

# JRRA. Демонстрация корпоративных закупок как пример прикладной системы

Антон Чечель

*Технологии P2P обновят Internet так же радикально, как первый браузер. Модель обмена ресурсами способна революционно изменить мир вычислительных систем*

**Патрик Гелсингер**, вице-президент и генеральный директор Intel Desktop Products Group. ([Computerworld, #32/2000](#))

Как уже, возможно, догадался проницательный читатель, в заглавие статьи вынесено название протокола, над которым работает наш коллектив, и сегодня речь пойдёт о простоте и элегантности разработки прикладных систем поверх оного. Итак, приступим.

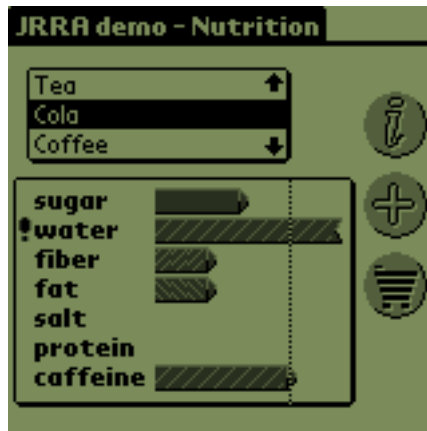


Для начала определимся с понятиями. **Peer-to-peer** — это не просто системы обмена файлами или мгновенными сообщениями. Я по личному опыту убедился в таком сложившемся стереотипном понимании данного понятия у большинства

## JRRA. Демонстрация корпоративных закупок как пример прикладной системы

пользователей и программистов, не имеющих пока непосредственного опыта работы с ними. Это - прежде всего *модель взаимодействия хостов в сети*. Ещё одна модель после традиционного client/sever. Описание достоинств и недостатков её по сравнению с остальными выходит за рамки этого материала. Хочу только отметить, что аналитики в области индустрии информационных технологий прогнозируют стремительное развитие P2P.

Итак, Вы программист, менеджер проекта или просто специалист, заинтересовавшийся разработкой систем под наш протокол. На днях выходит первый релиз демонстрационной программы, которая доступна [здесь](#).



### 1. Пример корпоративных закупок

Допустим несколько пользователей системы производят закупки продуктов питания в супермаркете. У каждого имеется некоторое wireless устройство оснащённое данной программой. В программу либо статически внесён, либо загружен с какого-нибудь уже проинициализированного т.н. *привилегированного* реер'а список продуктов питания с составляющими ингредиентами, а также лимиты (количественные ограничения) на каждый ингредиент (сахар, соль, жир и т.п). Имеется общая виртуальная корзина, для которой и ведётся подсчёт суммы ингредиентов по закупкам и отслеживание превышения лимитов. Покупатели физически могут находиться в разных местах, расстояния определяются только техническими характеристиками их мобильных телефонов или PDA устройств.

Выигрыш от P2P взаимодействия в таких системах получается от децентрализации.

## *JRRA. Демонстрация корпоративных закупок как пример прикладной системы*

Наличие сервера вовсе не является обязательным, пользователи системы равны по определению P2P. Список всех реер'ов каждый может получить, например, от одного *привилегированного* при описанной выше инициализации. Протокол JRRA позволяет скрыть от прикладного программиста все детали низкоуровневого взаимодействия и предоставляет свой API.



Product	Amount	Owner
Pear:	5	me
Cheese:	2	me
Tea:	4	me
Milk:	2	me
Fish:	1	me
Apple:	1	me

At the bottom of the window are three buttons: OK, Delete, and Clear.

В качестве платформы на wireless устройства была выбрана IBMJ9 Virtual Machine с их имплементацией Connected Limited Device Configuration на PDA с PalmOS. Также в демонстрации использованы некоторые особенности Mobile Information Device Profile, в частности Record Store Management System для хранения данных на самих устройствах.

Для взаимодействия по JRRA программисту необходимо создать свой класс, имплементирующий `org.jrra.controller.spi.Listener` для организации прослушивания определённого порта.

```
public static class NutrListener implements Listener {
public void update(EventMessage eventMessage) {
    instanceof                                CMDSendBasket) {
        BasketDescription bd = ((CMDSendBasket)
                                eventMessage.getPacketData()).getBD();
        receiveBasket(bd, eventMessage.getPeer().getUIN());
    }
    instanceof                                CMDClearBasket) {
        receiveClearBasket(eventMessage.getPeer().getUIN());
    }
}
```

## *JRRA. Демонстрация корпоративных закупок как пример прикладной системы*

```
instanceof CMDRemoveFromBasket) {
    BasketDescription bd = ((CMDRemoveFromBasket)
        eventMessage.getPacketData()).getBD();
    receiveRemoveFromBasket(bd,
        eventMessage.getPeer().getUIN());
}
}
}
```

Он отвечает за приём событий об изменениях состояния корзины, добавлениях новых peer'ов в систему и т.д. Это и будет аналог серверной части в клиент- сервере. В P2P каждый peer является как клиентом, так и сервером одновременно.

В JRRA, как видно из вышеприведённого кода, программисту дана возможность определять свои собственные прикладные пакеты, если набор стандартных не устраивает запросы. Таким образом достигается большая гибкость в разработке и сопровождении разрабатываемой прикладной системы. Вот пример простого пакета, отвечающего за изменение состояния корзины при добавлении того или иного продукта в неё.

```
public class CMDSendBasket implements PacketData {
    static {
        PacketDataFactory.addFactory(
            CMDSendBasket.class.getName(),
            new CMDSendBasket.Factory());
    }
    static class Factory extends PacketDataFactory {
        protected PacketData create() {
            return new CMDSendBasket();
        }
    }
    private BasketDescription bd =
        new BasketDescription();
    public CMDSendBasket() { }
    public CMDSendBasket(BasketDescription bd) {
        this.bd = bd;
    }
    public BasketDescription getBD() {
        return bd;
    }
}
```

## JRRA. Демонстрация корпоративных закупок как пример прикладной системы

```
}  
public InputStream read() throws IOException {  
    ByteArrayOutputStream arrayOut =  
        new ByteArrayOutputStream();  
    DataOutputStream dataOut =  
        new DataOutputStream(arrayOut);  
    try {  
        bd.write(dataOut);  
    } catch (IONutritionException ione) {  
        throw new IOException("Can't read: " + ione);  
    }  
    return new ByteArrayInputStream(  
        arrayOut.toByteArray());  
}  
public void write(InputStream in) throws IOException {  
    DataInputStream dataIn =  
        new DataInputStream(in);  
    try {  
        bd.read(dataIn);  
    } catch (IONutritionException ione) {  
        throw new IOException("Can't write: " + ione);  
    }  
}  
}
```



В настоящее время завершаются разработки по переходу всех наших прикладных пакетов в XML формат. Это позволит качественно улучшить такие характеристики протокола как универсальность, гибкость, переносимость и упрощения процесса

## *JRRA. Демонстрация корпоративных закупок как пример прикладной системы*

разработки систем под него. В таком формате любой прикладной пакет будет иметь структуру, которая описывает сама себя, то есть в нём будет присутствовать некоторая метаянформация.

Позволю себе немного отойти от темы и углубиться в дебри программирования под PalmOS. Стоит заметить, что специалисты IBM проделали хорошую работу, результатом которой явилась J9 VM под эту ОС. Вот так выглядит код, обрабатывающий события обновления корзины.

```
public void receiveBasket(BasketDescription bd,
                        int uin) {
    OS.FrmCustomAlert(Rsc.altBasketReceived,
                    new CharPtr("" + uin),
                    new CharPtr(),
                    new CharPtr());

    try {
        ProductFactory pFact = (ProductFactory)
            FactoryManager.lookup("ProductFactory");
        Product p = (Product)
            pFact.getInstanceByID(bd.getProductID());
        basket.addProduct(p, uin, bd.getValue());
        OS.FrmUpdateForm(Rsc.frmMain,
                        Rsc.eventRedraw);
    } catch(NutritionException ne) {
        OS.FrmCustomAlert(Rsc.altError,
                        new CharPtr(ne.getMessage()),
                        new CharPtr(), new CharPtr());
    }
}
```

В этой деманстрационной прикладной системе введен термин *персистентного* (persistent - постоянный) объекта как унифицирующего понятия для контроля над этими объектами. Все объекты системы имплементируют один интерфейс (com.offshorecode.nutrition.PersistentObject), что позволяет помимо ряда преимуществ перед обычными объектами ещё и абстрагироваться от способа их хранения (в нашем случае – это RMSMIDP). В дальнейшем будет осуществлён переход под перспективный формирующийся стандарт Java Data Objects для представления всех прикладных объектов.

## JRRA. Демонстрация корпоративных закупок как пример прикладной системы

После инициализации listener'a происходит регистрация своего peer'a во всей системе. Следующий фрагмент кода показывает, что необходимо для этого.

```
Address address = new Address();
// next.offshorecode.comaddress.getProperties().
setProperty("inetAddress",
            "193.41.161.133");
address.getProperties().setProperty("port", 9000);
Peer privilegedPeer = new Peer(PRIVELEGED_PEER);
privilegedPeer.setAddress("org.jrra.network.spi.UDP",
                        address);
// Getting peer from privileged onePeer myPeer = initMyPeer();
myPeerList = new PeerList();
// Adding peer to peerlistmyPeerList.bind(privilegedPeer);
controller = new Controller(myPeer);
NutrListener myListener = new NutrListener();
// Adding listener to controllercontroller.addListener(myListener);
// Starting listenercontroller.initReceiver();
```



Здесь инициализационная информация о своём peer'e получается по сети от привилегированного, который физически расположен на хосте next.offshorecode.com. В этой роли может выступать любой хост в сети с соответствующим JRRA сервисом, т.н. *searching and indexing service*. Так как протокол не зависит ни от какой конкретной аппаратной платформы, ни от какого конкретного стека сетевых протоколов, то им может быть любое устройство, поддерживающее Java (точнее хотя бы CLDC конфигурацию). Для демонстрации был выбран обычный PC с OS Linux в качестве

## *JRRA. Демонстрация корпоративных закупок как пример прикладной системы*

упомянутого searching and indexing сервиса.

В случае изменения корзины, инициатором которого является сам пользователь, должно отправляться сообщение остальным peer'ам (тут нами введено понятие peerlist'a) системы. Для этого достаточно передать соответствующему методу controller'a экземпляр класса прикладного пакета.

```
private void sendBasket(BasketDescription bd) {  
    CMDSendBasket cmd = new CMDSendBasket(bd);  
    controller.sendMessage(cmd, myPeerList);  
}
```

JRRA берёт на себя решение проблемы синхронизации и, соответственно, целостности данных в системе. Суть проблемы заключается в том, что в децентрализованных системах данные распределены физически и нарушение целостности неизбежно. Для синхронизации мы остановились на стратегии увеличения нагрузки на процессоры устройств и уменьшения трафика итак ненадёжных сетей wireless устройств. Сообщения передаются пакетами с т.н. индексами (уникальными в системе идентификаторами) объектов, в частности продуктов при добавлении или удалении их из корзины. В рамках синхронизации событий уже одного устройства проблема решена на уровне PalmOS. Прелесть нашей реализации в том, что обработчик событий системы работает в одном потоке (thread). И таким образом организованная очередь событий является реализацией проверенного временем надёжного механизма синхронизации.

Прикладному же программисту можно и не знать деталей реализации, хотя всё же желательно. Эту идею прекрасно выразил PeterWayner словами «Хорошие программы можно создавать и не очень разбираясь в тонкостях системы программирования, но отличные программы можно разрабатывать только при глубоком понимании механизмов работы языка» в книге «Java & JavaScript Programming». Здесь речь идёт несколько не о платформе или протоколе, а о языке, но, спроецировав эту мысль на API нашего протокола, имеем отличный принцип и стимул для его изучения. Но это уже вопрос методологии, что есть личное дело каждого профессионала или корпорации.

На этом я закончу свой краткий очерк. Это всё, что достаточно знать для начала разработки под JRRA. Иногда забавно отследить смысл слов «необходимо» и

## *JRRA. Демонстрация корпоративных закупок как пример прикладной системы*

«достаточно» с не математической точки зрения...

Скачайте демонстрацию, ознакомьтесь с нашим постоянно обновляющимся сайтом, ознакомьтесь с API и Вы будете уже в состоянии разрабатывать любую P2P систему. Наш протокол, а в будущем и целая платформа может стать основой для мобильных (и не только) программных решений бизнеса Вашей компании. А если Вы изъявите желание вступить в ряды opensource проекта JRRA, то получите прекрасную возможность реализовать свой талант, знания и опыт Java разработчика. Выбор за Вами!