2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Maßnahmen gegen das Code-Chaos in Standard-Software

Erweiterung von Standard-Software um kundenindividuelle Funktionalitäten

Martin Aschoff

AGNITAS AG

Martin Aschoff

- Founder of AGNITAS AG

  (e-mail & marketing automation)
- Managing Director Development & Technology
- Maintainer OpenEMM (open source application)
- Pastime Blogger (os-inside.org)

The Holy Grail: Standard Software

- develop once, sell/rent multiple times
- lots of input to improve the software
- community helps with support
- sometimes an user even pays for a new feature (and all other users benefit for free)

The Curse of Success

- more clients

  → more requests to customize features

- bigger clients

  → more requests of highly customized features

- more highly customized features

  → more difficulties to integrate into the mainline

The Challenge

Integrate customized functionality

AND

keep the mainline lean

If you fail in the short term

AGNiTAS

If you fail in the long term

AGNITAS

An Approach to Implement Customized Functionality

- features of general interest: integrate into **mainline**

  (advantage: update-stable)

- features of interest for several customers: integrate into **mainline**

  and wrap with **permission**

- features of interest for only one customer: **extension**

  (advantage: independent of core updates, mainline stays lean)

- comprehensive feature set of interest for only one client:

  separate development trunk based on main trunk

How it all begins

```
if (companyID == 815) {
        // special code
} else {
        // default code
}
```

…and how it continues

```
if (companyID == 815 || companyID == 4711) {
        // special code
        if (companyID == 4711) {
                // very special code
        }
} else {
        // default code
}
```

Handling of Instance-Wide Customized Parameters at Deploy Time

Template in build.properties.xslt

jdbc.emmDB.jndiName=<xsl:value-of select="properties/jdbc/emmDB/jndiName" />
jdbc.emmDB.dialect=<xsl:value-of select="properties/jdbc/emmDB/dialect" />

build.properties.xml for MySQL:

```
<jdbc>
        <emmDB>
                <jndiName>emm_db</jndiName>
                <dialect>org.hibernate.dialect.MySQLDialect</dialect>
        </emmDB>
        <cmsDB>
                <jndiName>cms_db</jndiName>
                <dialect>org.hibernate.dialect.MySQLDialect</dialect>
        </cmsDB>
</jdbc>
```

Handling of Instance-Wide Customized Parameters at Deploy Time

build.properties.xml for Oracle:

```
<jdbc>
        <emmDB>
                <jndiName>emm_db</jndiName>
                <dialect>org.hibernate.dialect.Oracle9Dialect</dialect>
        </emmDB>
        <cmsDB>
                <jndiName>cms_db</jndiName>
                <dialect>org.hibernate.dialect.Oracle9Dialect</dialect>
        </cmsDB>
</jdbc>
```

Merging by Ant script with xslt tag and tag extension xmltask

Handling of Instance-Wide Customized Parameters at Run Time

```
SELECT * FROM config_tbl;
+-------------+--------------+-----------------+
| class       | name         | value           |
+-------------+---------     +-----------------+
| system      | licence      | NULL            |
| pickup      | rdir         | asp@10.1.1.232  |
| linkchecker | linktimeout  | 20000           |
| linkchecker | threadcount  | 20              |
| velocity    | abortscripts | disabled        |
+-------------+---------+----+-----------------+
```

Handling of Customer-Specific Parameters at Run Time

```
DESC company_tbl;
```

Field	Type	Null	Key	Default
company_id	int(11) unsigned	NO	PRI	NULL
shortname	varchar(30)	NO		
description	varchar(100)	NO		
status	varchar(20)	YES		NULL
mailtracking	int(11) unsigned	NO		0
creator_company_id	int(11)	NO		1
mailerset	int(11)	NO		0
mailloop_domain	varchar(200)	NO		
rdir_domain	varchar(100)	NO		http://rdir.de
customer_type	varchar(50)	NO		UNKNOWN
send_immediately	int(11)	NO		0
offpeak	int(11)	NO		0
notification_email	varchar(100)	YES		NULL
expire_stat	int(11)	NO		0
max_fields	int(11)	NO		0
expire_bounce	int(11)	NO		0
expire_onepixel	int(11)	NO		0
expire_cookie	int(11)	NO		0
expire_recipient	int(11)	YES		30
expire_upload	int(11)	NO		14
max_recipients	int(11)	YES		0
export_notify	int(11)	YES		0
max_login_fails	int(3)	NO		3
login_block_time	int(5)	NO		300
creation_date	timestamp	NO		0000-00-00 00:00:00
timestamp	timestamp	NO		CURRENT_TIMESTAMP
sector	varchar(50)	YES		NULL
business_field	varchar(100)	YES		NULL

And if that is not enough

```
DESC company_info_tbl;
+---------------+---------------+------+-----+---------------------+
| Field         | Type          | Null | Key | Default             |
+---------------+---------------+------+-----+---------------------+
| company_id    | int(11)       | NO   |     | 0                   |
| cname         | varchar(32)   | NO   |     |                     |
| cvalue        | varchar(4000) | NO   |     |                     |
| description   | varchar(250)  | NO   |     |                     |
| creation_date | timestamp     | NO   |     | 0000-00-00 00:00:00 |
| timestamp     | timestamp     | NO   |     | CURRENT_TIMESTAMP   |
+---------------+---------------+------+-----+---------------------+
```

```
SELECT * FROM company_info_tbl;
+------------+----------------------+-----------------------------------------------------------+-----+---------------------+---------------------+
| company_id | cname                | cvalue                                                    | desc| creation_date       | timestamp           |
+------------+----------------------+-----------------------------------------------------------+-----+---------------------+---------------------+
|          0 | keep-xml-files       | true                                                      | ... | 2013-05-23 16:44:18 | 2013-05-23 16:44:18 |
|          0 | use-extended-usertypes | true                                                    | ... | 2013-05-23 16:44:18 | 2013-05-23 16:44:18 |
|          0 | url-default          | http://rdir.de                                            | ... | 2013-05-23 16:44:18 | 2013-05-23 16:44:18 |
|          0 | generate-coded-urls  | false                                                     | ... | 2013-05-23 16:44:18 | 2013-05-23 16:44:18 |
|          0 | url-profile          | http://www.agnitas.de/emm/profil.html?id=%(company-id)    | ... | 2013-05-23 16:44:18 | 2013-05-23 16:44:18 |
|          0 | url-unsubscribe      | http://www.agnitas.de/emm/abmeldung.html?id=%(company-id) | ... | 2013-05-23 16:44:18 | 2013-05-23 16:44:18 |
|          0 | flush-buffer-size    | 2000                                                      | ... | 2013-05-23 16:44:18 | 2013-05-23 16:44:18 |
+------------+----------------------+-----------------------------------------------------------+-----+---------------------+---------------------+
```

Interfaces and Dependency Injection

Open Source:

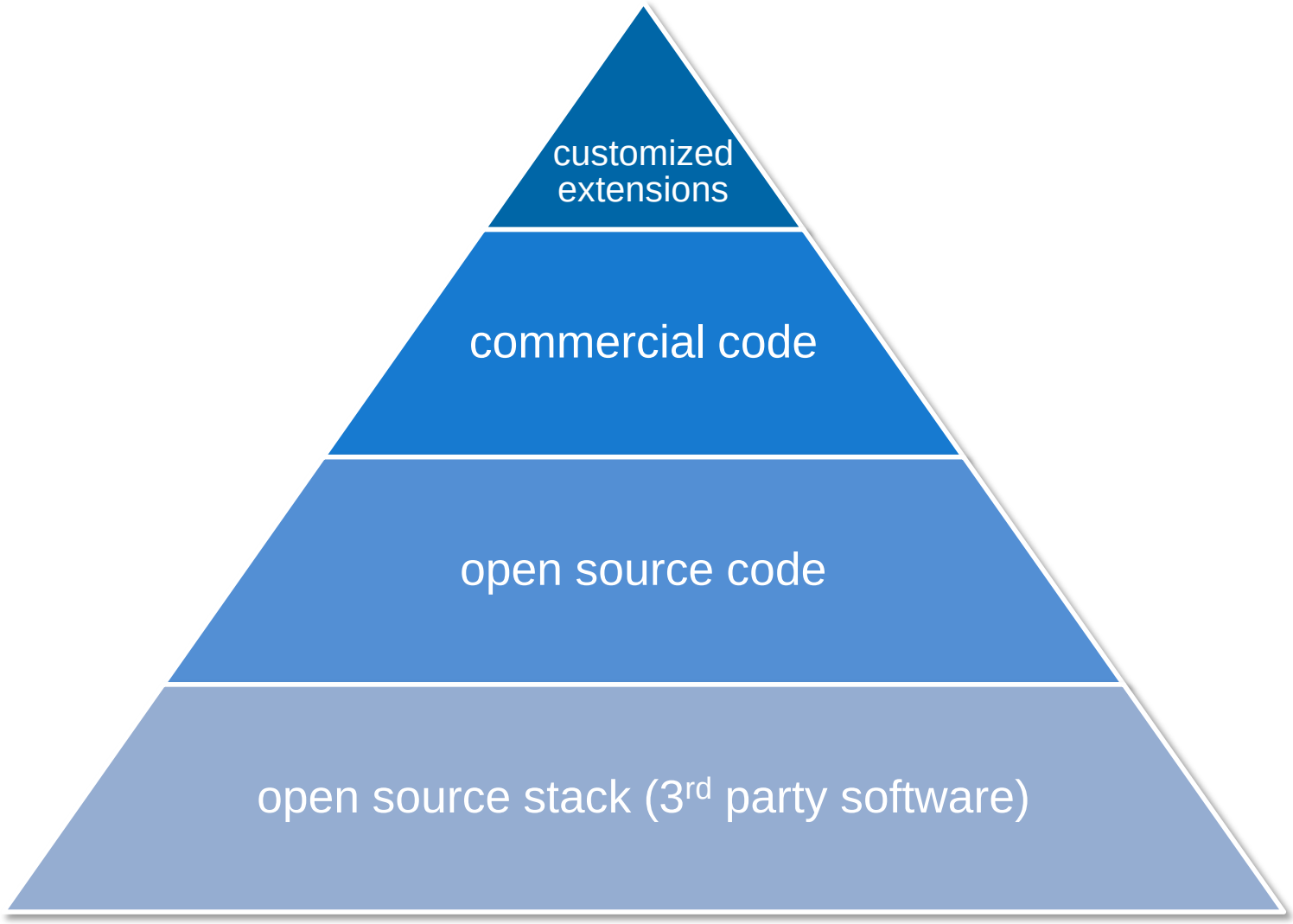
```
<bean id="MailingStat" class="org.agnitas.stat.impl.MailingStatImpl" singleton="false">
        <property name="dataSource" ref="dataSource" />
        <property name="targetDao" ref="TargetDao" />
        <property name="mailingDao" ref="MailingDao" />
</bean>
<bean id="MailingDao" class="org.agnitas.dao.impl.MailingDaoImpl">
        <property name="dataSource" ref="dataSource" />
</bean>
<bean id="TargetDao" class="org.agnitas.dao.impl.TargetDaoImpl">
        <property name="dataSource" ref="dataSource" />
</bean>
```

Interfaces and Dependency Injection

Commercial Version:

```
<bean id="MailingStat" class="com.agnitas.stat.impl.ComMailingStatImpl" singleton="false">
        <property name="dataSource" ref="dataSource" />
        <property name="targetDao" ref="TargetDao" />
        <property name="mailingDao" ref="MailingDao" />
        <property name="companyDao" ref="CompanyDao" />
</bean>
<bean id="MailingDao" class="com.agnitas.dao.impl.ComMailingDaoImpl">
        <property name="dataSource" ref="dataSource" />
        <property name="companyDao" ref="CompanyDao" />
        <property name="undoMailingDao" ref="UndoMailingDao" />
        <property name="undoMailingComponentDao" ref="UndoMailingComponentDao" />
        <property name="undoDynContentDao" ref="UndoDynContentDao" />
</bean>
<bean id="TargetDao" class="org.agnitas.dao.impl.TargetDaoImpl">
        <property name="dataSource" ref="dataSource" />
</bean>
```

Code Components Building on Each Other



Trunks Building on Top of Each Other

```
public abstract class BaseDaoImpl
{ // code of base class }
public class MailingDaoImpl extends BaseDaoImpl implements MailingDao
{ // code of open source class }

public class ComMailingDaoImpl extends MailingDaoImpl implements
ComMailingDao
{ // code of commercial class }

public class ConradMailingDaoImpl extends ComMailingDaoImpl
implements ConradMailingDao
{ // code of customer-specific class}
```

Plugin Architecture

Advantages for developers:

- an extension developer does not have to comprehend the whole code of the software but only needs to understand its extension interface, consisting of extension points and the extension API
- the software core does not get bloated with functionality (less complexity, less dependencies)
- extensions can be maintained independently from the core software and vice versa

Plugin Architecture

Advantages for developers:

- updates of the software core are easier to execute because the only dependency on existing extensions are the locations of extension points and the signatures of API classes and methods
- customer-specific features do not have to be integrated into the software core
- when secondary features have to be refactored/extended, they can be sourced out to an extension to keep the core lean

Plugin Architecture

Advantages for users:

- development time for customer-specific features implemented as extensions will drop, leading to lower costs
- extensions can be deployed at any time, independently from release cycles of the software core
- users can develop their own extensions
- users can use extensions developed by third parties
- if the code of the whole extension system is open source, it makes comprehension of the extension interface even easier because it provides maximum transparency

Components of an Extension Architecture

- **Extension API**: official methods to be used by extension
- **Extension Points**: pre-defined hook in the software core (the contact), which defines an interface (the contract) that will be implemented by a class of an extension that was registered for this extension point before
- **Extension Manager**: sub-system of the software core with functionality to manage the lifecycle of extensions

Components of an Extension Architecture

- **Extension Registry**: central storage where the extension manager registers (and unregisters) extensions and holds all necessary configuration data from all installed extensions
- **Extension Repository**: location where public extensions will be available for download

Types of Extension Points

- GUI extension points to insert output of an extension into an existing JSP
- navigation extension points to add new menu items or tabs to the existing navigation
- feature extensions point to add new code which enhances the functionality

Components of an Extension

- a **manifest file** plugin.xml in XML format, holding configuration data to store in the extension registry (like version info, extension points, entries for extension points, etc.)
- **resource files** like property files for messages, navigation, etc.
- Java **classes** with program code of an extension (usually service layer classes, DAO layer classes and third-party JARs)
- **JSPs** to implement the GUI of an extension (sometimes accompanied with files for Javascript, HTML, CSS and images)

Example of a Manifest File

```
<plugin id="mailing_statistics_export" version="0.0.1" vendor="Agnitas AG">

        <attributes>
                <attribute id="i18n-bundle" value="messages-plugin" />
                <attribute id="plugin-name" value="Export mailing statistics" />
        </attributes>

        <!-- Dependencies to other plugins -->
        <requires>
                <import plugin-id="emm_core" />
                <import plugin-id="emm_core_navigation" />
        </requires>

        <!-- File structure setup -->
        <runtime>
                <library id="mailing-statistics-export-feature" path="classes/" type="code" />
        </runtime>

        <!-- Feature and navigation extension point bindings -->
        <extension plugin-id="emm_core" point-id="featurePlugin" id="mailing-statistics-export">
                <parameter id="class" value="org.agnitas.emm.plugin.mailingstatisticsexport.MailingStatisticsExportFeature" />
        </extension>

        <extension plugin-id="emm_core_navigation" point-id="tabs.statistics.compare" id="mailing-statistics-export-navigation">
                <parameter id="navigation-bundle" value="mailingStatExportTabs"/>
        </extension>
</plugin>
```

JPF Framework (jpf.sourceforge.net)

- inspired by plugin mechanism of Eclipse 2.x
- development efforts retired in 2007
- just one active (but positive) reference found (DMS LogicalDOC, www.logicaldoc.com)

JPF Framework (jpf.sourceforge.net)

Java Plug-in Framework (JPF) Project

Welcome to the **J**ava **P**lug-in **F**ramework project, the open source, LGPL licensed plug-in infrastructure library for new or existing Java projects. JPF can greatly improve the modularity and extensibility of your Java systems and minimize support and maintenance costs.

Home
System Overview
Project Roadmap
TODO List
Questions & Answers
References
License

Concepts
JPF Boot Library
Tutorial
JPF & Java IDE
API Reference
Plug-in DTD
JPF Tools Reference
Configuration
Reference

Project Summary
Project News
Download
Forum

German (de)

SOURCEFORGE.NET

What is JPF?

JPF provides a runtime engine that dynamically discovers and loads "plug-ins". A plug-in is a structured component that describes itself to JPF using a "manifest". JPF maintains a registry of available plug-ins and the functions they provide (via extension points and extensions).

One major goal of JPF is that the application (and its end-user) should not pay any memory or performance penalty for plug-ins that are installed, but not used. Plug-ins are added to the registry at application start-up or while the application is running but they are not loaded until they are called.

Main features

Open framework architecture
The framework API is designed as a set of Java interfaces and abstract classes. Developers can choose to implement their own "vision" of plug-ins and Framework runtime behavior. "Standard" or default implementations are provided by JPF so developers can start using the framework quickly and easily.

Clear and consistent API design
The JPF API has been carefully designed in order to reduce the the time developers need to become familiar with it.

Built-in integrity check
Registered plug-ins are checked for consistency during JPF start up and a detailed report of results is available.

Plug-ins are self-documenting
Plug-in developers may include documentation in the plug-in manifest. This includes inline comments or references to documents bundled with the plug-in.

Plug-in dependency check
Plug-in developers can declare dependencies between plug-ins. Dependency declarations can include the desired version ID and versions matching rules.

Strongly typed extension parameters
The plug-in manifest syntax provides a mechanism for declaring **typed** extension points parameters. This information is used by JPF when finding and loading extensions.

Lazy plug-in activation
Plug-in classes are loaded into memory only when they are actually needed. This feature is provided by specially designed Java class loaders instantiated for each plug-in.

"On the fly" plug-in registration and activation
Plug-ins can be "hot-registered" and even de-registered during application execution. What's more, registered plug-ins can be activated and deactivated "on the fly", minimizing runtime resource usage.

What can JPF bring to your Java project?

Plug-in component model
A JPF Plug-in is a component that has: a name (ID), a version identifier, code and/or other resources, a well-defined import interface, a well-defined export interface, and well-defined places where it can be extended (extension points). You can think of plug-ins as a module for your application.

Divide large applications into smaller, more manageable parts
Building applications as a set of independent, cooperating components is particularly useful when developing in teams.

Explicitly define the systems architecture
Plug-ins, prerequisites, extension points and extensions allow you to clearly define the architecture of your system in an easy-to-understand and standard way.

Make the application easily extendable
With extension points you can allow other developers to extend your application simply.

Documentation embedded into the system
This is more than javadoc. You can include documentation in the plug-in manifest and link it with any additional resources.

Tight control over application consistency
JPF's built-in integrity check keeps a close watch on your application's health, reducing maintenance costs.

Component Architecture (Java)

- standard module system "Jigsaw": first draft in 2006, postponed from Java 7 to Java 8 and from Java 8 to Java 9 (2016?)
  - latest news: according to the Java lead developer, Jigsaw will be restarted
- OSGi: de-facto standard for modular Java applications with dependency resolution and definable visibility at bundle level
  (to hide code)
- OSGi is big in application servers (GlassFish, JBoss), development tools (Eclipse, JIRA) and frameworks (Apache Camel, Sling)
- OSGi is not so big in (standard) applications

Problems with OSGi for applications (my personal opinion)

- great technology, but steep learning curve
- developers have to deal with a lot of low level stuff (lifecycle management, service listeners, bundle versioning, hot deployment, etc.)
- building web applications looks even more complicated

Problems with OSGi for applications (my personal opinion)

- everything must be a bundle - including 3rd party JARs
- for smaller projects and teams often overkill

  (application developers just want to get the job done)
- additional barrier (learning curve) for third parties to

  develop extensions

Sie wollen noch mehr wissen?

Martin Aschoff

c/o AGNITAS AG

Werner-Eckert-Str. 6

81829 München

Telefon: 089/552908-68

E-Mail: maschoff@agnitas.de