??????????

???????????? ??????????????

????? ???????????

В процессе завершения последней версии Cyclos было проведено расширенное исследование различных решений (технологий, фреймворков и платформ) для Cyclos. Основные требования и ограничения для выбора были следующими:

- Долгосрочная поддержка выбранных технологий
- Высокая масштабируемость
- Безопасность
- Простота настройки

Наш вывод заключается в том, что "Spring Framework" и "Google Web Tools (GWT)" предлагают лучшее решение для наших требований. Это текущая архитектура Cyclos:

Архитектура Cyclos 4

В приложении есть 4 концептуальных уровня:

- **Клиент**: Может быть веб-браузер, который взаимодействует с cyclos, получая JSON-данные через HTTP от веб-сервера; или клиентское приложение, взаимодействующее через REST с веб-сервисами или использующее Spring HTTP.
- **Веб**: Обрабатывает HTTP-запросы и взаимодействует с нижними уровнями, используя интерфейсы, определенные в проекте `cyclos4-api`. User façade / guest façade являются точками входа в уровень сервисов для обычных веб-клиентов. Клиенты веб-сервисов используют другой фасад.
- **Сервисы**: Точки входа (фасады) являются либо EJB, либо простыми Spring beans (в зависимости от развертывания). Сервисы разделены на 2 компонента: уровень безопасности (отвечает за проверку прав вызывающего на методы) и реализация (выполняет бизнес-правила и доступ к постоянным сущностям). Сервисы используют JPA (с помощью Querydsl), который взаимодействует с сервером базы данных.
- **База данных**: СУБД, которая хранит данные.

Также Java-клиент может использовать RMI для прямого взаимодействия с уровнем безопасности сервисов (например, CRM-программное обеспечение).

?????? MVC

Архитектурный шаблон Model-View-Controller (Модель-Представление-Контроллер) применяется с GWT в Cyclos следующим образом:

?????? (Model)

VOs, DTOs и QueryParameters - все это наблюдаемые объекты (подклассы org.cyclos.model.Bean). Они поддерживают регистрацию слушателей для изменений свойств и имеют служебные методы для чтения и записи свойств, поскольку GWT не поддерживает рефлексию напрямую.

?????????? (Controller)

Контроллер выступает посредником между Представлением и Моделью, получая данные из сервисов, обновляя компоненты Представления и отправляя данные обратно в сервисы для сохранения.
В Cyclos в роли Контроллеров выступают Страницы (подклассы org.cyclos.client.ui.Page). Они создают экземпляры компонентов представления (виджетов) и содержат ссылки на удаленные сервисы.
Методы Контроллера вызываются Представлением косвенно, например, onSearch(), onSave(). Когда Представлению требуются дополнительные данные (например, выбор категории объявления при регистрации объявления), ответственность Контроллера - получить эти данные и передать их представлению. Вероятно, в этом случае компонент представления будет иметь метод типа setAdCategories(Collection<AdCategoryVO>).

????????????? (View)

Визуальное представление данных использует компонентные страницы. В основном страницы строятся путем расширения панели макета (например: SearchFiltersPanel, FormFlowPanel, DataTable, и т.д.) и включения элементов ввода, особенно полей (например: TextField, IntegerField, и т.д.).
Этот компонент показывает значения из Модели (например, <Module>Query) и обновляет их по мере того, как пользователь заполняет поля формы. Это делается с помощью механизма привязки. Класс Field имеет метод bind(), который принимает org.cyclos.model.Bean и имя свойства. Также есть ярлык для него на FormFlowPanel, который принимает имя свойства и поле. Когда привязка выполнена, значение свойства объекта модели автоматически отражается на значении поля, и любые обновления с любой стороны отражаются на другой.

??????

Клиентское приложение будет взаимодействовать с веб-сервером, отправляя/получая данные в формате JSON по HTTP.
Веб-сервер будет обращаться к уровню сервисов (в сервере приложений) через интерфейсы. Фактическая реализация будет получена через JNDI lookup, но конфигурация будет выполняться через Spring. Таким образом, мы можем прозрачно изменять локальные интерфейсы на удаленные (разные имена JNDI), когда веб-сервер работает в той же / другой JVM, что и сервер приложений.

Уровень сервисов реализован с использованием stateless session beans, которые используют JPA EntityManager для доступа к базе данных.

??????? ????????? ?????????

Сущности (отображенные с JPA) будут ограничены областью видимости реализации сервиса. Общение между сервисом и верхними уровнями будет осуществляться с использованием DTO и VO, а не самих сущностей. Обычно DTO будут содержать все редактируемые поля сущности, а VO будут содержать только поля, необходимые для конкретных страниц. Это решит некоторые проблемы:

- Во многих случаях для выполнения сложной задачи одной сущности недостаточно, поэтому даже в Cyclos 3 для таких случаев используются DTO (например, платежи выполняются с помощью DoPaymentDTO, а не сущности Payment). Это стандартизирует данный подход.
- Мы решим проблемы с ленивой загрузкой данных. Мы гарантируем, что когда к сущностям осуществляется доступ, существует EntityManager, который прозрачно загружает ленивые связи. Когда данные отправляются на верхние уровни, ни в DTO, ни в VO не будет ленивых прокси или неинициализированных коллекций.

Кроме того, когда необходимо загружать большие объемы данных (в основном отчеты и CSV-выгрузки), это должно обрабатываться на уровне сервисов, и загружаться должно только результирующее содержимое файла (PDF или CSV), а не сами сущности.

????????????? ??????? (Module Enum)

Cyclos4 организован в модули и подмодули. Список существующих модулей определен в перечислении `org.cyclos.utils.Module` в '''cyclos4-api-core'''. Кроме того, для каждого модуля существует соответствующее перечисление подмодулей, содержащее имеющиеся в нем подмодули.

Основная причина для подмодулей - позволить отправлять переводы приложения по запросу, а не отправлять все переводы сразу. То есть основной веб-клиент может запросить только переводы, необходимые для конкретной страницы.
Cyclos4 определяет специальный класс Enum, который перечисляет все модули, существующие в приложении. Enum находится в cyclos4_api, в пакете org.cyclos.model.Module. Также существуют различные перечисления подмодулей, которые определяют подмодули в каждом модуле.
Эти перечисления используются для генерации ключей перевода через реализации MessageKey, которые определены в Enum с областью видимости 'Module'.
Например, вызов перевода будет выглядеть так:
`message(SystemKeys.Configurations.HEADING_ACTIVE)`
Enum с областью видимости 'Module' (SystemKeys) содержит множество реализаций MessageKey.

Реализация MessageKey (Configurations) будет иметь метод, ссылающийся на подмодуль.
Подмодуль будет иметь метод, ссылающийся на модуль.
Файлы переводов можно найти в cyclos4_impl/src/META-INF/translations.
Пример выше будет расположен в файле ''Translation-SYSTEM.properties'', а ключ будет ''CONFIGURATIONS.headingActive''.

CRUD ???????

CRUDService в модели - это сервис, позволяющий выполнять основные операции с базой данных: Create (создать/сбросить), Read (загрузить), Update (сохранить) и Delete (удалить). Большинство интерфейсов Cyclos для сервисов расширяют этот интерфейс CRUDService. Только очень немногие сервисы в Cyclos расширяют nl.strohalm.cyclos.services.Service напрямую.

?????

Скриншоты
Объяснение разделов и элементов макета

??????????

Мы не используем JAAS для безопасности, а наше собственное решение. Это потому, что JAAS обрабатывает только роли, а нам нужно больше. Пожалуйста, ознакомьтесь с параграфом о безопасности для полного обзора архитектуры безопасности.

????? ??????????????

Вот поток, который происходит, когда клиент вызывает метод сервиса:

1. Запрос отправляется от клиента на сервер с использованием GWT-RPC или механизма связи Spring's HTTP invoker;
2. HTTP-фильтр гарантирует, что получены некоторые основные данные о запросе, такие как сеть и канал;
3. Сервлет Spring обрабатывает запрос и определяет метод сервиса, который нужно вызвать;
4. Используется прокси к интерфейсу сервиса. Этот прокси использует `ServiceFacade` для вызова метода на сервере;
5. Вызывается соответствующий класс `ServiceSecurity`, который применяет правило безопасности;
6. Класс безопасности делегирует вызов фактической реализации сервиса через его локальный интерфейс сервиса;
7. Наконец, реализация сервиса выполняет бизнес-логику.

Вызов метода сервиса

Revision #1
Created 30 October 2025 17:58:11 by admin
Updated 30 October 2025 18:04:57 by admin