if such a component became a JSR standard), but in its present state, the core does not have full functionality that would enable it to avoid using Spring.

An application relying on Improve Foundations must therefore integrate in its classpath the IFC and IFC-Spring modules but no specific knowledge of Spring is needed to use it.

3.6. Compatibility and dependencies

Version 5.3 of Improve Foundations is compatible and supported with:

- JDK 1.5 and 1.6
- JEE 5

This means it should be operational on any application server which is compliant with these standards. Please note however that a simple servlet container such as Apache Tomcat is usually enough to run an application based on Improve Foundations.

The foundation may also operate with other JDK and versions of JEE, but these configurations have not been appropriately tested.

The most recent versions of the open source libraries have been integrated to the degree that they remain compatible with all the requirements and technical constraints (see extensions documentation).