Сервисно - ориентированные архитектуры

Лекция 5

Архитектура информационной системы

Архитектурой информационной системы называется концепция, согласно которой взаимодействуют компоненты информационной системы.

Интерфе́йс (от англ. *interface*) — общая граница между двумя функциональными объектами, требования к которой определяются стандартом[1]; совокупность средств, методов и правил взаимодействия (управления, контроля и т.д.) между элементами системы[2].

Примеры:

- элементы электронного аппарата (телевизора, автомагнитолы, часов и т. п.), такие как дисплей, набор кнопок и переключателей для настройки, плюс правила управления ими, относятся к человеко-машинному интерфейсу;
- клавиатура, мышь и пр. устройства ввода — элементы интерфейса «человек—компьютер».

Элементы информационной системы

Любая информационная система (ИС) включает в себя три компонента:

- Управление данными;
- Бизнес-логику;
- Пользовательский интерфейс.

Данные хранятся в базах данных, а управление ими осуществляется с помощью системы управления базами данных (СУБД).

Бизнес-логика определяет правила, по которым обрабатываются данные. Она реализуется набором процедур, написанных на различных языках программирования.

Пользователь работает с интерфейсом, где логика работы ИС представлена в виде элементов управления – полей, кнопок, списков, таблиц и т.д.

Сервис(но)-ориентированная архитектура **SOA, service-oriented architecture)**

Под сервис-ориентированной архитектурой понимается модульный подход к проектированию прикладных информационных систем, который руководствуется следующими принципами:

- явное отделение бизнес-логики прикладной системы от логики презентации информации;
- реализация бизнес-логики прикладной системы в виде некоторого количества программных модулей (сервисов), которые доступны извне (пользователям и другим модулям), чаще всего в режиме "запрос-ответ", через четко определенные формальные интерфейсы доступа;
- при этом "потребитель услуги", который может быть прикладной системой или другим сервисом, имеет возможность вызвать сервис через интерфейсы, используя соответствующие коммуникационные механизмы.

Базовыми понятиями в такой архитектуре являются *"информационная услуга“ (сервис)* и *"композитное приложение"*.

Принципы SOA:

- *архитектура*, как таковая, не привязана к какой-то определенной технологии;
- независимость организации системы от используемой вычислительной платформы (платформ);
- независимость организации системы от применяемых языков программирования;
- использование сервисов, независимых от конкретных приложений, с единообразными интерфейсами доступа к ним;
- организация сервисов как слабосвязанных компонентов для построения систем.

Информационная услуга (сервис)

Информационная услуга (сервис) -
это *атомарная* прикладная *функция автоматизированной системы*, которая пригодна для использования при разработке приложений, реализующих прикладную логику автоматизируемых процессов как в самой системе, так и для использования в приложениях других автоматизированных систем.

Сервис обычно характеризуется следующими свойствами:

- возможность многократного применения;
- услуга может быть определена одним или несколькими технологически независимыми интерфейсами;
- выделенные услуги слабо связаны между собой, и каждая из них может быть вызвана посредством *коммуникационных протоколов*, обеспечивающих возможность взаимодействия услуг между собой.

Композитное (составное) приложение

- *Композитное (составное) приложение* - программное решение для конкретной прикладной проблемы, которое связывает прикладную логику процесса с источниками данных и информационных услуг, хранящихся на *гетерогенном* множестве базовых информационных систем.
- Обычно композитные приложения ассоциированы с процессами деятельности и могут объединять различные этапы процессов, представляя их пользователю через единый *интерфейс*.

SOA позволяет:

- создать систему корпоративных композитных приложений, основанных на системе корпоративных Web-сервисов;
- организовать *интеграцию приложений*, бизнес-процессов, с автоматизацией бизнес-процессов;
- использовать различные транспортные протоколы и стандарты форматирования сообщений, средства обеспечения безопасности, надежной и своевременной доставки сообщений;
- существенно повысить скорость разработки прикладных приложений и снизить затраты на эти цели.

Технологии

- Архитектура не привязана к какой-то определённой технологии. Она может быть реализована с использованием широкого спектра технологий, включая такие технологии как веб-сервисы REST и SOAP.

Веб-сервис

- Веб-сервис – идентифицируемая веб-адресом программная система со стандартизированными интерфейсами. Веб-службы могут взаимодействовать между собой и со сторонними приложениями посредством сообщений, основанных на определенных протоколах.
- Интерфейс сервиса описан в Web Service Description Language (WSDL)
- Веб-сервис – единица модульности при создании СОА приложения.
- Характеристики:
- функциональная совместимость;
- расширяемость;
- возможность машинной обработки описания.

Веб-сервисы взаимодействуют между собой через протокол SOAP в соответствии с WSDL.

SOAP

- SOAP (Simple Object Access Protocol – простой протокол доступа к объектам) - протокол обмена структурированными сообщениями в распределенной вычислительной среде; первоначально предназначался для удаленного вызова процедур (RPC), сейчас используется так же для обмена произвольными сообщениями в формате XML.
- SOAP может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTP, HTTPS и др.
- Как правило, применяется в бизнес-приложениях и на основе ранее существовавших систем.
- Ориентирован на написание методов и ОО-подход.

REST

- **REST** (***Representational State Transfer*** — «передача состояния представления») представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. REST является альтернативой RPC.
- В сети Интернет вызов удалённой процедуры может представлять собой обычный HTTP-запрос (обычно «GET» или «POST»; такой запрос называют *«REST-запрос»*), а необходимые данные передаются в качестве параметров запроса.
- Для веб-служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «**RESTful**».

Требования к архитектуре REST

1. **Модель клиент-сервер**
2. **Отсутствие состояния**

В период между запросами клиента никакая информация о *состоянии* клиента на сервере не хранится.

Информация о состоянии сессии может быть передана сервером какому-либо другому сервису (например, в службу базы данных) для поддержания устойчивого состояния, например, на период установления аутентификации. Клиент инициирует отправку запросов, когда он готов (возникает необходимость) перейти в новое состояние.

- Во время обработки клиентских запросов считается, что клиент находится в *переходном состоянии,* между обработками – *в состоянии покоя*. Каждое отдельное *состояние* приложения представлено связями, которые могут быть задействованы при следующем обращении клиента.

3. Кэширование

Кэширование (кэш) – некоторой промежуточный буфер, в котором хранятся данные.

Клиенты, а также промежуточные узлы, могут выполнять кэширование ответов сервера. Ответы сервера, в свою очередь, должны иметь явное или неявное обозначение как кэшируемые или.

4. Единообразие интерфейса

Унифицированные интерфейсы позволяют каждому из сервисов развиваться независимо.

Требования к унифицированным интерфейсам:

- **Идентификация ресурсов**

Ресурс – многократно используемый и относительно стабильный объект, который может распределяться внутри системы**.**

Все ресурсы идентифицируются в запросах, например, с использованием URI в интернет-системах.

- **Манипуляция ресурсами через представление**

Представление ресурса – документ, отражающий текущее или требуемое состояние ресурса.

- **«Самоописываемые» сообщения**

Каждое сообщение содержит достаточно информации, чтобы понять, каким образом его обрабатывать.

- **Гипермедиа как средство изменения состояния приложения**

Клиенты изменяют состояние системы только через действия, которые динамически определены в гипермедиа на сервере (к примеру, гиперссылки в гипертексте). Исключая простые точки входа в приложение, клиент не может предположить, что доступна какая-то операция над каким-то ресурсом, если не получил информацию об этом в предыдущих запросах к серверу.

5. Слои

Клиент обычно не способен точно определить, взаимодействует он напрямую с сервером или же с промежуточным узлом, в связи с иерархической структурой сетей (подразумевая, что такая структура образует слои).

6. Код по требованию (необязательное ограничение)

REST может позволить расширить функциональность клиента за счёт загрузки кода с сервера в виде апплетов или сценариев.

Апплет - это несамостоятельный компонент программного обеспечения, работающий в контексте другого, полновесного приложения, предназначенный для одной узкой задачи и не имеющий ценности в отрыве от базового приложения.

Сценарий — это программа, имеющая дело с готовыми программными компонентами.

Учет интересов клиентов и выбор оптимального решения

Отличия REST от SOAP:

- REST поддерживает различные форматы: text, JSON, XML; SOAP — только XML,
- REST работает только по HTTP(S), а SOAP может работать с различными протоколами,
- REST может работать с ресурсами. Каждый URL это представление какого-либо ресурса. SOAP работает с операциями, которые реализуют какую-либо бизнес логику с помощью нескольких интерфейсов,
- SOAP на основе чтения не может быть помещена в кэш, а REST в этом случае может быть закэширован,
- SOAP поддерживает SSL и WS-security, в то время как REST — только SSL.

SOAP и REST

Критерий	SOAP	RESTful
Спецификация Java	JAX-WS	JAX-RS
Назначение	Инкапсулирует бизнес-логику	Доступ к ресурсам/данным
Методология разработки	Объектно-ориентированная (сервисно-)	Ресурсно-ориентированная
Независимость от языка	Да	Да
Независимость от платформы	Да	Да
Независимость от транспорта	Да	Нет, только HTTP
Стандартизирован	Да	Нет
Безопасность	SSL, WSL-Security	SSL
Транзакции	WS-AtomicTransaction	Нет

Критерий	SOAP	RESTful
Производительность	Ниже	Выше
Кэширование	Нет	Есть для метода GET
Транспорт	HTTP, SMTP, JMS	HTTP
Размер сообщений	Большой, служебные данные	Небольшой
Протокол сообщений	XML	XML, JSON, любой тип MIME
Описание сервисов	WSDL	Формального нет
Инструменты разработки	Требуются	Можно без них

Выводы: REST против SOAP можно перефразировать как «Простота против Стандарта». В случае REST (простота) у вас будет скорость, расширяемость и поддержка многих форматов. В случае с SOAP у вас будет больше возможностей по безопасности (WS-security) и транзакционная безопасность (ACID).