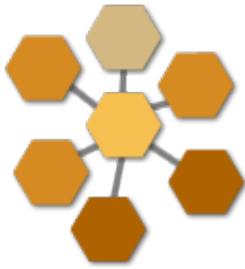


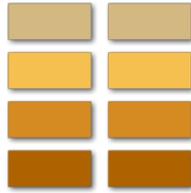
G A T H E R

TECHNICAL OVERVIEW

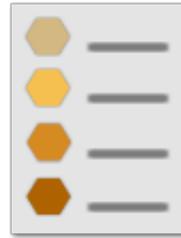
January 2009



gather



transform



report

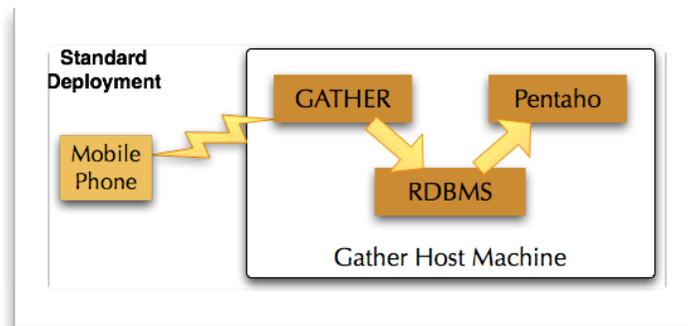
Introduction

Gather, Transform, Report

The steps in conducting a field study are simple: gather data from the field, transform the data into an analyzable format, and then produce reports on the data set. Whether conducting environmental impact studies, researching the migration of an endangered species, or monitoring public health, the pattern is consistent: gather, transform, report.

The GATHER framework implements that pattern for building data collection applications. Though the base application is domain-neutral, it can be used out-of-the-box to capture data, store it, and present it in a meaningful way.

Paired with other open source data collection and analysis tools, such as the JavaRosa mobile client or Pentaho for analytic reporting, the GATHER framework provides a best-of-breed collection of open source tools for the entire data management life-cycle.



Architecture

Overview

Written in Java, GATHER is a collection of OSGi bundles, organized into feature sets which can be assembled into a specific application at runtime. Instead of a monolithic application that makes servicing different domains difficult, GATHER is architected with everything as a plug-in. Features can be updated, swapped-out, or augmented easily at runtime, enabling easy domain-specific customization. This is particularly beneficial when delivering updates to bandwidth-constrained deployments.

Standard Deployment

The standard GATHER deployment is paired with JavaRosa mobile clients and a customized Pentaho installation. JavaRosa enables a mobile user to fill out an XForm and submit the instance data over HTTP. The GATHER server accepts the HTTP submission, stores it, and may perform additional handling. The data is stored in both archival (original) and reportable (transformed) schemas.

Standard and ad-hoc reporting is available through Pentaho and Jasper, the two leading open source reporting tools.

Bundle of Joy

Developing OSGi bundles encourages a classic library-oriented development style, making it easier to assemble a command-line application, Eclipse RCP, simple proxy server, or a server cluster. This flexibility is crucial to allowing domain-specific modifications to be easily created and added to a GATHER deployment.

Stacked Up

To the right, the common software stack view shows how the layers of a GATHER Application fit together.

The base is an OSGi Platform, providing a runtime environment and bundle lifecycle management.

The System Bundles provide generic and 3rd party services.

Of course the GATHER Framework is the most exciting layer. More about that in the next section.

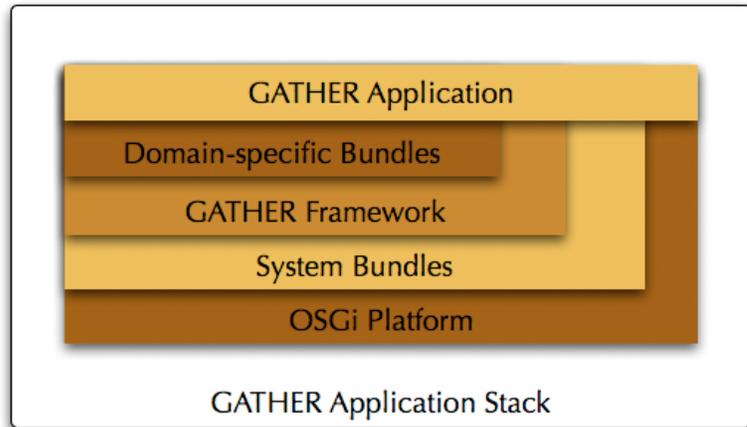
To all that, some Domain-specific Bundles for special handling of particular forms.

The main application finally assembles all the those bundles into a running application.

Standard Feature Sets

Currently, the GATHER Framework provides bundles organized into the following feature sets:

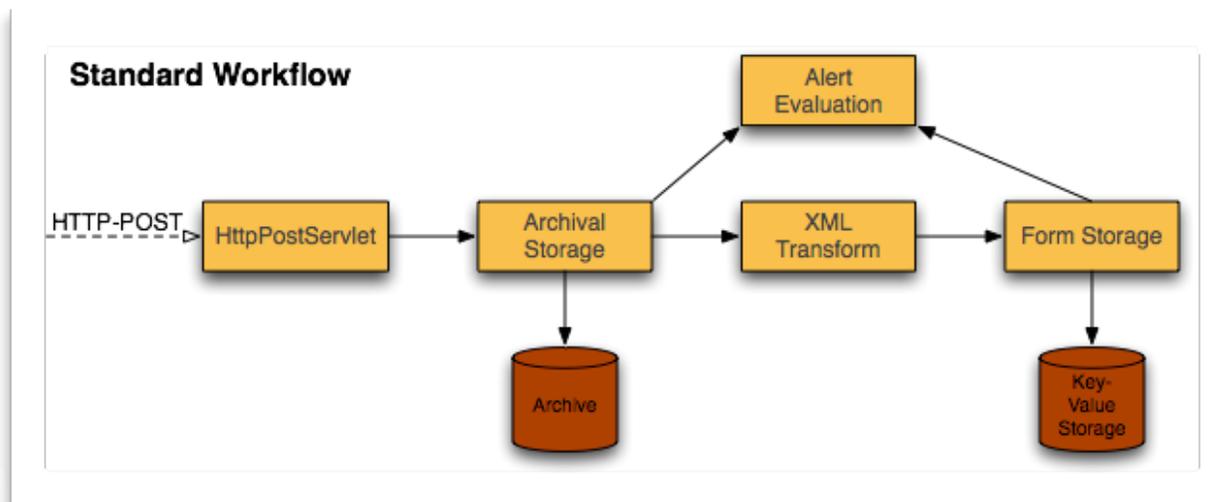
- gather-framework - core data models, interfaces, base classes, and utilities;
- gather-web - Spring-based web application;
- gather-archiver - archival storage of raw data, using RDBMS or file system;
- gather-forms - simple key-value based storage of form-like data, with a transform for XForm instance data;
- gather-tables¹ - dynamically generated relational tables for form data;
- gather-alert - content-triggered alert notification, using scripting-language predicates; and
- gather-report² - Jasper-based integrated reporting.



¹ under development

² under development

GATHER Application



Workflow

Out-of-the-box, the GATHER Application assembles all the feature bundles into a server for capturing and storing generic form data, with a web-based user interface. The standard workflow accepts XForm instance data over HTTP. The data is saved immediately in the Archival Storage, with Date and Time information as well as checksums. This is the lowest-risk operation, protecting against any data loss.

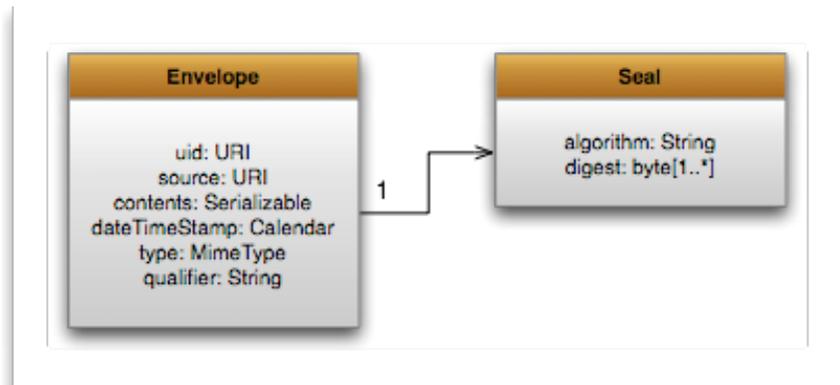
Next, the data is passed to the gather-alert subsystem to be evaluated. If any of the alert predicates are triggered, notification, such as an email or SMS, can be sent. The predicates themselves are written in a supported scripting language, simply returning a true/false value based on the input.

The data is then passed to any available transformer. The XML-to-form transformer is available by default, deconstructing XML documents (expected to be XForm instance data) into hierarchical collections of key-value properties.

The transformed XML is finally saved to the Form Storage, ready for basic reporting. This allows for generic XForms submitted to GATHER to be transformed to flat table structure for export to CSV, EXCEL, or use in one of the supported reporting engines.

Envelope Stuffing

When a form is received by HTTP, it is stuffed into an Envelope, a wrapper which is used to pass all data along the workflow. An Envelope is sealed with a SHA-1 message digest and annotated with enough metadata so that content can be handled uniformly. Each handling component is at least expected to handle Envelopes, possibly supplemented with content-specific services or operations. This allows for a generic workflow to be augmented by domain-specific extensions.



Envelopes can be stuffed with any Serializable content. The content is uniquely identified by a URI (either supplied or generated). The diagram to the right shows all the properties maintained by an Envelope. Envelopes are considered immutable. Though some implementations may allow the alteration of properties, the Seal can be used to ensure that the contents are still valid.

Envelope Handling

These are the generic interfaces for GATHER services which handle Envelopes, along with the features which implement them:

- EnvelopeSource - a service which produces content, wrapped in an Envelope
 - http-post servlet
- EnvelopeDestination - a service which can accept Envelopes
 - gather-alert
- EnvelopeStorage - a service which can store Envelopes
 - gather-archiver, gather-forms
- EnvelopeTransformer - a service which produces new content (with new Envelopes) based on source content
 - xml-to-forms transformer

Domain Specific

Alert Reports

While GATHER is immediately useful for capturing, transforming, and reporting data, the base application can be customized to fulfill domain specific needs. Custom reports and alert scripts are the first place to start.

Reports can be written against any of the default storage schemas, using either Pentaho or Jasper. At the moment, Pentaho offers more capability, including running ad-hoc reports; GATHER is supporting both reporting engines, however, as they are both feature-rich and widely used

Alert scripts follow a simple interface to accept input, returning either a true or false to trigger notification.

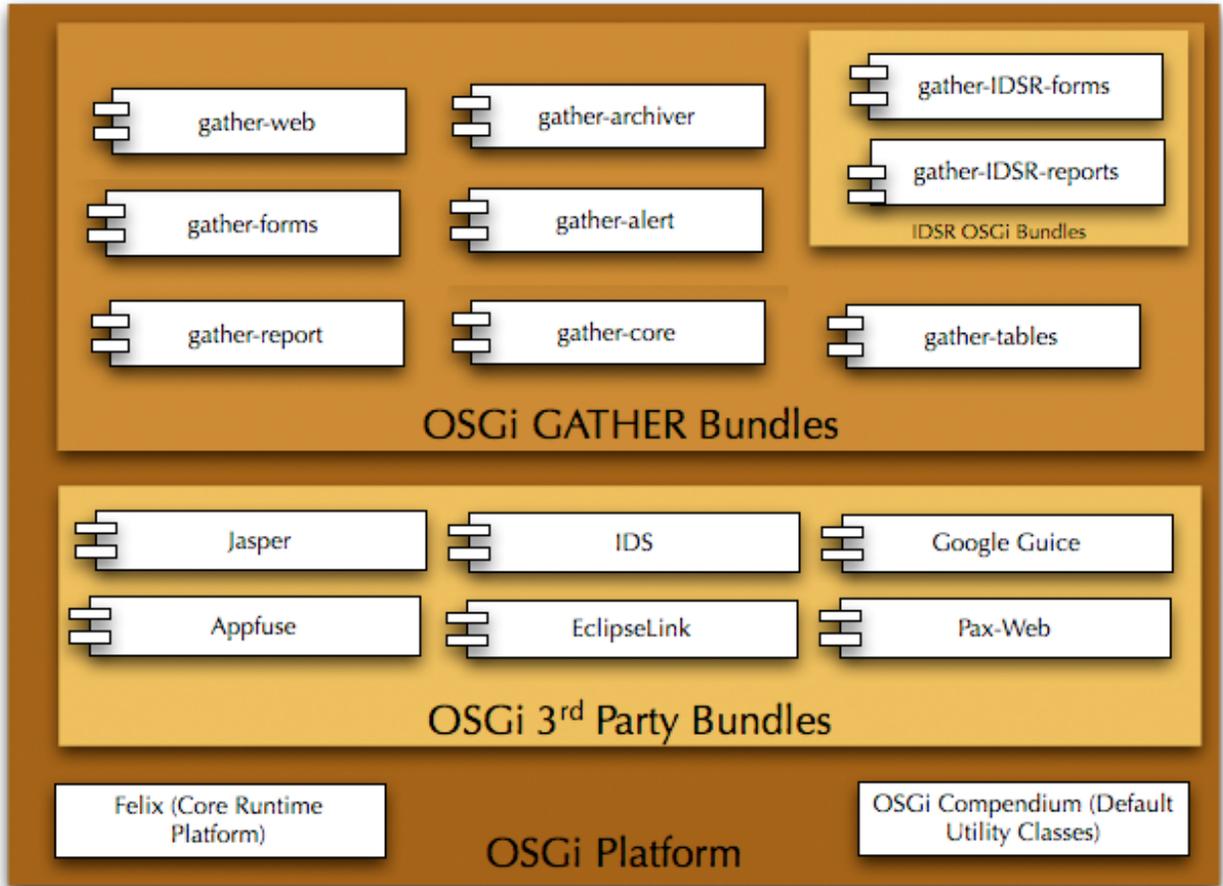
Extended Service

Everything is a plug-in, meaning that more technical requirements can be addressed by developing custom Envelope handlers which implement the appropriate interfaces. Build an OSGi compliant bundle, drop it in the appropriate GATHER directory, and the framework will notice the new services. Appropriate data will get routed as it arrives.

This modular approach will provide core developers and domain-specific developers an architecture that supports maximum flexibility and re-usability. It will also allow implementers to dynamically test, deploy, and remove (if necessary) features with ease.

Detailed Technical IDSR modules

The modular nature of the GATHER platform and OSGi means that there are many libraries to pick and choose from for any domain-specific application. Among the first GATHER applications is one that can accept and report on the internationally-accepted Integrated Disease Surveillance Report (IDSR). As depicted below, most of the functionality in the IDSR application comes from the core of the GATHER platform. There is a small amount of functionality that is created to support IDSR-specific characteristics of the application, such as specific alerts and custom reports.



Annex – Definition of Technical Terms

(Definitions from Wikipedia)

Felix is a community open source effort to implement the [OSGi R4 Service Platform](#), which includes the OSGi framework and standard services, as well as providing and supporting other interesting OSGi-related technologies. The ultimate goal is to provide a completely compliant implementation of the OSGi framework and standard services and to support a community around this technology. Felix currently implements a large portion of the OSGi release 4 specification, but additional work is necessary for full compliance. Despite this fact, the OSGi framework functionality provided by Felix is very stable.

OSGi technology originally targeted embedded devices and home services gateways, but it is ideally suited for any project that is interested in principles of modularity, component-oriented, and/or service-orientation. OSGi technology combines aspects of these aforementioned principles to define a dynamic service deployment framework that is amenable to remote management. As an example of a simple use case, Felix can be easily embedded into other projects and used as a plugin or dynamic extension mechanism; it serves this purpose much better than other systems that are used for similar purposes, such as [Java Management Extensions \(JMX\)](#).

JasperReports is the world's most popular open source Java reporting library. You can easily embed it into any Java application to deliver sophisticated print or web reporting. You can also use it to create output to files for further processing in applications like Excel. For users with more sophisticated report management requirements, reports designed for JasperReports can be easily migrated to the JasperSoft stand-alone report server, JasperDecisions.

JavaRosa (<http://code.javarosa.org>) is a prominent open source software for data collection on a wide-range of Java-enabled phones. It is under ongoing development by members of the OpenROSA (www.openrosa.org) consortium, a group of organizations working together to foster open-source, standards-based tools for mobile data collection, aggregation, analysis, and reporting. No other open source mobile data collection software has attracted such an active, broad community of software developers. JavaROSA has active developers in Bangladesh, Kenya, India, Norway, Pakistan, South Africa, Tanzania, Uganda, and the United States.

XForms is an [XML](#) format for the specification of a data processing model for XML data and [user interface\(s\)](#) for the XML data, such as [web forms](#). XForms was designed to be the next generation of [HTML](#) / [XHTML](#) forms, but is generic enough that it can also be used in a standalone manner or with presentation languages other than XHTML to describe a user interface and a set of common [data](#) manipulation tasks.

Unlike the original HTML forms, the creators of **XForms** have used a [Model-View-Controller](#) approach. The "model" consists of one or more XForms models describing form data, constraints upon that data, and submissions. The "view" describes what controls appear in the form, how they are grouped together, and what data they are bound to. [CSS](#) can be used to describe a form's appearance.

An XForms document can be as simple as an HTML form (by only specifying the submission element in the model section, and placing the controls in the body), but XForms includes many advanced features. For example, new data can be requested and used to update the form while it is running, much like using [XmlHttpRequest](#) / [AJAX](#) except without scripting. The form author can validate user data against [XML Schema](#) data types, require certain data, disable input controls or change sections of the form depending on circumstances, enforce particular relationships between data, input variable length arrays of data, output calculated values derived from form data, prefill entries us-

ing an XML document, respond to actions in real time (versus at submission time), and modify the style of each control depending on the device they are displayed on (browser versus mobile versus text only, etc.). There is often no need for any scripting with languages such as JavaScript.

Like legacy forms, XForms can use various non-XML submission protocols ([multipart/form-data](#), [application/x-www-form-urlencoded](#)), but a new feature is that XForms can send data to a server in XML format. XML documents can also be used to prefill data in the form. Because XML is a standard, many tools exist that can parse and modify data upon submission, unlike the case with legacy forms where in general the data needs to be parsed and manipulated on a case by case basis. XForms is itself an XML dialect, and therefore can create and be created from other XML documents using [XSLT](#). Using transformations, XForms can be automatically created from [XML Schemas](#), and XForms can be converted to legacy XHTML forms: this is basically how server side XForms work today.

The Pentaho BI Project is [Open Source application software](#) for [enterprise reporting](#), analysis, [dashboard](#), [data mining](#), [workflow](#) and [ETL](#) capabilities for [business intelligence](#) (BI) purposes.

Its name comes from the number of the products it has integrated (5, penta) and its hybrid OLAP (ho) nature (partially relational and partially dimensional).

The five products are:

- Mondrian : [Mondrian OLAP server](#) engine
- JPivot : [JSP](#) custom tag library that renders [OLAP](#) tables and charts
- Kettle : metadata-driven [ETL](#) tool
- Weka : [data mining](#) tool
- JFreeReport : library for generating [reports](#)

EclipseLink is an [open source object-relational mapping](#) package for [Java](#) developers. It provides a powerful and flexible framework for storing Java objects in a relational database or for converting Java objects to XML documents.

EclipseLink's original contribution came from the [Oracle TopLink](#) product in 2006.

EclipseLink supports several standard persistence related APIs including:

- [Java Persistence API \(JPA\)](#)
- [Java XML Binding \(JAXB\)](#)
- [Service Data Objects \(SDO\)](#)

EclipseLink provides support for:

- Object relational mapping (ORM)
- Object XML mapping (OXM)
- Object persistence to Enterprise Information Systems (EIS)
- Database web services

AppFuse is an [open-source Java EE web application framework](#). It is designed for quick and easy start up of development, while also using open-source Java technologies such as [Spring framework](#), [Hibernate](#) and [Struts](#). AppFuse was originally created by Matt Raible, who wanted to eliminate the "ramp up" time in building new web applications.

AppFuse provides a project skeleton, similar to the one that's created by an [IDE](#) when one clicks through a "new web project" wizard. AppFuse 1.x uses Ant to create the project, as well as build/test/deploy it, whereas AppFuse 2.x uses Maven 2 for these tasks. IDE support was improved in 2.0 by leveraging Maven plugins to generate IDE project files. AppFuse 1.x uses [XDoclet](#) and JDK 1.4+.

Unlike other "new project" wizards, the AppFuse wizard creates a number of additional classes and files that implement features, but also serve as examples for the developer. The project is pre-configured to talk to a database, to deploy in an appserver, and to allow logging in.

When AppFuse was first developed, it only supported Struts and Hibernate. In version 2.x, it supports Hibernate, [iBATIS](#) or JPA as [persistence](#) frameworks. For implementing the MVC model, AppFuse is compatible with [JSE](#), Spring MVC, Struts 2 or [Tapestry](#).

Features integrated into AppFuse includes the following:

- [Authentication](#) and [Authorization](#)
- [User Management](#)
- Remember Me (saving your login information so you don't have to login every time)
- [Password Reminder](#)
- [Signup/Registration](#)
- SSL Switching
- E-Mail
- [URL rewriting](#)
- [Skinnability](#)
- Page Decoration
- Templated Layout
- File Upload

This out-of-the-box functionality is one of the main features in AppFuse that separates it from the other "CRUD Generation" frameworks, including [Ruby on Rails](#) and [Grails](#). The aforementioned framework, as well as AppFuse, allow you to create [master/detail](#) pages from database tables or existing model objects.

Google Guice is an [open source software framework](#) for the [Java platform](#) released by [Google](#) under an [Apache license](#). It provides support for [dependency injection](#) using [annotations](#) to configure Java objects.

Guice allows implementation classes to be programmatically bound to an [interface](#), then injected into constructors, methods or fields using an [@Inject](#) annotation. When more than one implementation of the same interface is needed, the user can create custom annotations that identify an implementation, then use that annotation when injecting it.

Pax-Web Pax Web Extender is a set of utilities related to web development under OSGi and it contains the following modus:

- Pax URL - war — war:/war-i: protocol handlers. Automates war deployment.
- Pax Web Extender - War — Pax Web Extender WAR is an extender bundle that makes possible to deploy WAR files into OSGi.

- Pax Web Extender - Whiteboard — Pax Web Extender Whiteboard is an extender bundle that ease the pain of registering servlets, resources, filters and listeners and keeping track of Http Service availability.