Документация

Jahia v6
Обзор архитектуры

Jahia – первое **ПО интеграции веб-содержимого,** объединяющее систему управления веб-содержанием (WCM) с управлением документами и функциями портала.

Jahia

9, route des Jeunes
CH-1227 Les acacias
Geneva, Switzerland

www.jahia.com
The Company website

www.jahia.org
The Community website

Январь 2010

Jahia – обзор архитектуры

В документе представлен высокоуровневый обзор архитектуры корпоративного портала Jahia. Обзор призван помочь читателю разобраться с различными компонентами системы и связями между ними. Также данный обзор может послужить основой для понимания исходного кода системы, который является свободно доступным. Документ содержит много различных схем для более ясного освещения материала. Авторы пытались сделать их как можно более логичными и понятными, не отрицая тот факт, что каждую подсистему портала можно рассматривать с разных сторон и различными способами.

Перевод с английского и редакция: Антон Щастный.

Оглавление

Общая картина

Jahia имеет в своем составе богатый арсенал разнообразных функций и подсистем. Основная идея работы портала заключается в их эффективной интеграции так, чтобы было удобно пользователям.

Ниже на рисунке приведены основные подсистемы портала:

Вкратце рассмотрим их сверху вниз:

- Компоненты пользовательского интерфейса (UI Components): начиная с версии 6.0, Jahia стала использовать AJAX-фреймворк GWT (Google Web Toolkit), предоставляющий широкие возможности повторного использования компонентов интерфейса. Применение фреймворка облегчило разработчикам и интеграторам системы писать дружественные пользовательские интерфейсы, корректно работающие в основных браузерах.
- Кэширование (Caching): для быстрой генерации страниц Jahia поддерживает многоуровневый кэш, начиная с уровня баз данных и до уровня html-страниц. Кэш является кластеро-совместимым и помогает делать систему масштабируемой и производительной. Реализации кэша являются подключаемыми.
- Однократная регистрация (SSO) или технология единого входа: Jahia совместима со всеми основными реализациями механизма SSO (Single Sign-On), включая NTLM, CAS,

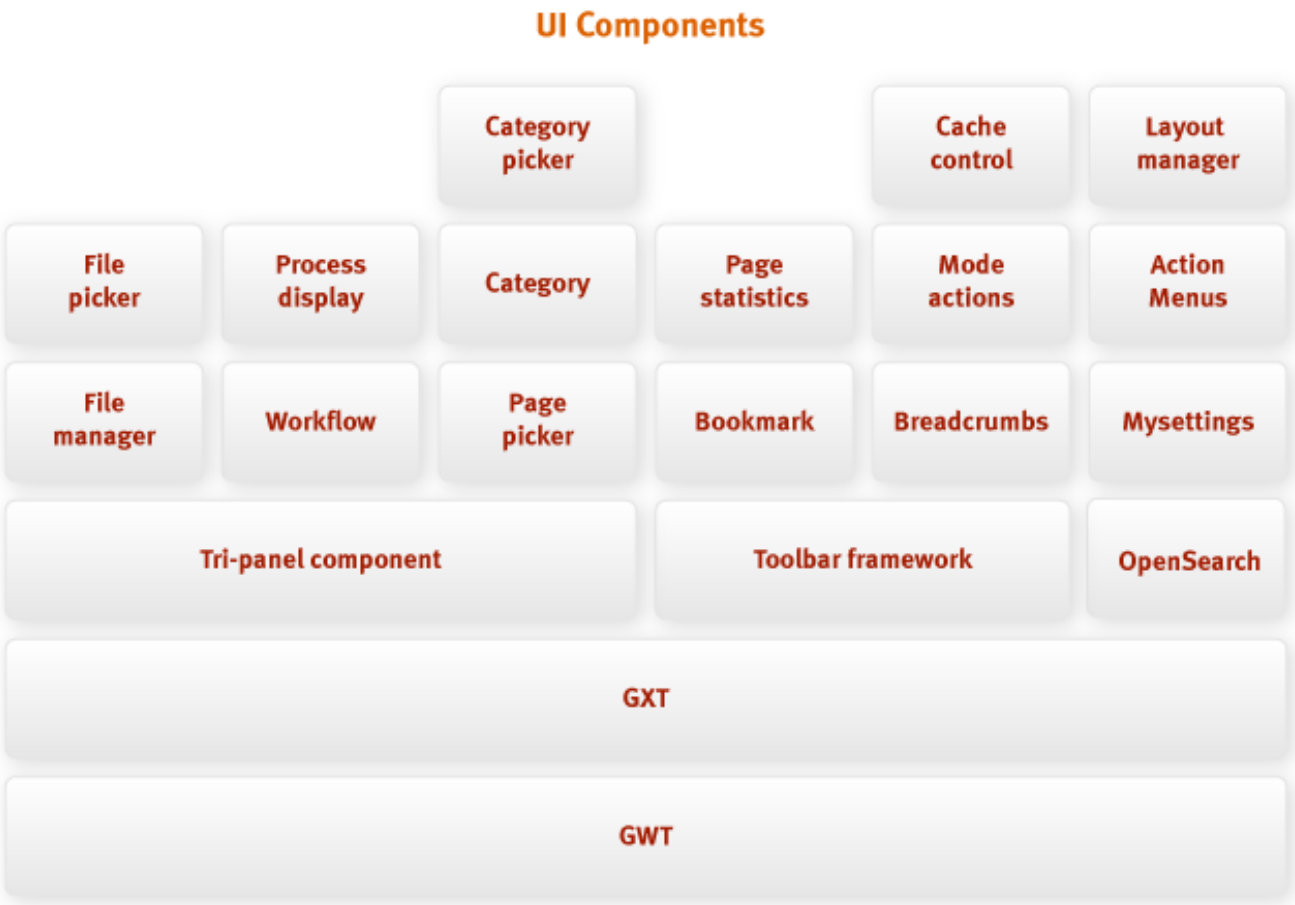
аутентификацию контейнера JEE, и т.д. Подсистема является подключаемой с помощью механизма конвейеров (pipeline mechanism), который будет описан позднее в этом документе.

- Система управления содержимым (WCM): вначале Jahia разрабатывалась как система управления веб-содержимым (Web Content Management System) и поэтому этот компонент является наиболее зрелым. Начиная с версии 2.0 система управления содержимым обладает возможностями редактирования контента прямо на странице, имеет в своем составе типовые объекты содержимого и позволяет использовать портлеты.
- Портал (Portal): Jahia не является портальным сервером в его строгом понимании, так как она предоставляет намного более настраиваемую среду, по сравнению с традиционными серверами. Бэк-энд системы полностью удовлетворяет стандарту JSR-286, поэтому есть возможность легко интегрировать портлеты на веб-страницы системы для их совместного использования с обычными объектами содержимого.
- Файловое хранилище (File repository): Портал интегрирован с файловым хранилищем JCR (Java Content Repository), которое позволяет пользователям системы публиковать двоичные данные на веб-сайты и осуществлять доступ к хранилищу через различные интерфейсы (например WebDAV или Windows Shared Folders).
- Списки контроля доступа (ACL, Access Control Lists): Jahia является модульной системой, широко использующей подсистему контроля доступа. Ее применение дает возможность эффективно разграничивать доступ пользователей к различным подсистемам и службам Jahia. Подсистема контроля доступа доступна по всему порталу.
- Кластеризация (Clustering): Jahia допускает развертывание системы на кластере, таким образом обеспечивая резервирование и распределение нагрузки. В данном документе будет рассмотрена общая архитектура кластера Jahia.
- Система поиска и индексирования (Search & Indexing): в списке основных возможностей большинства систем управления содержимым лежит поддержка поиска и индексирования. Jahia делает возможным поиск и индексирование не только по содержимому, но и во встроенном файловом хранилище. Портал позволяет производить поиск по конкретному семейству объектов содержимого, а также распределять задачи индексирования по нескольким серверам.
- Администрировние (Administration): В системе представлены два отдельных интерфейса управления порталом: для администраторов сервера и администраторов сайта. Это разделение позволяет распределить задачи по поддержке портала по ролям. Интерфейсы дают пользователям возможность управлять всеми подсистемами Jahia, включая администрирование пользователей и груп, разрешений, загруженных шаблонов оформления, портлетов, категорий и так далее…

Последующие низкоуровневые подсистемы не входят в состав портала, но их применение обеспечивает хранение и доступ к информации, которую обрабатывает Jahia.

- Базы данных (Database): Jahia поддерживает все основные СУБД, включая и СУБД с открытым исходным кодом (MySQL, PostgreSQL). Архитектура портала не привязана к конкретной реализации СУБД так как Jahia использует технологию Hibernate ORM.
- Файловая система (Filesystem): файловая система необходима для хранения поискового индекса, массива двоичных данных (BLOB) или массива символьных данных (CLOB). (По умолчанию портал хранит двоичные и символьные данные в БД). Файловая система должна быть распределенной, если планируется использовать кластер, при этом ответственность за корректную работу распределенных функций возлагается на саму файловую систему.
- Облегчённый протокол доступа к каталогам (LDAP): Jahia не требует дополнительной настройки для работы с большинством LDAP-серверов (Lightweight Directory Access Protocol), включая OpenLDAP и ActiveDirectory. Вместе с технологией единого входа LDAP является основным средством для обеспечения комфортного доступа к содержимому конечным пользователям. Например, применив NTLM-аутентификацию и ActiveDirectory, Jahia сможет использовать аутентификацию в домене, которую проходят пользователи Windows при начале работы с компьютером.

Компоненты пользовательского интерфейса



На данном рисунке приведены различные компоненты пользовательского интерфейса, которые предоставляет Jahia.

Начиная с Jahia 6.0 основой для всех компонентов является Google Web Toolkit. Используя GWT, разработчики могут быстро писать и отлаживать AJAX компоненты пользовательского интерфейса. В своей основе Jahia использует библиотеку компонентов GXT с уже готовыми компонентами для работы с таблицами, деревьями и другими стандартными элементами интерфейса.

Компонент tri-panel – стандартный компонент пользовательского интерфейса Jahia, отображающий панель иерархических связей с выводом списка потомков и окном информации.

Компонент панели инструментов используется для быстрого доступа к инструментам, переключения режимов навигации/редактирования содержимого. Панель иснтрументов похожа на те, что используются в стандартных настольных приложениях, кроме того, что инструменты на ней отображаются в зависимости от контекста (прав просмотра и редактирования для текущего пользователя, текущего режима просмотра).

Поверх перечисленных компонентов в Jahia имеется полный набор элементов пользовательского интерфейса: от компонентов управления файлами и до управления категориями, включая компоненты просмотра и навигации по содержимому типа «хлебных крошек» (breadcrumbs) и закладок.

Аутентификация и авторизация

Jahia позволяет использовать следующие механизмы аутентификации:

- CAS SSO (Single Sign-On с CAS, Central Authentication Service);
- NTLM (NT LAN Manager от Microsoft для Windows NT);
- аутентификация контейнера JEE;
- подключаемый конвейер аутентификации, с помощью которого можно с легкостью добавить поддержку других технологий однократной регистрации.

После того как однократно пользователь будет идентифицирован, работу продолжит система разграничения прав. Состав системы разграничения прав:

- списки контроля доступа для объектов содержимого;
- разрешения на все действия пользователя;
- подключаемые службы для пользователей (user services);
- подключаемые службы для групп (group services);
- совместимость с LDAP;
- хранилище базы пользователей в БД (в настройках по умолчанию).

На рисунке выше показано, как различные уровни подсистемы (в том числе и подключаемые интерфейсы) взаимодействуют между собой. Служба маршрутизации позволяет использовать единые интерфейсы доступа к БД и LDAP-хранилищу базы пользователей. Реализовав интерфейсы поставщика сервисов вы сможете добавлять дополнительные службы.

Шаблоны оформления

От системы шаблонов Jahia зависит то, как информация будет отображаться на странице.

Шаблоны оформления группируют в набор. Набор шаблонов в последующем загружают на сервер. Внешний вид сайта будет зависеть от того, какой набор шаблонов вы выбрали. Один набор может расширять другой, «наследуя» его.

Состав набора шаблонов:

- файлы описания содержания (content definition files, CND);
- JSP страницы;
- темы и оболочки оформления (skins);
- дескриптор развертывания (deployment descriptor) для набора шаблонов (обязательно);
- ресурсы (resource bundles);
- Java-классы.

При запуске Jahia определит все имеющиеся в системе шаблоны (папка WEB-INF/var/shared_templates) и развернет их на сервере. В режиме разработки (development mode) любые изменения в дескрипторе развертывания повлекут перезагрузку набора шаблонов, этим самым позволив мнгновенно видеть изменения в оформлении на сервере.

Как было выше показано на рисунке, для оформления портала в наборах шаблонов можно применить достаточно много различных технологий:

- Библиотеки тегов: в целях облегчения поддержки и обновления шаблонов в Jahia решили максимально отказаться от использования скриплетов. Для этого в систему включили богатые библиотеки тегов, прекрасно работающие со стандартным набором тегов JSTL. Данные библиотеки предоставляют функции по работе с елементами пользовательского интерфейса (например работа с панелью инструментов), возможности делать запросы к объектам содержимого, механизмы подключения родительских шаблонов тегом Include (поддержка наследования). Подробная информация по библиотекам тегов приведена в Руководстве по разработке шаблонов (Template developer guide).

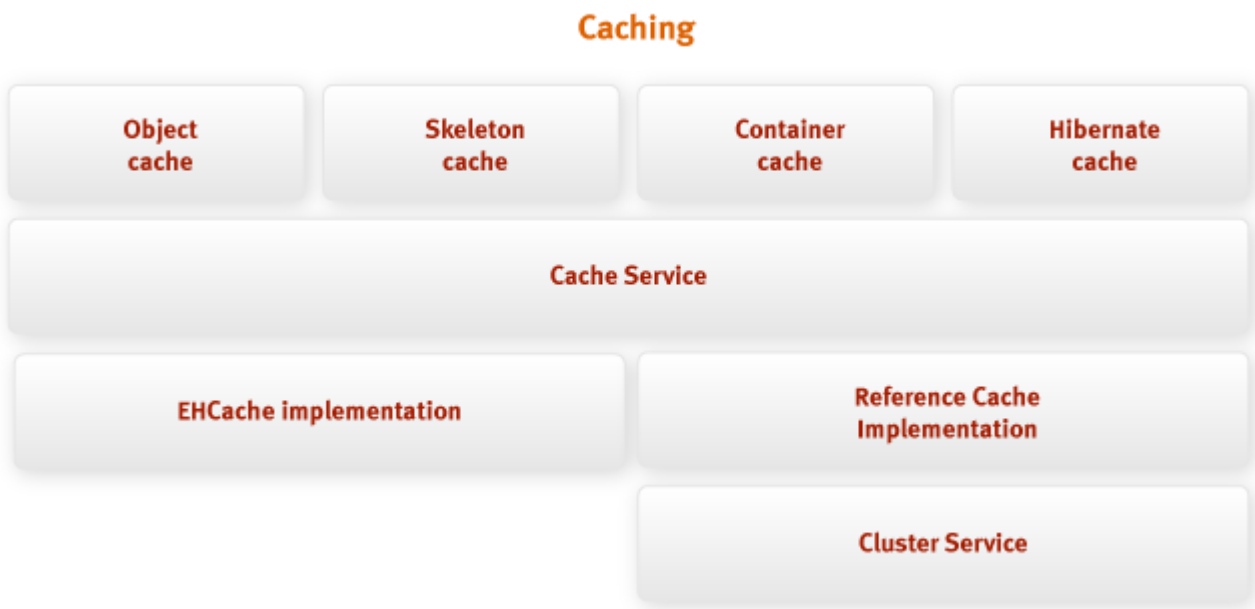
- Механизм наследования: для облегчения разработки новых наборов в шаблонах можно использовать механизм наследования, подобный наследованию в языке Java. В случае разработки нового набора с использованием механизма наследования нужно переопределить только те шаблоны, в которых есть изменения, а остальные будут унаследованы от шаблона-родителя. Все это также применимо к файлам, которые включены тегом Include (просматривается шаблон-родитель на предмет наличия файла, который нужно подключить, если он не был переопределен локально). Механизм наследования позволит администраторам гарантировать согласованность оформления на большом количестве сайтов портала, при этом сохраняя возможность использовать совершенно различные наборы шаблонов на каждом конкретном сайте.

- Развертывание: как было сказано ранее, наборы шаблонов могут быть развернуты в Jahia при запуске сервера. Поэтому необходимо просто добавлять новые наборы в инсталляцию Jahia. Однажды загрузив набор, вы его делаете доступным в системе. Ну и администраторы сайта в свою очередь уже будут определять наборы, которые будут использоваться на сайтах портала: либо это будут различные наборы шаблонов для каждого сайта, либо в системе будет один набор-родитель, от которого все остальные шаблоны будут наследовать оформление.

- Темы и оболочки оформления (skins): темы используются для придания одному набору шаблонов разного вида. В качестве примера можно привести набор, в котором есть темы по временам года: зимняя тема, весенняя, летняя и осенняя темы. Тема может быть установлена администратором, либо предоставлена на выбор пользователю. Оболочки позволяют авторам контента по-разному оформлять объекты содержимого одного типа.

- Описание контента (content definitions): Jahia использует стандартный формат описания содержимого JCR, который был расширен (интернационализацией, строгой типизацией элементов контента) для более мощной работы с содержимым.

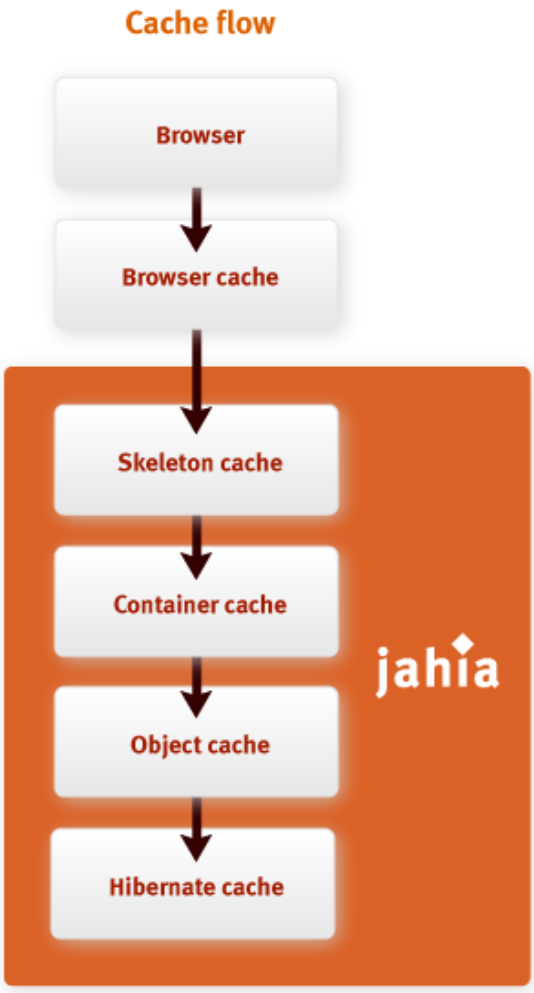
Кэширование



Для высокопроизводительных веб-систем наподобие Jahia кэширование необходимо для избежания повторной генерации динамического содержимого при высоких нагрузках. На рисунке показана архитектура подсистемы кэширования. Все высокоуровневые кэши базируются на службе кэширования (cache service), представляющей реализации кэша. Ниже этой службы могут использоваться различные реализации кэша, некоторые из них могут зависеть от других служб Jahia. Например кэш ссылок («Reference Cache») зависит от службы кластеров «Cluster service», позволяющей обмениваться сообщениями в кластере Jahia.

ТИПЫ КЭША

Все типы высокоуровневых кэшей используют одну и ту же службу кэширования, которая представляет реализации кэша. Начиная с Jahia 5.0SP4 в системе появилась возможность применять различные реализации кэша для различных типов кэша, делая систему гибкой для конфигурирования. Например, можно использовать *EHCache* для кэша пользовательских html-страниц и использовать *ReferenceCache* для остальных типов кэшей. Это конфигурация по-умолчанию для Jahia 5.0SP4 и Jahia 6.0. Естественно также есть возможность инегрировать другие реализации кэша с помощью интерфейса поставщика служб (SPI, service provider interface), который предоставляет Jahia.



Для оптимизации производительности доставки веб-страниц Jahia использует несколько уровней кэша:

- кэш браузера,
- кэш пользовательских html-страниц (front-end HTML caches),
- кэш объектов,
- кэш базы данных.

Каждый из этих уровней решает свою задачу, преследуя единую цель: чтобы значения вычислялись только один раз.

КЭШ БРАУЗЕРА

Несмотря на то, что данный кэш не входит в состав портала Jahia, а является частью браузера, именно он играет критическую роль в гарантировании хорошей производительности системы для конечного пользователя. Например, использование в Jahia фреймворка GWT позволило интенсивно кэшировать AJAX-код в кэше браузера, тем самым избавив портал от необходимости

перегружать скрипты, которые не менялись. Jahia корректно работает с кэшем тех страниц, которые были изменены и контролирует время жизни страничного кэша, чтобы не обновлять без надобности редко меняющееся содержимое.

КЭШ ПОЛЬЗОВАТЕЛЬСКИХ HTML-СТРАНИЦ

Исторически сложилось так, что в Jahia имеется много реализаций для кэша html-страниц. Первой реализацией был полностраничный html-кэш (full-page HTML cache). Данный кэш очень эффективен для простых страниц, если они уже есть в кэше; и он не очень подходит для страниц с различающимися фрагментами (например отображающие информацию о пользователе на странице).

В Jahia 5 мы представили сервер кэша ESI (ESI cache server), который добавил возможность кэшировать фрагменты html-страниц. Для эффективного использования данной технологии требуется отдельный сервер кэша, который будет выполняться на отдельной виртуальной машине. Являясь более эффективной технологией для генерации динамических страниц по сравнению с полностраничным кэшем, система ESI страдает от проблем с внутрисерверными коммуникациями, которые нужно очень хитро настраивать для нормальной работы. Также применение кэша ESI требует от интеграторов хорошего знания модели фрагментарного кэширования (fragment caching model), что не всегда им нравится.

Jahia 6 взяла от обоих подходов лучшее, комбинируя эффективность встроенного полностраничного кэша с поддержкой фрагментарного кэширования от ESI cache server. Эта реализация кэша была названа «Кэш контейнера» («Container Cache»), она применяет фрагментарное кэширование на уровне контейнера, делая взаимодействие с шаблонами очень простым. Разработчикам шаблонов не придется добавлять специальную разметку для настройки корректного фрагментарного кеширования. Даже если появится необходимость управлять генерированием фрагментов, то сделать это будет намного легче, чем в предыдущих версиях Jahia.

Каркасный кэш («Skeleton Cache») – еще одна реализация кэша html-страниц, кэширующая все остальное (кроме фрагментов) содержимое. Группировкой двух данных подсистем кэширования по производительности можно получить эквивалент полностраничного html-кэша, но уже с возможностью кэшировать фрагменты.

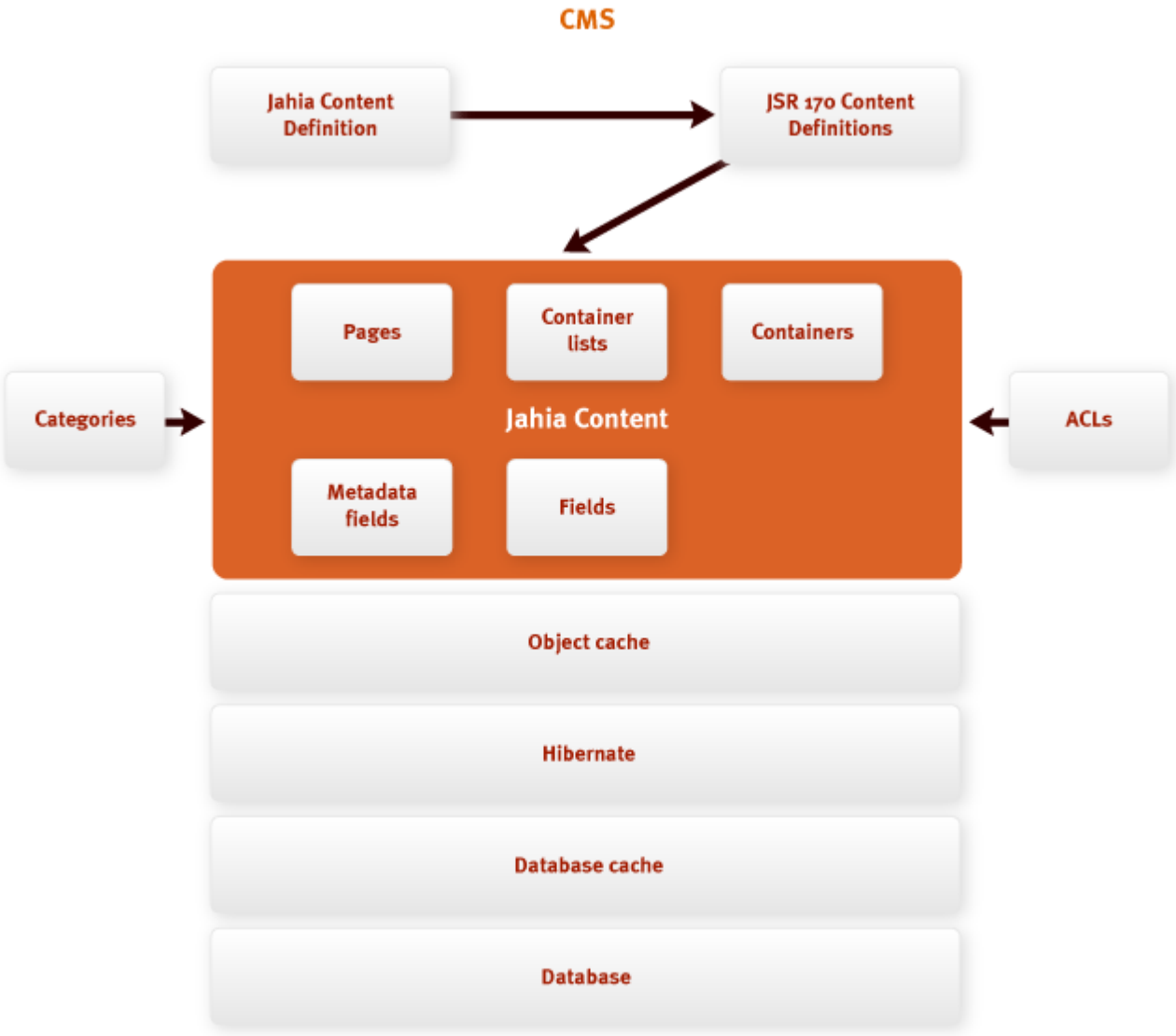
КЭШ ОБЪЕКТОВ

Кэш объектов лежит ниже подсистемы кэша html-страниц. Этот уровень кэша обслуживает все Java-сущности системы: объекты содержимого, объекты пользователей, групп, категорий, описания контента и т.д. Кэш объектов существует над кэшем базы данных для избежания реконструирования объектов при каждом запросе к модели. Уровень является прозрачным для разработчиков, взаимодействовать с ним через Jahia API придется только в том случае, если изменения объектов не влекут за собой автоматическое обновление кэша.

КЭШ БАЗЫ ДАННЫХ

Последний уровень кэша – уровень базы данных, обеспечивающий минимализацию запросов к БД. Он очень важен. При работе с базой данных происходит сериализация/десериализация объектов, нагружающая сеть и ресурсы системы так, что накладные расходы на запросы к БД могут оказаться весьма существенными. Этот уровень кэша полностью реализован средствами Hibernate ORM.

Объекты содержимого



Перечень объектов содержимого в Jahia:

- страницы (pages), включая страницы контента, внутренние и внешние ссылки;
- списки однотипных контейнеров (container lists);
- контейнеры (содержат поля разных типов);
- простые поля (fields) и поля метаданных (metadata fields, могут быть различного типа: SmallText, BigText, Integer, Date, File reference, и т.д.).

Все эти объекты имеют общие черты:

- их можно разбивать на категории;

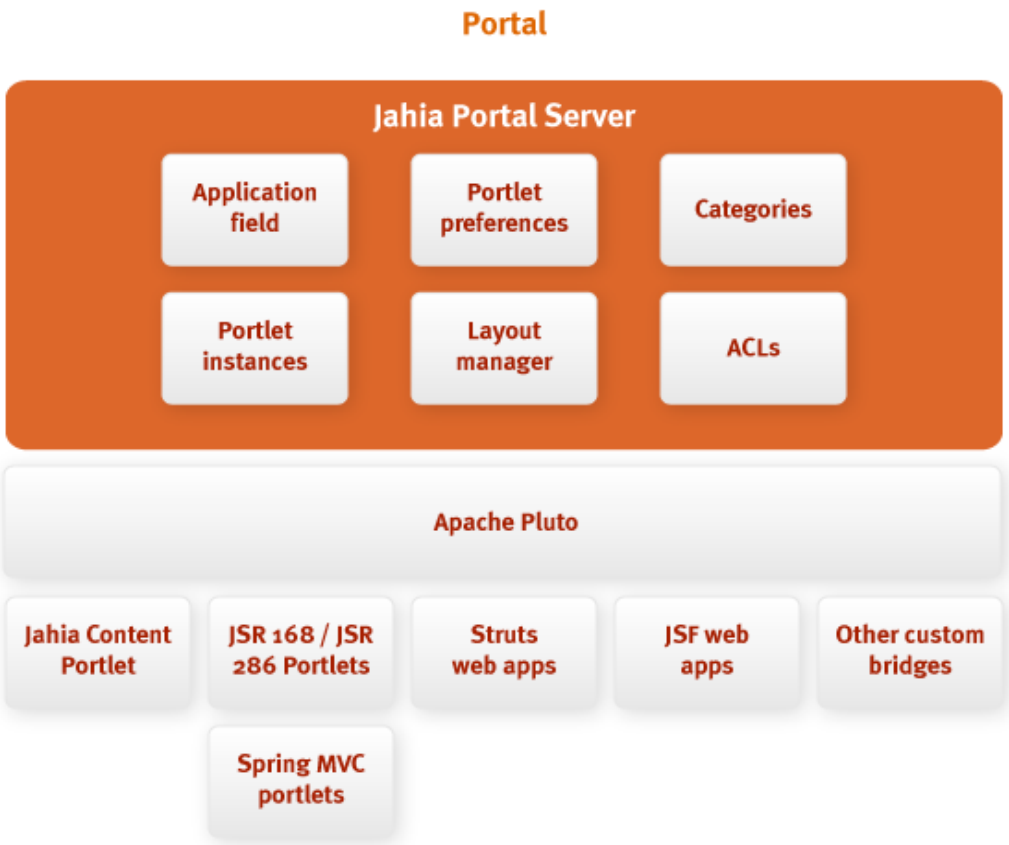
- доступ к ним может регламентироваться с помощью списков контроля доступа (ACL);
- они описываются с помощью файлов описания контента (content definitions files, cnd-файлов);

Формат файлов описания удовлетворяет стандарту Java Content Repository (JCR) на файлы CND с добавлением следующих возможностей:

- мультиязычные значения полей;
- более точный ввод типов полей (property types);
- возможность указания ресурсов (resource bundle support);
- списки выбора (choice lists);
- многое другое …

Объекты содержимого загружаются из базы данных и различных уровней кэша. Затем они передаются в подсистему шаблонов, в которой интеграторы могут окончательно отрисовать страницу с использованием библиотек тегов Jahia.

Портал



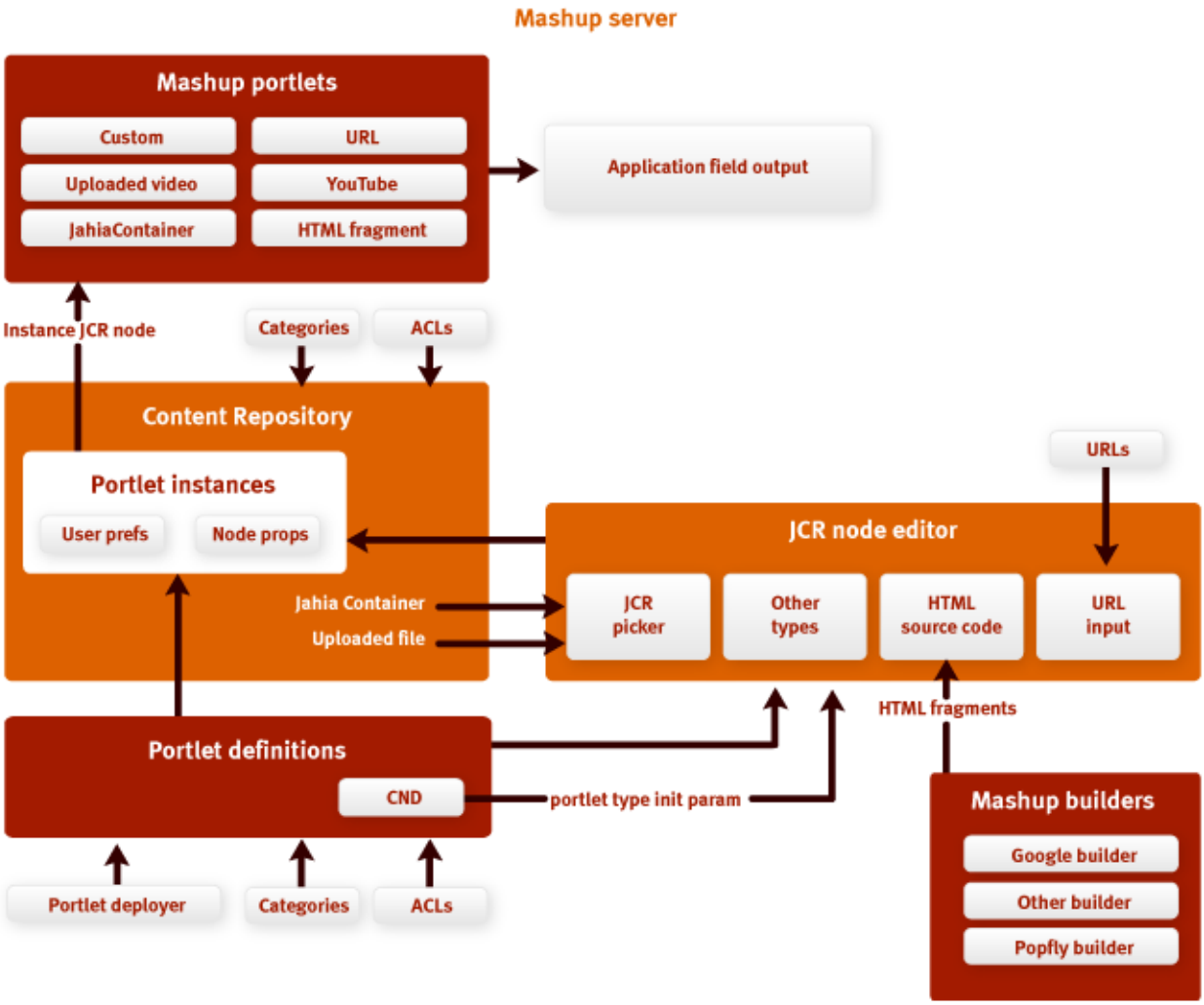
В состав Jahia включен сервер порталов, основанный на Apache Pluto – эталонной реализации спецификации Java портлетов (JSR-286 Portlet API). Целью применения Apache Pluto является предоставление возможности интеграторам применять портлеты на страницах. Любой портлет, следующий стандартам, может быть с легкостью встроен в портал Jahia. На рисунке выше приведены различные типы портлетов, которые могут использоваться в Jahia.

Одно из полей объекта содержимого – поле приложения (application field), ссылающееся на портлет. Это поле позволяет интеграторам определить место в структурированном дереве веб-содержимого, где приложение может появиться. Портлеты также могут быть разбиты на категории и доступ к ним можно контролировать с помощью списков контроля доступа.

Jahia 6 также использует портлеты внутренне. Внутренние портлеты могут включать в себя описания контента, используются они для генерации различных слоёв объектов содержимого. Слои в портале используются в качестве функциональных блоков для построения веб-страниц. Их можно использовать повторно на различных страницах.

Этот механизм получил название Мэшап — концепции построения веб-приложений путем комбинирования функциональности различных программных интерфейсов и источников данных. Для построения блоков в портальном сервере Jahia используется мэшап-сервер (Mashup Server).

Мэшап-сервер



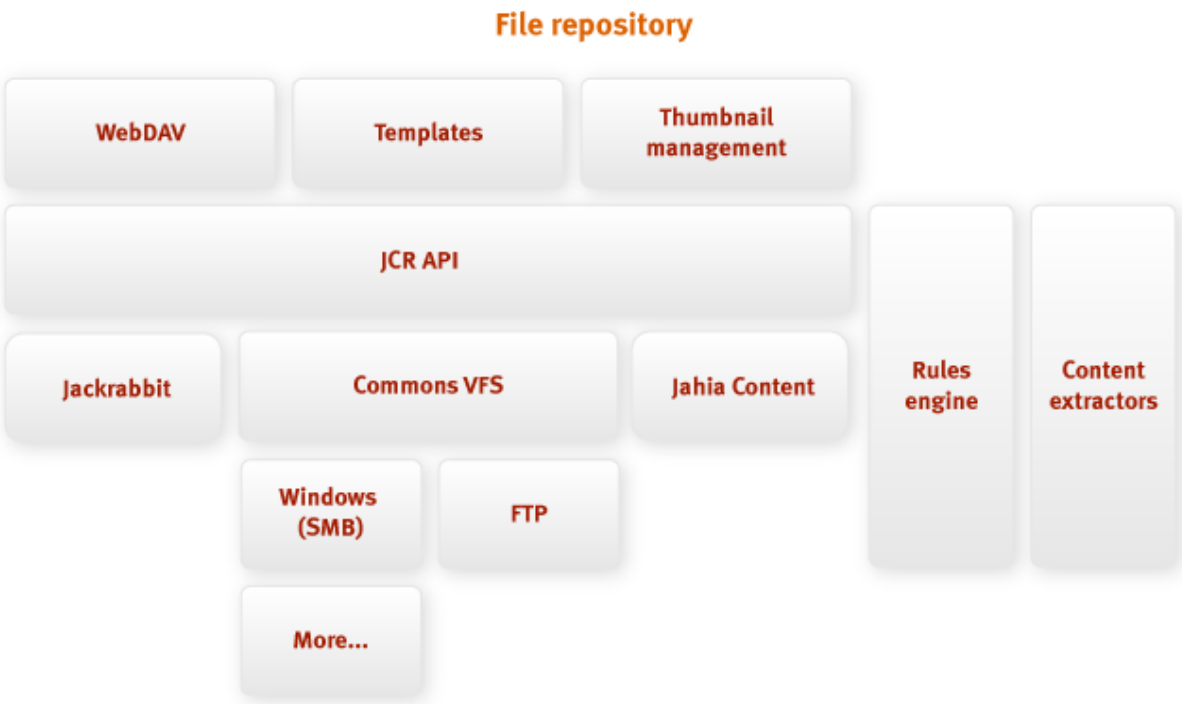
В последних версиях Jahia в системе стали чаще использовать Javascript, поэтому для возможности определять и повторно использовать мэшапы на веб-сайтах Jahia в состав портала включили новый мэшап-сервер.

Мэшап-сервер надстроен над сервером порталов, он позволяет с легкостью подключать новые источники данных (mashup types), просто загружая в систему новые портлеты. На рисунке выше мы показали различные источники данных – мэшапы: Google Gadgets, видео с YouTube, фрагменты html и Javascript. Эти источники фактически являются типами портлетов включающими в себя описание контента. По описанию контента пользовательский интерфейс будет автоматически настраивать и отрисовывать мэшап. Администраторы сайта смогут иметь возможность строить мэшапы, разбивать их по категориям, ограничивать доступ с помощью списков контроля доступа и позволять авторам контента использовать их в своих разделах и в мерах своей компетенции в системе. Разработчкики могут группировать мэшапы в пакеты (как

обыкновенные Java-портлеты) или просто повторно использовать уже существующие портлеты, если они удовлетворяют нуждам (например html-мэшап достаточно общий и может подойти во множестве случаев). Мэшапы также можно импортировать в систему, используя функции импорта/экспорта Jahia.

Для улучшения системы разработчики Jahia всегда с интересом смотрят на новые технологии. Стандарты для мэшапов сейчас еще только разрабатываются, так что Jahia будет ингегрировать новые возможности данной технологии по мере ее развития.

Файловое хранилище



Файловое хранилище Jahia было полностью переписано в версии 6. В Jahia 5 хранилище было основано на использовании библиотеки Apache Slide library, которая на то время была де-факто стандартом файловых репозитариев с открытым исходным кодом.

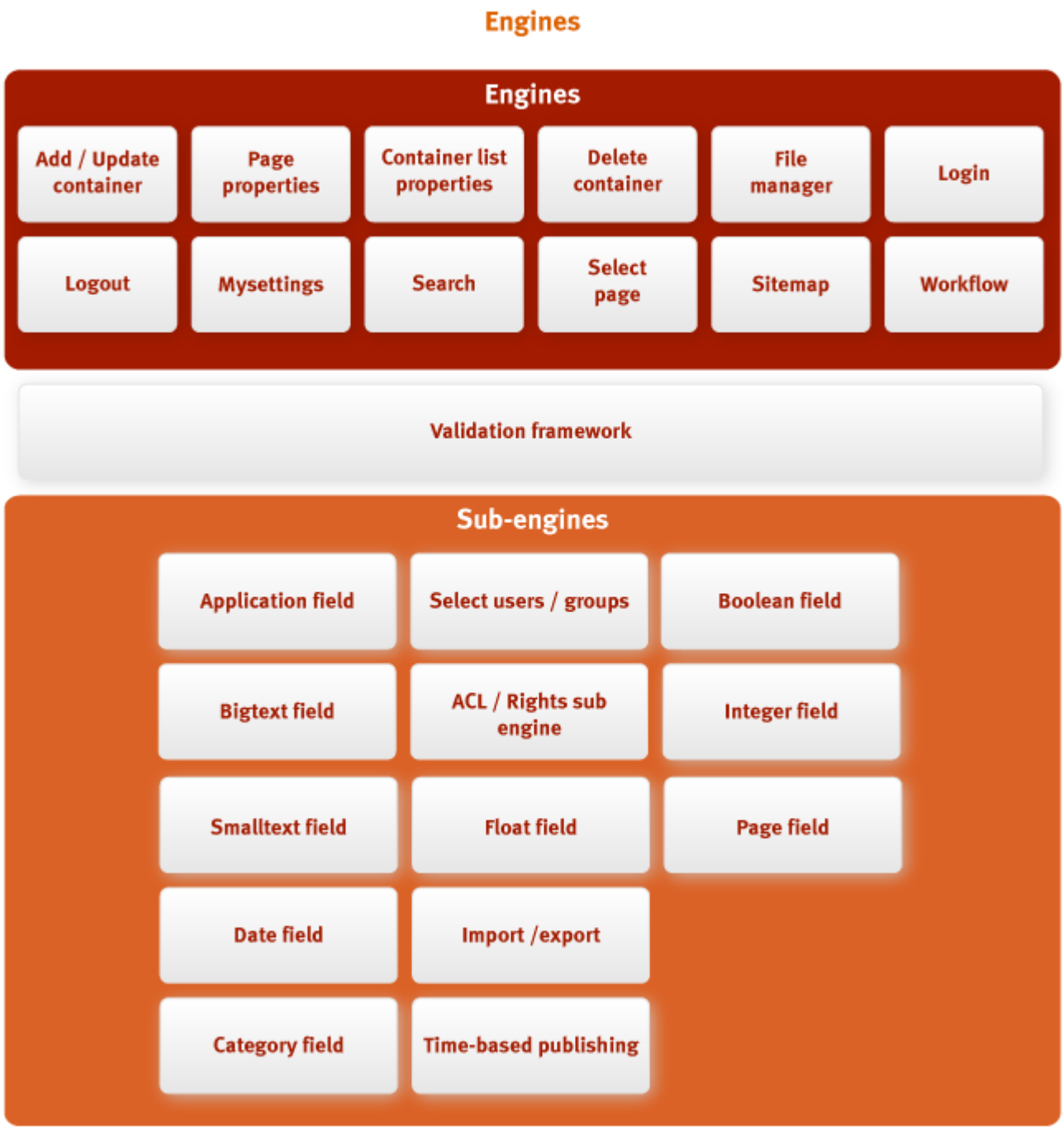
Но всвязи с бурным развитием и хорошим качеством Apache Jackrabbit – реализации спецификации Java Content Repository, Jahia 6 стала использовать ее в качестве стандартного файлового хранилища и строить поверх данной библиотеки свои сервисы. На самом деле архитектура Jackrabbit не предполагает тесной зависимости от ее использования, а использует стандартный JCR API для возможности предоставления доступа к различным репозитариям. В Jahia 6 стало возможным иметь доступ к файловым хранилищам CIFS/SMB. Некоторые другие развивающиеся реализации доступны в песочницах репозитариев (sandbox repositories), среди них коннекторы FTP, Alfresco, Exo Platform, Nuxeo.

На рисунке выше вы можете отметить, что содержимое Jahia (Jahia Content) доступно в качестве JCR-провайдера, это тоже новая возможность Jahia 6. Пока что в данном направлении проведена только первоначальная работа по совместимости и интерфейс к данной реализации не может похвастаться высокой производительностью, но работы по его улучшению будут ведутся постоянно от версии к версии.

Поверх службы файлового хранилища интерфейсы отдают контент с помощью различных технологий: WebDAV, просмотр файлов в шаблонах и в пользовательском AJAX-интерфейсе

Jahia. Вместе с хранилищем могут функционировать дополнительные службы портала: Rules engine – для задания правил и прав доступа к файлам, Thumbnail management – для генерирования миниатюр изображений (thumbnail), Content extractors – для извлечения метаданных.

Инструменты и средства (движки)



Инструменты и средства в Jahia принято называть движками (engines). Движки Jahia можно сравнить с объектами Action в Struts. Это блоки логики приложения, каждый их которых выполняет конкретную задачу. Субдвижки (sub-engines) – еще более мелкие блоки, контролирующие взаимодействие с примитивами пользовательского интерфейса (например контролирующие редактирование поля).

Наиболее важный движок – это движок ядра («Core engine»), отвечающий за генерирование страницы содержимого, он вызывается каждый раз, когда портал должен отобразить страницу. Названия других движков говорят сами за себя: например движки Login и Logout отвечают за отрисовку пользовательского интерфейса входа и выхода из системы и обработку данных, введенных пользователем при этих действиях.

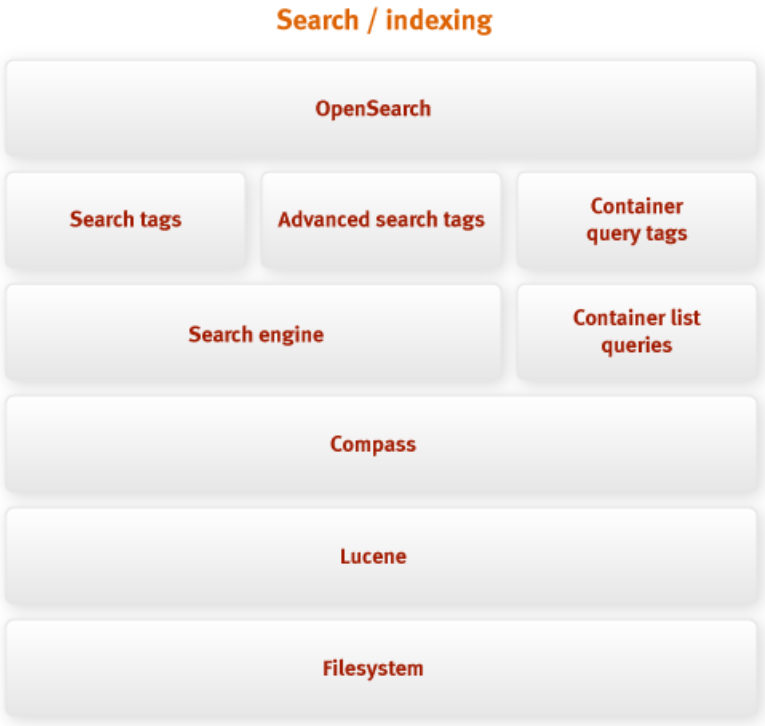
Движки редактирования (обработки) контента интегрированы с фреймворком валидации (validation framework), что позволяет интеграторам задавать правила проверки введенных пользователем данных.

Движки также интегрированы с пользовательским AJAX интерфейсом. Цель применения AJAX – сделать пользовательский интерфейс современным, построенным на компонентах, которые можно многократно повторно использовать. Интеграторы смогут применять уже существующие многочисленные библиотеки компонентов при построении приложений. Работа в данном направлении начата в Jahia 6 применением компонентов GWT.

Субсредства в Jahia используются для отображения интерфейсов редактирования контейнеров. Контейнер может содержать в себе поля разного типа. При редактировании поля какого-либо типа движок редактирования контейнера (container edition engine) обращается к субдвижку, отвечающему за этот тип поля. Например, поле типа File, представляющее файл, будет предоставлять UI-интерфейс, позволющий пользователю просматривать директории и выбирать файлы, в то время как, поле типа Small text просто будет отображать поле для ввода текста.

Jahia обрабатывает запросы к движкам, используя параметр «/engineName» в URL. Если параметр без значения – запрос принимает движок ядра. Строковые значения, которые используются для именования движков, объявляются самими движками, а в классе JahiaEngine, метод getName() будет определять ключ для механизма разрешения параметра «/engineName».

Подсистема поиска и индексирования



Подсистема поиска и индексирования Jahia в данном документе будет представлена лишь обзорно – это очень обширная подсистема. Выше на рисунке мы проиллюстрировали основные составляющие данной подсистемы.

Новшеством Jahia 6 является интеграция стандарта OpenSearch. Jahia может играть роль как потребителя, так и как генератора OpenSearch-запросов. Это значит, что пользователи могут, например, интегрировать Jahia в панель поиска своего браузера и напрямую делать поисковые запросы из панели браузера. С другой стороны, Jahia может пользоваться услугами различных OpenSearch-провайдеров и предоставлять на одной странице сведенные результаты для поиска по Jahia, Google, MSN и т.п.

Что касается более традиционных решений веб-поиска, Jahia предлагает библиотеки тэгов для поиска по содержимому либо в простом формате полнотекстового поиска, либо используя расширенные поисковые запросы. Для облегчения формировния расширенных поисковых запросов в UI-компоненты Jahia встроены удобные селекторы параметров запросов. Эти компоненты сделаны с использованием AJAX-фреймворка и являются подобными тем, которые применялись при введении данных в интерфейсе входа в систему.

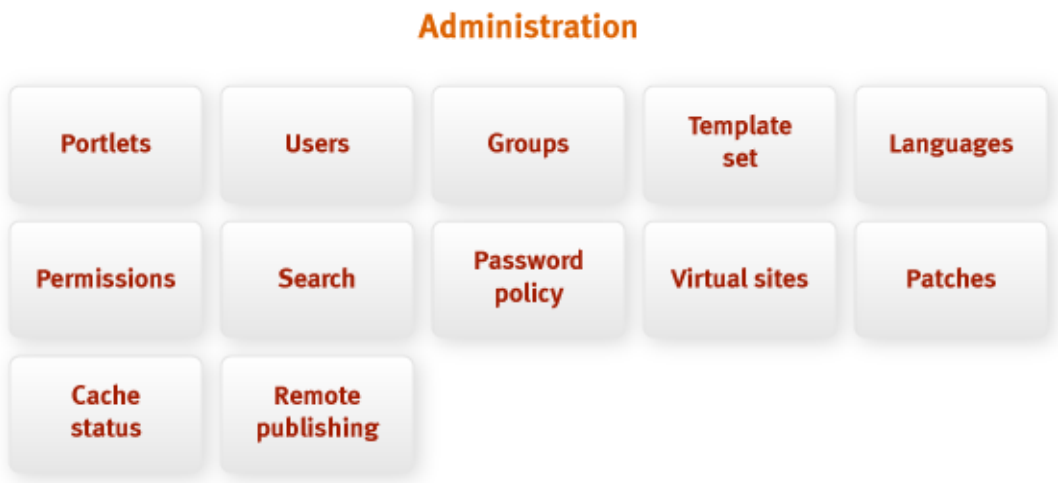
Другая альтернатива для запросов к объектам содержимого – напрямую встраивать запросы на уровне шаблонов, а пользователи на экране будут видеть результат уже подготовленного запроса. Это называется «Запросы контейнера» (Container queries) и использоваться они могут,

например, для загрузки последних пяти объектов содержимого с типа «Новость» при выводе на страницу последних пяти новостей.

В основе всех этих технологий формирования пользовательских запросов лежат серверные системы, непосредственно выполняющие поиск и индексирование контента. Поисковый механизм портала дает возможность осуществлять полнотекстовый поиск, поиск с уточняющими запросами как по содержимому Jahia, так и по файловому содержимому.

Основа подсистемы поиска и индексирования – opensource фреймворки Compass и Apache Lucene. Они хранят свой индекс, используя средства файловой системы, но также имеют средства для хранения информации в базах данных или в распределенных файловых системах. Стандартная реализация подсистемы с использованием файловой системы на данный момент наиболее ефективная и готовая к использованию (production-ready), так что Jahia поставляется в комплекте с этой реализацией.

Администрирование

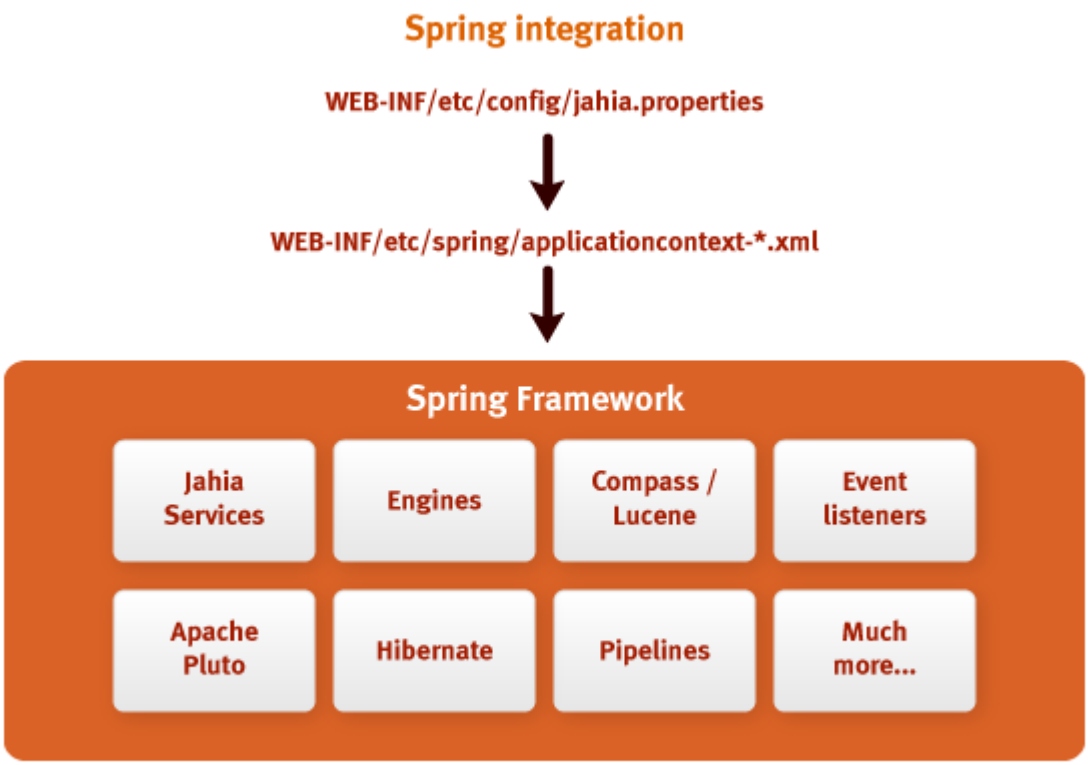


Интерфейс администрирования – это средство управления всеми различными подсистемами Jahia из одного места и для администраторов сайта, и для администраторов сервера. Интерфейс администрирования позволяет администратором выполнять все виды задач:

- создавать, обновлять, удалять пользователей, группы, сайты и категории;
- устанавливать глобальные разрешения;
- устанавливать разрешения на шаблоны и портлеты, разбивать шаблоны и портлеты на категории;
- просматривать статус внутреннего кэша и сбрасывать его;
- устанавливать политику назначения паролей;
- определять языки сайта;
- настраивать возможность удаленной публикации содержимого;
- выполнять другие функции, детально описанные в Руководстве администратора (Administrator’s guide).

Интерфейс администрирования основан на сервлете-диспетчере и коллекции классов, контролирующих UI для различных настроек сервера. Также для удобства администраторов интерфейс поддерживает компоненты пользовательского интерфейса на AJAX.

Интеграция со Spring



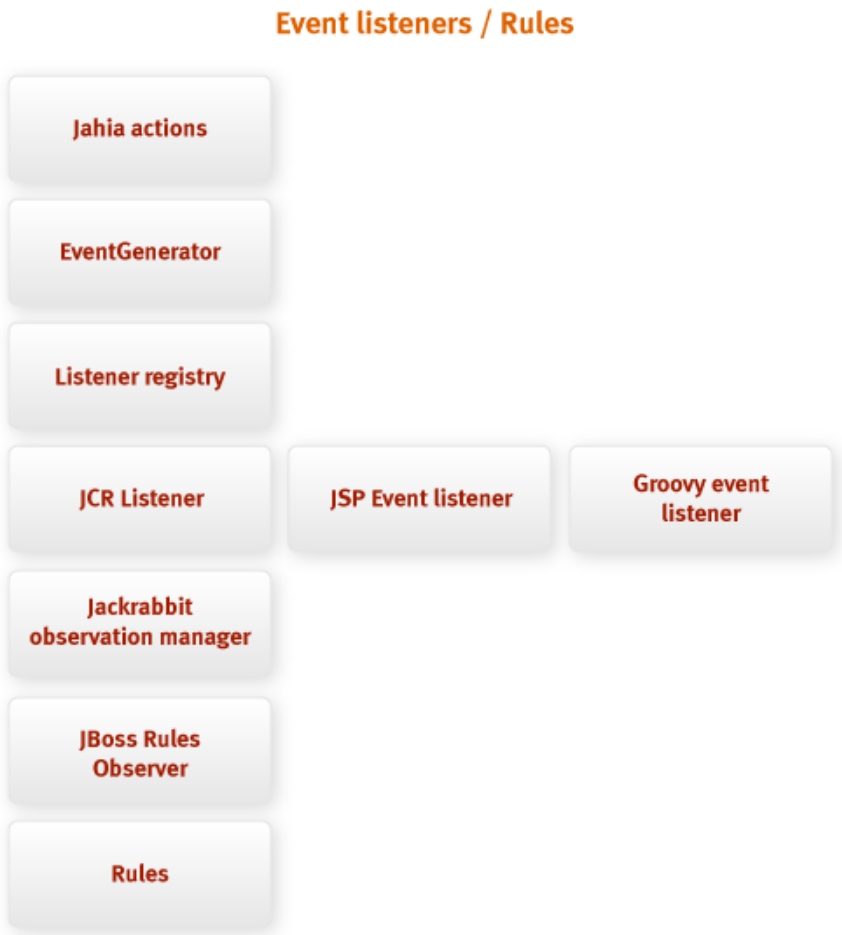
Разработчики Jahia любят Spring. Начиная с пятой версии системы Jahia была интегрирована со Spring Framework для облегчения построения целостных, быстрых и гибких портальных решений.

Для подключения подсистем Jahia к многочисленным сервисам, интеграторам теперь просто следует использовать хорошо известные механизмы настроек и внедрения зависимостей (dependency injection), которые предоставляет Spring.

Рисунок выше поясняет настройки, используя которые с помощью Spring можно инициализировать все различные подсистемы Jahia. Внешние библиотеки портала: Apache Pluto, Hibernate, Apache Lucene – также работают со Spring, что делает возможным реализовывать стройную архитектуру портала.

Опыт интеграции Jahia со Spring был успешным и зависимостей системы от Spring практически не появилось, что делает возможным в дальнейшем в случае необходимости свободно мигрировать на другой фреймворк.

Слушатели событий и правила

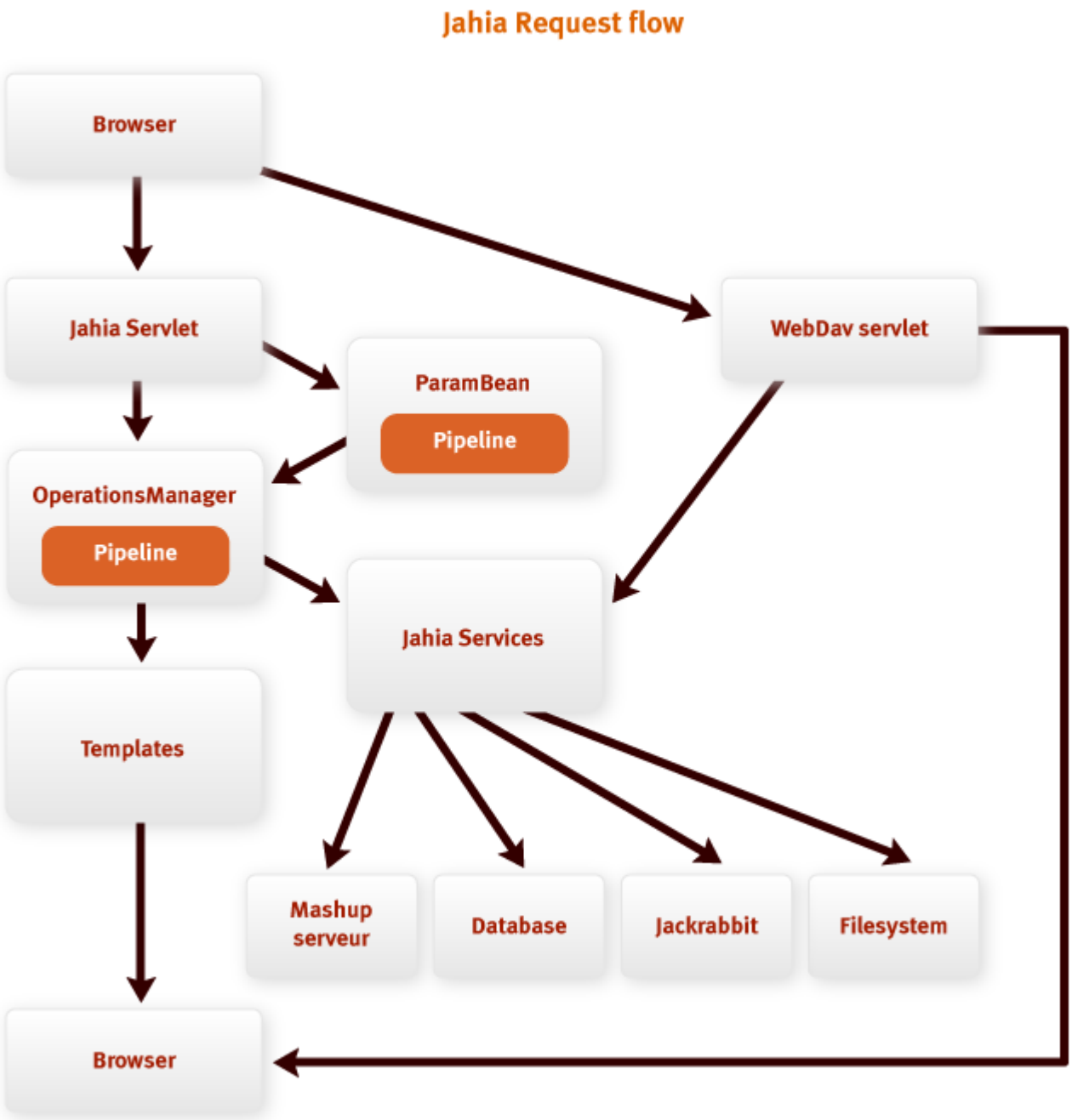


Действия, выполняемые пользователями портала, порождают события, которые могут использоваться для выполнения определённых задач. Jahia позволяет интеграторам регистрировать свои слушатели событий. На приведенном выше графике показано, как события, передаются по различным подсистемам и как они могут быть использованы разработчиками.

Событие порождается при выполнении некоторого действия порталом. Это может быть команда пользователя, действие по расписанию, фоновый процесс. Действие вызывает генератор события (event generator) для создания объекта события. Данный объект далее будет передан в реестр слушателей (listeners registry), который зарегистрированным слушателям будет вызывать необходимые классы. Некоторые из слушателей событий являются шлюзами к другим слушателям, использующим другие технологии. Есть возможность использовать слушатель событий Groovy (Groovy event listener), позволяющий интегратором использовать язык Groovy в реализациях слушателей; слушатель событий JSP (JSP event listener), делающий то же для JSP-страниц.

Еще один специфический слушатель – менеджер наблюдения JCR (JCR observation manager), который связует механизм событий Jahia со стандартной моделью наблюдения JCR (JCR observation). Под данной реализацией находится наблюдатель JBoss Rules (JBoss Rules observer), позволяющий интеграторам использовать правила (rules) для опработки событий.

Обработка запросов в Jahia



Выше представлена схема обработки системой Jahia запроса от браузера. Схема может использоваться разработчиками для определения подсистем портала, задействованных в процессе обработки конкретного запроса.

Запрос браузера может быть направлен либо на Jahia servlet, либо на WebDAV servlet (если запрашивается бинарный файл).

Во втором случае (WebDAV servlet) запрос будет перенаправлен в службу файлового хранилища, пройдет валидацию на право доступа к запрашиваемому ресурсу и потом результат будет возвращен обратно в браузер.

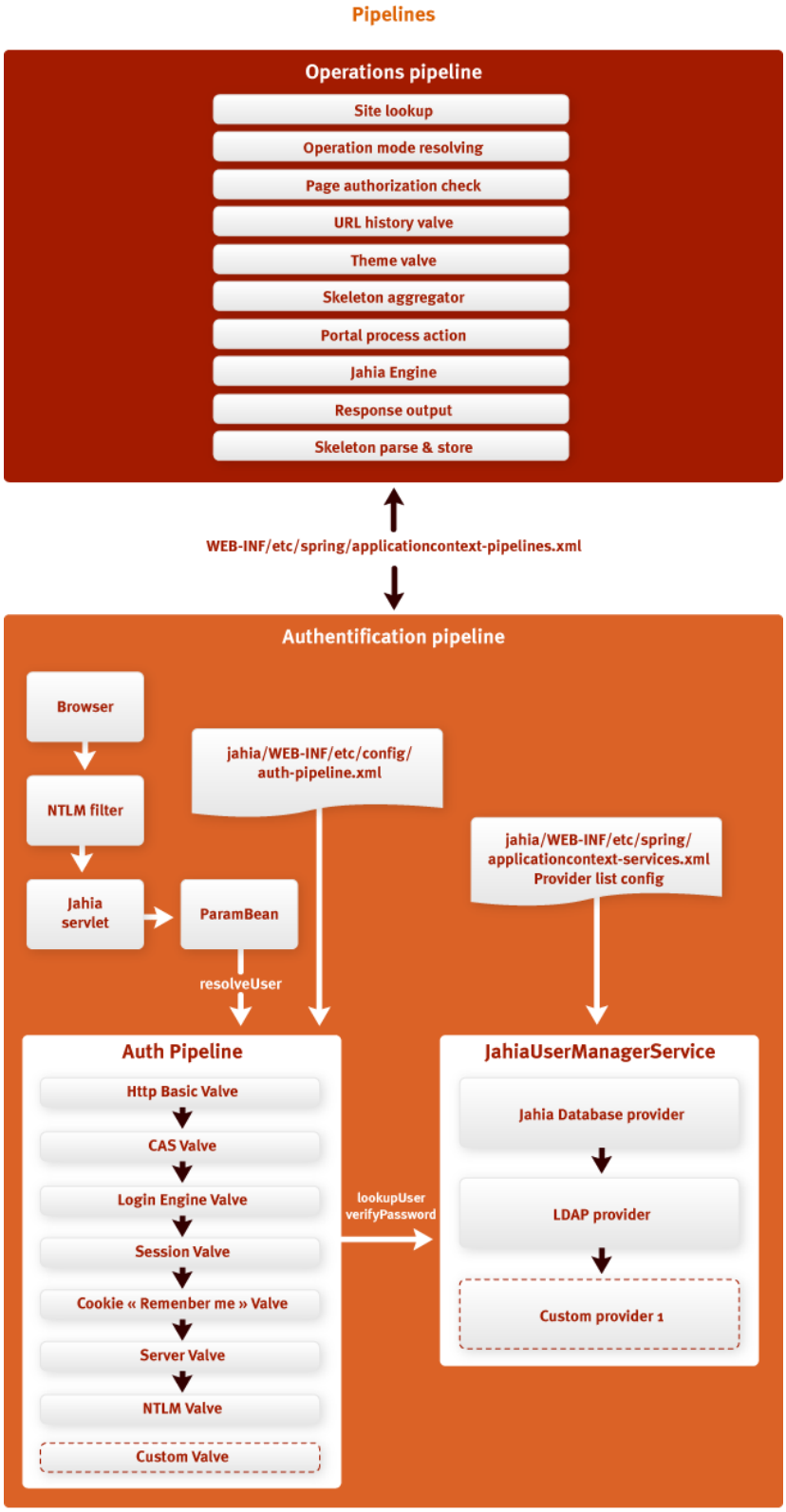
В первом случае будет вызван Jahia servlet. Jahia немедленно создаст объект контекста (context object) под названием «ParamBean». Данный будет содержать в себе всю информацию о запросе, включая такие вещи как запрошеная страница, текущий пользователь, язык просмотра и так далее. Внутренне ParamBean использует механизм конвейерной обработки (pipeline mechanism) для разрешения пользователя. Механизм настраиваемый для интеграции других технологий разрешения пользователей, детали этого механизма описаны в разделе «Конвейеры».

После создания объекта ParamBean, Jahia передает управление объекту OperationManager, который также использует конвейер. Конвейер действий описан далее в этом документе в соответствующем разделе. Конвейерные операции будут взаимодействовать со службами Jahia для выполнения желаемых действий, и результат будет зависеть от того какие движки были вызваны в конвейер.

В случае использования движка ядра, конвейер действий создаст все необходимые объекты содержимого, дающие разработчику шаблонов тот контент, который он будет размещать на JSP-страницах. Поэтому после формирования объектов контента портал напрявляет пользовательский запрос к JSP-шаблону для отрисовки страницы.

В заключение конечный результат отправляется обратно в браузер. Естественно, это очень упрощенный пример работы портала, но он может послужить базой для понимания схемы обработки запросов в Jahia. Взаимодействие портала с кэшами было опущено для упрощения описания процесса, но очевидно, что интеграторам придется брать их во внимание при разработке своих портальных решений.

Конвейеры


Pipelines
Operations pipeline
Site lookup
Operation mode resolving
Page authorization check
URL history valve
Theme valve
Skeleton aggregator
Portal process action
Jahia Engine
Response output
Skeleton parse & store
WEB-INF/etc/spring/applicationcontext-pipelines.xml
Authentification pipeline
Browser
NTLM filter
Jahia servlet
ParamBean
jahia/WEB-INF/etc/config/auth-pipeline.xml
jahia/WEB-INF/etc/spring/applicationcontext-services.xml Provider list config
resolveUser
Auth Pipeline
Http Basic Valve
CAS Valve
Login Engine Valve
Session Valve
Cookie « Remenber me » Valve
Server Valve
NTLM Valve
Custom Valve
lookupUser
verifyPassword
JahiaUserManagerService
Jahia Database provider
LDAP provider
Custom provider 1

Как было сказано в предыдущем разделе, Jahia использует конвейеры (pipelines) для предоставления настраиваемых цепочек обработки. Конвейер – это упорядоченый список уровней (клапанов, valves), которые по очереди вызываются друг за другом. Что является специфическим для шаблона проектирования Конвейер (pipeline design pattern) – так это то, что ответственностью каждого уровня является вызвать следующий. Другими словами, уровень может выбирать, передавать выполнение следующему, или нет.

Конвейер аутентификации вызывается объектом ParamBean для определения пользователя, основываясь на текущем контексте. Пользователи могут быть определены с использованием всех методов: это поиск информации в сохраненных сессиях, если они ранее уже были определены; или с помощью интегрированной системы единого входа типа CAS, или используя аутентификацию контейнера JEE. На рисунке выше нижний конвейер – конвейер аутентификации, в нем видны различные уровни для определения пользователей. Мы не будем углубляться в детали каждого уровня, но стоит отметить, что данный механизм разрешения является настраиваемым с помощью xml-файлов Spring, что облечает интеграторам применять их собственные уровни.

Конвейер действий (верхняя часть рисунка) используется в Jahia для предоставления подключаемых цепочек обработки запросов. Уровни конвейера могут принимать решение остановить обработку запроса в любой точке, что особенно важно при работе с кэшем. Если содержимое можно прочитать из кэша, обработка запроса приостанавливается в этом месте, и это освобождает сервер от выполнения ненужных операций, делая процесс обработки запроса очень эффективным. Уровень конвейера, ответственный за чтение из кэша на рисунке назван «Сборщик каркаса» («Skeleton aggregator»). Уровень конвейера «URL history» используется для записи всех урлов, запрошенных пользователем в целях построения навигационной цепочки (breadcrumbs).

Интеграторы могут найти интересной возможность настраивать конвейер действий для расширения функционала, задействованного при обработке пользовательских запросов. Следует отметить, что порядок размещения уровней в конвейере очень важен, и что не рекомендуется этот порядок менять без глубокого понимания принципов работы портала Jahia.

Импорт и экспорт

Подсистема импорта и экспорта портала Jahia – это мощный механизм миграции контента различными способами между веб-сайтами Jahia или даже между различными инсталляциями системы.

Для экспорта контента система использует xml-формат хранения данных JSR-170 (или JCR), вместе с другими специфическими файлами (типа файловых иерархий для экспорта двоичных данных). Все эти файлы сжимаются в zip-архив, который можно будеть использовать в подсистеме импорта.

Стандартизованный механизм импорта и экспорта делает возможным экспортировать полную инсталляцию Jahia, набор сайтов, одиночный сат или даже один раздел сайта; перемещать контент между сайтами и разделами сайтов Jahia; экспортировать и импортировать его в не-Jahia системы.

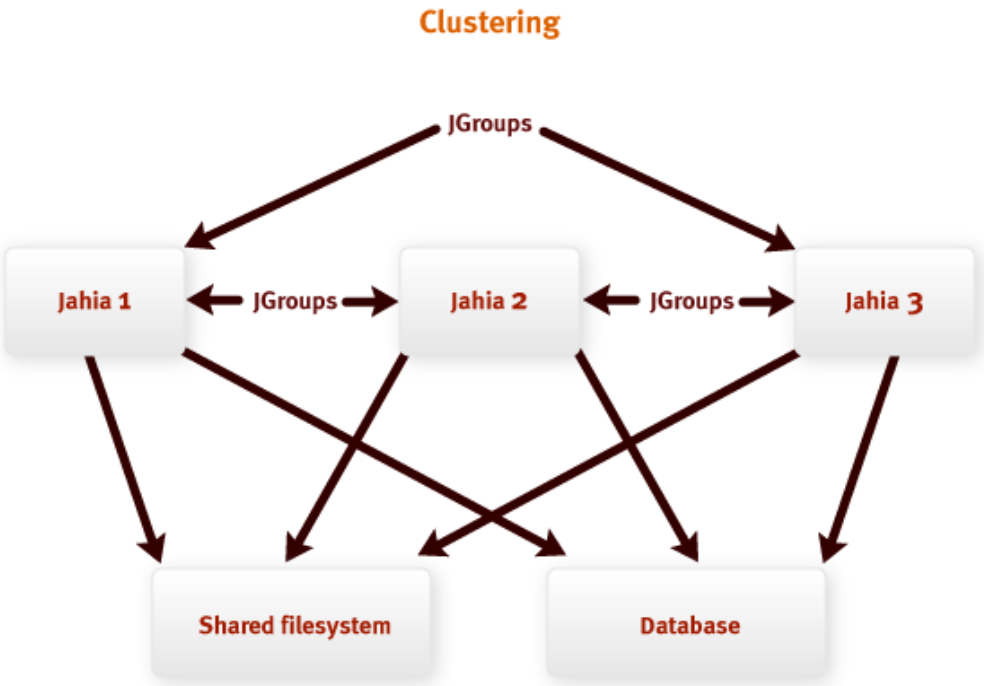
Jahia также использует технологию импорта и экспорта для возможности удаленной публикации содержимого. В данном случае обычно импорт (экспорт) содержимого вызывается планировщиком задач. Технология импорта и экспорта умеет распознавать, когда содержимое было изменено, и в таком случае просто обновляет объект содержимого вместо создания нового.

ОТНОШЕНИЕ К COPY/PASTE

Хоть это и не очевидно, но функция copy/paste тоже использует систему импорта и экспорта. Данный факт проясняет, почему содержимое можно копировать и вставлять только между совместимыми описаниями контента, так как Jahia должна знать, как преобразовывать контент между источником и приемником контента.

Связанные копии (Linked copies) также используют систему систему импорта и экспорта при отправке содержимого получателю (приемнику).

Кластеризация



Развертывание системы Jahia на кластер – это эффективный путь уменьшения нагрузки на центральный процессор и память для обеспечения работы высоконагруженых сайтов.

Типичная установка Jahia на кластер показана на рисунке выше. Узлы Jahia обмениваются напрямую друг с другом (direct messaging) и имеют доступ к разделяемым ресурсам: файловой системе и базе данных. Файловая система используется для хранения поискового индекса и для хранения двоичных данных (если сервер настроен на хранение двоичных данных в файловой системе, настройка по умолчанию – хранение в БД). В базе данных хранится все остальное. Поэтому очень важно иметь высокопроизводительную инсталляцию базы данных для обеспечения хорошей масштабируемости всей системы.

Jahia также может различать узлы по типам в кластере для обеспечения более специализированной обработки. Рассмотрим вкратце различные типы узлов.

УЗЛЫ ПРОСМОТРА

Узлы просмотра (Jahia «browsing» nodes) – это специализированные узлы Jahia, которые функционируют как узлы публикации содержимого. Также они взаимодействуют с портлетами для отрисовки страниц. Использование данного типа узла позволяет отделить нагрузку на систему выдачи контента от нагрузки на систему авторизации и систему фоновой обработки запросов.

УЗЛЫ АВТОРИЗАЦИИ

Узлы авторизации (Jahia «authoring» nodes) – это узлы кластера, использующиеся и при просмотре и при редактировании содержимого портала. Это наиболее часто используемый тип узла в кластере Jahia, поэтому нужно иметь несколько экземпляров таких узлов с тем, чтобы распределить нагрузку.

УЗЛЫ ОБРАБОТКИ

В Jahia длительно выполняющиеся задачи (это действия по валидации документов (workflow validation operations), copy/paste-операции, импорт и индексирования содержимого) выполняются как фоновые процессы. Таким образом при выполнении этих длительных операций другие узлы по-прежнему могут обрабатывать запросы по просмотру и редактированию содержимого.

СЕРВЕР ИНДЕКСИРОВАНИЯ

Индексирование содержимого и файлов может оказаться очень дорогой операцией по использованию центрального процессора и по нагрузке на оперативную память сервера. Например, для индексирования pdf-файла сервер должен выполнить скрипты, которые содержатся в файле для того, чтобы извлечь текстовое содержимое. Это требует выполнения инструкций Postscript-подобного языка, включая управление памятью и обработку инструкций, просто для того, чтобы извлечь контент. Индексирование больших файлов также требует памяти, которая не может быть освобождена до момента завершения открытия файла.

Индексирование обычно относят к операции, которая не требует выполнения в режиме реального времени, поэтому она выполняется в фоне. Для того, чтобы понизить нагрузку на сервера Jahia, выполняющие обработку текущих запросов пользователей, настоятельно рекомендуется установить отдельный сервер индексирования содержимого.

Можно даже установить в системе несколько узлов сервера индексирования для обеспечения бесперебойного выполнения данной операции.

9, route des Jeunes
CH-1227 Les acacias
Geneva, Switzerland

www.jahia.com
The Company website

www.jahia.org
The Community website