Степанов Е.О., к.ф.-м.н.
Ярцев Б.М.

Учебно-методическое пособие по дисциплине «Архитектуры и технологии разработки распределенного программного обеспечения»

Описание самостоятельной работы студентов (СРС)

1 **Введение**

В данном пособии содержатся задания для самостоятельной работы, которые помогут вам глубже разобраться в изучаемых технологиях и получить опыт их применения в проектах, приближенных к реальности.

По каждому из заданий, которые вы будете делать, необходимо представить откомпилированное работающее приложение, и быть готовым продемонстрировать его функциональность. Так же нужно предоставить на проверку исходный текст программы, снабженный поясняющими комментариями.

Очень важным аспектом создания коммерческих программных продуктов является написание качественной документации, как пользовательской, так и предназначенной для разработчиков, которые будут поддерживать данный продукт. Это позволит избавиться от большого количества возможных проблем и сократить стоимость дальнейшей разработки продукта.

Пользовательская документация должна содержать инструкцию по установке и использованию программы, основные характеристики и требования к окружению. Так же требуется изложить основные возможные отказы и инструкции по устранению их причин.

Документация для разработчика должна содержать функциональную спецификацию, описание архитектуры, входов и выходов программы. Нужно подробно описать интерфейсы и реализации классов и используемые структуры данных. Требуется привести схемы взаимодействия компонентов системы, события, сигналы и реакции на них. Провести тесты производительности и описать их результаты.

2 Технология CORBA

2.1 *Разработка системы бронирования билетов*

Разработать распределенную систему, обеспечивающую бронирование билетов на самолеты различных авиакомпаний. Предполагается, что на удаленном сервере находится база данных с информацией об авиарейсах (номер, количество свободных мест, стоимость билетов), которая может изменяться со временем.

Задачами системы являются:

1) поддержка возможностей обновления и коррекции информации в БД (внесение новых рейсов, изменение цен, и.т.п.). Для этого необходимо разработать специальное клиентское приложение.
2) осуществление постоянного мониторинга состояния базы клиентами. После загрузки клиент должен отображать состояние базы в своем окне и обновлять его по мере необходимости (при изменении количества свободных билетов, вследствие бронирования, произведенного другими клиентами или при модификации базы).

Клиентские приложения должны быть реализованы в виде Java-апплетов. Изначально предполагается, что апплеты считываются с другого сервера, нежели тот, на которой поддерживается БД (таким образом, в соответствии с правилами безопасности Java, клиент не может устанавливать сетевые соединения с хостом БД).

2.2 *Разработка системы заказов для книжного склада*

Разработать систему заказов для книжного склада, на котором хранятся книги различных издательств. Предполагается, что на удаленном сервере находится база данных с информацией о книгах (Название, автор, издательство, стоимость и т. д.), которая может изменяться со временем.

Задачами системы являются:

1) поддержка возможностей обновления и коррекции информации в БД (внесение новых книг, изменение цен, и.т.п.). Для этого необходимо разработать специальное клиентское приложение.
2) выполнение заказов клиентов и своевременное их информирование о произошедших изменениях. После загрузки клиент должен отобразить состояние базы в своем окне и обновлять его перед выполнением каждого запроса пользователя, а также по требованию.

Клиентские приложения должны быть реализованы в виде Java-апплетов. Изначально предполагается, что апплеты считываются с другого сервера, нежели тот, на которой поддерживается БД (таким

образом, в соответствии с правилами безопасности Java, клиент не может устанавливать сетевые соединения с хостом БД).

2.3 *Разработка простейшей банковской системы*

Разработать систему, обеспечивающую хранение информации о счетах клиентов, выдачу сведений по запросу авторизованных клиентов и модификацию счетов (снятие денег со счета, перевод со счета на счет, начисление на счет) авторизованными клиентами. Предполагается, что данные хранятся на удаленном сервере. Система должна выдавать клиентам отчеты о транзакциях, совершенных в течение заданного периода времени.

Задачами системы являются:

1) выполнение заказов клиентов (открытие и закрытие счетов, перевод денег, и.т.п.) и своевременное их информирование о произошедших изменениях. После загрузки клиент должен отобразить состояние счета в своем окне и обновлять его при любых изменениях. Также система должна хранить сведения о произведенных транзакциях и выдавать их по запросу владельца счета.
2) обеспечение безопасности работы. По условиям задания перевод денег на счет может выполняться любым клиентом, снятие денег, перевод со счета – только его владельцем. Мониторинг осуществленных транзакций также может осуществляться только владельцем счета.

Клиентские приложения должны быть реализованы в виде Java-апплетов. Изначально предполагается, что апплеты считываются с другого сервера, нежели тот, на которой поддерживается БД (таким образом, в соответствии с правилами безопасности Java, клиент не может самостоятельно устанавливать сетевые соединения с хостом БД).

2.4 *Общие требования и рекомендации*

Предлагается следующая схема взаимодействия компонент такой распределенной системы, использующая технологию CORBA:

- *Апплет Monitor*. Задачами этого приложения будут реализация GUI, выполнение пользовательских запросов по бронированию и отображение текущего состояния базы. Апплет будет одновременно CORBA – клиентом, осуществляя вызовы методов Application Server (другого приложения, служащего прослойкой между клиентом и БД - см. далее) и CORBA-сервером, чьи методы будут вызываться Application Server в случае изменений в БД.
- *Application Server*. Это приложение, осуществляющее непосредственное взаимодействие с БД. В его задачи входит:
  - установка соединения с базой данных

- выполнение запросов апплетов Monitor и Editor по модификации БД (эти методы Application Server и должны удаленно вызываться клиентами)

Одновременно Application Server должен реализовывать механизм обратной связи (так как клиенты типа Monitor должны вовремя отображать изменения в БД): в этом случае Application Server будет удаленно вызывать методы всех Monitor-апплетов, выступая в роли клиента[1]. Это приложение дожно быть написано на C++.

- *Апплет Editor.* Его функциями будет внесение изменений в БД по запросу пользователя. Для внесения изменений необходимо вызвать соответствующие удаленные методы Application Server[2].
- *База данных Oracle 8i или 9i.* Необходимо реализовать следующую схему взаимодействия БД с Application Server: триггера на основных таблицах БД информируют Application Server об изменениях в БД. Для этого можно использовать возможности Oracle JServer. Можно также воспользоваться стандартным пакетом AQ.

I. **Средство разработки**: любая версия JDK, начиная с 1.1 и любая C++ -среда. Возможно также использование Visual Cafe, Visual J++, Visual Age или JBuilder. Необходимая документация и различные версии JDK присутствуют на сервере фирмы Sun (http://java.sun.com). Пользовательский интерфейс апплетов можно разработать с помощью JFC/Swing.

II. **Спецификации CORBA** и соответствующее ПО (ORB) есть на сервере OMG group (www.omg.org). Читающим только на русском языке стоит заглянуть на http://www.corba.ru. Также информацию по CORBA можно найти на следующих серверах:

Teach Yourself CORBA In 14 Days: http://docs.cn.ua/Lang/Java/TeachCORBA7Days

CORBA Tutorial Material : http://www.cs.wustl.edu/~schmidt/tutorials-corba.html

CORBA for Beginners : http://www.omg.org/corba/beginners.html

Getting Started with Distributed Objects and CORBA : http://www.thomtech.com/~ootm/content/papers/gettingStartedCorba.htm

Cetus Links: 14,848 Links on Objects and Components / CORBA ORBs: http://www.cetus-links.org/oo_object_request_brokers.html

---

[1] Существуют и другие способы организации Callback (обратной связи). В данном случае необходимо использовать именно описанную выше схему. Таким образом, при осуществлении бронирования одним из Monitor, Application Server должен вызвать соответствующие методы всех Monitor-клиентов для отображения изменений. То же верно и для Editor.

[2] Как уже отмечалось, по правилам безопасности Java, Editor не может самостоятельно устанавливать сетевые соединения с БД

Internet, Java & CORBA Resource Guide: http://www.componentware.com/wp/ijcprimer.htm#Head:Internet, CORBA and Java Glossary

CORBA Applications http://cic.cstb.fr/ilc/people/va/corba2_v/sld019.htm

Introduction to CORBA(LCP) : http://www.object-ideas.com/corba/index.htm

Distributed Computing Systems Lab http://www.dcs.uky.edu/

An Object Transaction Service based on the CORBA: http://www.cs.nccu.edu.tw/~s8301/corba1.html

CORBA Object Transaction Service: http://www.peerlogic.com/products/dais/otsexp.html

Overview of Transaction Processing : http://www.inprise.com/techpubs/books/its/its/programmer/itspg01.htm

CORBA Object Trader Service: http://amazon.postech.ac.kr/trader.html

CORBA Trader Service : http://www-mtl.mit.edu/~mvermins/corba/trader/index.html

CORBA Software http://www.postech.ac.kr/~dry/CORBA/software.html

III. **Интерфейс Application Server** удобно описать на языке IDL, а потом воспользоваться транслятором с IDL на нужные языки (например, MIDL, idl2java). Такие трансляторы обычно входят в комплект поставки ORB. Рекомендуется (хотя это и необязательно) использовать в качестве ORB продукты Visibroker для Java, Visibroker для C++ (http://www.inprise.com). Кстати, и в качестве C++-среды можно использовать C++ Builder.

Информацию об Oracle (в т.ч. и документацию по Oracle Jserver) можно найти на http://www.oracle.com. В качестве клиента для выполнения SQL-запросов, создание таблиц, триггеров и процедур удобно пользоваться не стандартным средством SQL*Plus, а, например, продуктами Quest Software: SQL Navigator, TOAD (http://www.quest.com).

В отчет по заданию должны входить:

- откомпилированный работающий код;
- исходные тексты программ с комментариями;
- пользовательская документация - руководство пользователя, включающее назначение системы, основные характеристики, требование к окружению, инструкция по использованию, инструкция по установке, возможные отказы, их причины и способы устранения;
- документация для разработчика - руководство разработчика, включающее назначение системы, функциональную спецификацию, подробное описание архитектуры, описание входов, выходов и интерфейсов, описание структуры данных, описание особенностей реализации, в т.ч. схемы взаимодействия классов и их описание, основные события и реакции на них;

требования к окружению и их объяснение, характеристики и результаты тестов производительности, возможные отказы, их причины и способы устранения, история изменений.

3 Технология Enterprise Java Beans

3.1 *Разработка системы бронирования билетов*

Разработать распределенную web-систему, обеспечивающую бронирование билетов на самолеты различных авиакомпаний. Предполагается, что на удаленном сервере находится база данных с информацией об авиарейсах (номер, пункт вылета и посадки, количество свободных мест, стоимость билетов), которая может изменяться со временем. С помощью web-броузера клиент указывает начальный и конечный пункт назначения и другие интересующие параметры билета (например дату вылета или максимальное количество пересадок). Система в ответ на запрос пользователя определяет все возможные маршруты, удовлетворяющие его запросу и предлагает их клиенту. После этого клиент может либо забронировать какой-либо из предложенных рейсов либо изменить свой запрос.

Конкретная реализация базы - на усмотрение разработчика.

Задачами системы являются:

1) поддержка возможностей обновления и коррекции информации в БД (внесение новых рейсов, изменение цен, и.т.п.). Для этого необходимо разработать специальное клиентское приложение.
2) осуществление постоянного мониторинга состояния базы клиентами. После загрузки клиент должен отображать состояние базы в своем окне и обновлять его по мере необходимости.

Клиентские приложения должны быть реализованы в виде Java-servlet’ов. При этом запрещается непосредственное подсоединение сервлета к БД. Для этого должен использоваться посредник – EJB.

3.2 *Разработка системы заказов для книжного склада*

Разработать систему заказов для книжного склада, на котором хранятся книги различных издательств. Предполагается, что на удаленном сервере находится база данных с информацией о книгах (Название, автор, издательство, стоимость и т. д.), которая может изменяться со временем.

Задачами системы являются:

1) поддержка возможностей обновления и коррекции информации в БД (внесение новых книг, изменение цен, и.т.п.). Для этого необходимо разработать специальное клиентское приложение.
2) выполнение заказов клиентов. Клиенту предлагается отобрать интересующие его издания в специальную “корзину”, после чего он может оформить заказ на эти издания. После загрузки клиент должен

отобразить состояние базы в своем окне и обновлять его по таймеру или по требованию.

Клиентские приложения должны быть реализованы в виде Java-апплетов. Изначально предполагается, что апплеты считываются с другого сервера, нежели тот, на которой поддерживается БД (таким образом, в соответствии с правилами безопасности Java, клиент не может устанавливать сетевые соединения с хостом БД). Для этого должен использоваться посредник – EJB.

3.3 *Разработка простейшей банковской системы*

Разработать систему заказов для книжного склада, на котором хранятся книги различных издательств. Предполагается, что на удаленном сервере находится база данных с информацией о книгах (Название, автор, издательство, стоимость и т. д.), которая может изменяться со временем.

Задачами системы являются:

1) поддержка возможностей обновления и коррекции информации в БД (внесение новых книг, изменение цен, и.т.п.). Для этого необходимо разработать специальное клиентское приложение.
2) выполнение заказов клиентов. Клиенту предлагается отобрать интересующие его издания в специальную “корзину”, после чего он может оформить заказ на отобранные издания.

Клиентские приложения должны быть реализованы в виде Java-сервлетов. При этом запрещается непосредственное подсоединение сервлета к БД. Для этого должен использоваться посредник – EJB.

3.4 *Общие требования и рекомендации*

Предлагается следующая схема взаимодействия компонент такой распределенной системы, использующая технологию EJB (Enterprise Java Bean):

- *Servlet Monitor*. Задачами этого приложения будут реализация пользовательского интерфейса (HTML, JSP), выполнение пользовательских запросов по бронированию и отображение текущего состояния базы. Сервлет будет одновременно EJB – клиентом, осуществляя вызовы методов AppServer (другой компоненты, служащей прослойкой между клиентом и БД - см. далее).
- *Servlet Editor*. Его функциями будет внесение изменений в БД по запросу пользователя. Для внесения изменений необходимо вызвать соответствующие удаленные методы AppServer.

- *AppServer*. Это EJB компонента, осуществляющая непосредственное взаимодействие с БД. В его задачи входит:
- установка соединения с базой данных
- выполнение запросов Monitor и Editor по модификации БД (эти методы AppServer и должны удаленно вызываться клиентами)
- AppServer должен быть реализован в виде stateless session bean
- в случае, если маршрут состоит из нескольких билетов (пересадок), необходимо обеспечить целостность маршрута. Если билет на какую-то часть маршрута невозможно забронировать, это должно приводить к отказу в бронировании всего маршрута. Данную функциональность необходимо реализовать, используя встроенный механизм транзакций в EJB.
- *База данных Oracle 8i или 9i*.

I. **Средство разработки**: любая версия JDK, начиная с 1.1 и любая C++ -среда. Возможно также использование Visual Cafe, Visual J++, Visual Age или JBuilder. Необходимая документация и различные версии JDK присутствуют на сервере фирмы Sun (http://java.sun.com). Пользовательский интерфейс апплетов можно разработать с помощью JFC/Swing.

II. **Информацию об Oracle** (в т.ч. и документацию по Oracle Jserver) можно найти на http://www.oracle.com. В качестве клиента для выполнения SQL-запросов, создание таблиц, триггеров и процедур удобно пользоваться не стандартным средством SQL*Plus, а, например, продуктами Quest Software: SQL Navigator, TOAD (http://www.quest.com).

III. **Информацию о технологии EJB** можно найти на следующих серверах:

http://java.sun.com/products/ejb - Enterprise JavaBeans(TM) Technology

http://www.theserverside.com - TheServerSide.com J2EE Community

http://www.onjava.com - ONJava.com: The independent Source for Enterprise Java

http://www.javable.com/columns/serv_side/workshop/10/ch1/ Методология EJB

IV. **Информацию о Java Servlet и Java Server Pages (JSP)** можно найти на следующих серверах:

http://java.sun.com/products/jsp - JavaServer Pages(TM) Technology

http://java.sun.com/products/servlet - Java(TM) Servlet Technology

Демонстрировать работающее задание необходимо с использованием следующего программного обеспечения:

web-сервер Apache (http://httpd.apache.org/)
servlet контейнер Tomcat (http://jakarta.apache.org/)
EJB-сервер Orion Application Server (http://www.orionserver.com)

В отчет по заданию должны входить:

- откомпилированный работающий код;
- исходные тексты программ с комментариями;
- пользовательская документация - руководство пользователя, включающее назначение системы, основные характеристики, требование к окружению, инструкция по использованию, инструкция по установке, возможные отказы, их причины и способы устранения;
- документация для разработчика - руководство разработчика, включающее назначение системы, функциональную спецификацию, подробное описание архитектуры, описание входов, выходов и интерфейсов, описание структуры данных, описание особенностей реализации, в т.ч. схемы взаимодействия классов и их описание, основные события и реакции на них; требования к окружению и их объяснение, характеристики и результаты тестов производительности, возможные отказы, их причины и способы устранения, история изменений.

4 Темы для самостоятельного изучения

4.1 *Использование технологии RMI*

Разработать систему, обеспечивающую хранение информации о счетах клиентов, выдачу сведений по запросу авторизованных клиентов и модификацию счетов (снятие денег со счета, перевод со счета на счет, начисление на счет) авторизованными клиентами. Предполагается, что данные хранятся на удаленном сервере (конкретная реализация базы и выбор СУБД - на усмотрение разработчика). Система должна выдавать клиентам отчеты о транзакциях, совершенных в течение заданного периода времени. Сервер также должен брать комиссионные за обслуживание (перевод на спецсчет) – либо за количество транзакций, либо за время обслуживания. Необходимо также реализовать специальное приложение – “администратор счетов”, имеющее возможность изменять права доступа пользователей.

Задачами системы являются:

3) поддержка возможностей обновления и коррекции информации о правах доступа “администратором счетов”.
4) выполнение заказов клиентов (открытие и закрытие счетов, перевод денег, и.т.п.) и своевременное их информирование о произошедших изменениях. После загрузки клиент должен отобразить состояние счета в своем окне и обновлять его при любых изменениях. Также система должна хранить сведения о произведенных транзакциях и выдавать их по запросу владельца счета.
5) обеспечение безопасности работы. По условиям задания перевод денег на счет может выполняться любым клиентом, снятие денег, перевод со счета – только его владельцем. Администратор счетов должен иметь возможность заблокировать любой счет. Мониторинг осуществленных транзакций может осуществляться только владельцем счета и администратором.
6) контроль за количеством транзакций и уменьшение переводимых сумм на размер комиссионных.

Клиентские приложения должны быть реализованы в виде Java-апплетов. Изначально предполагается, что апплеты считываются с другого сервера, нежели тот, на которой поддерживается БД (таким образом, в соответствии с правилами безопасности Java, клиент не может самостоятельно устанавливать сетевые соединения с хостом БД).

Предлагается следующая схема взаимодействия компонент такой распределенной системы, использующая технологии Jini и RMI (Remote Method Invocation):

- *Апплет Client.* Задачами этого приложения будут реализация GUI,

выполнение пользовательских запросов и отображение текущего состояния счета. Апплет является RMI – клиентом, вызывающим удаленные методы Application Server (другого RMI-приложения, служащего прослойкой между клиентом и БД - см. далее). Для осуществления обратной связи необходимо использовать jini.event: при изменениях в БД Application Server формирует соответствующее событие; в апплетах Client имеется специальный объект (Listener), обеспечивающий его регистрацию[3]. Если событие происходит, Client обновляет свое окно.

- *Application Server.* Это RMI-сервер осуществляющий непосредственное взаимодействие с БД. В его задачи входит:
  - установка соединения с базой данных
  - выполнение запросов апплетов Monitor и Administrator, с использованием jini.transaction[4]. Эти методы Application Server и должны удаленно вызываться клиентами
  - реализация механизма обратной связи: в случае перевода денег на счет клиента, Application Server формирует специальное событие, на которое реагируют апплеты типа Client.
  - контроль за правами доступа – для этого необходимо использовать jini.lease.
  - начисление комиссионных при транзакциях
- Апплет Administrator. Его функциями будет изменение прав доступа пользователей.
- *База данных Oracle 8i или 9i.*

V. **Средством разработки** может быть любая версия JDK, начиная с 1.1. Возможно использование визуальных сред типа Visual Cafe, J++ или Java Builder. Необходимая документация и пакеты присутствуют на сервере фирмы Sun (http://java.sun.com). Пользовательский интерфейс можно разработать с помощью JFC/Swing.

VI. **Информацию об Oracle** (в т.ч. и документацию по Oracle Jserver) можно найти на http://www.oracle.com. В качестве клиента для выполнения SQL-запросов, создание таблиц, триггеров и процедур удобно пользоваться не стандартным средством SQL*Plus, а, например, продуктами Quest Software: SQL Navigator, TOAD (http://www.quest.com).

---

[3] Существуют и другие способы организации Callback (обратной связи). В данном случае необходимо использовать именно описанную выше схему.

[4] Очевидно, что все транзакции со счетами должны обладать свойствами ACID (Atomicity, Consistency, Isolation, Durability – неделимость, согласованность, изолированность и устойчивость). Невозможно частичное выполнение запроса по модификации счета – в случае сбоя необходим откат к предыдущему состоянию. Для соответствия разработки этим требованиям и предлагается использовать jini.transaction.

Демонстрацию системы можно проводить либо на институтских серверах, либо на локальной машине.

В отчет по заданию должны входить:

- откомпилированный работающий код;
- исходные тексты программ с комментариями;
- пользовательская документация - руководство пользователя, включающее назначение системы, основные характеристики, требование к окружению, инструкция по использованию, инструкция по установке, возможные отказы, их причины и способы устранения;
- документация для разработчика - руководство разработчика, включающее назначение системы, функциональную спецификацию, подробное описание архитектуры, описание входов, выходов и интерфейсов, описание структуры данных, описание особенностей реализации, в т.ч. схемы взаимодействия классов и их описание, основные события и реакции на них; требования к окружению и их объяснение, характеристики и результаты тестов производительности, возможные отказы, их причины и способы устранения, история изменений.

4.2 *Использование JavaSpaces*

Разработать прототип системы заказов, обеспечивающей взаимодействие покупателей, дистрибьюторов и продавцов. В систему входят ряд складов с готовой продукцией и несколько фирм – дистрибьюторов, отличающихся ценовой политикой. Итоговая цена формируются следующим образом: склады выставляют цены на продукцию, а дистрибьюторы добавляют к ним свою наценку, зависящую от вида продукции, ее количества и скорости доставки. Изначально клиенты выставляют заказы на продукцию всем своим дистрибьюторам (указывая при этом сроки доставки, необходимый диапазон цен, etc.); последние, в свою очередь, опрашивают склады и, выбирая наиболее выгодные предложения, представляют его клиентам.

Предложения складов определяются следующими параметрами: срок поставки, количество товара, стоимость продукции (последняя может варьироваться, в зависимости от срока поставки). Предложения дистрибьюторов определяются параметрами предложений складов и собственной ценовой политикой (наценками). Конкретная реализация задачи (выбор видов продукции, формализация описания ценовой политики дистрибьюторов и складов, а также способы хранения данных на складе) остается разработчику.

Задачами разработки являются:

1) создание “складских” и “дистрибьюторских” рабочих мест. Необходимо разработать формальное описание политики сбыта (в виде набора параметров) и обеспечить возможность ее модификации пользователем. Также, в случае склада нужно позволить пользователю изменять содержимое складской базы данных.
2) обеспечение взаимодействия дистрибьюторов и складов - выполнения сложных запросов типа “товар X, в количестве n штук со сроком поставки до dd:mm и стоимостью не более P” и задание крайних сроков для подачи предложений (после этого запрос снимается и предложения не рассматриваются). Также необходимо реализовать возможность подачи складом сложных предложений (типа “стоимость P_1 при поставке до d_1d_1:m_1m_1, стоимость P_2 при поставке до d_2d_2:m_2m_2”; эту проблему можно решить и разбивая одну сложную заявку на несколько простых).
3) автоматизация процедуры селекции предложений дистрибьютором. Клиенту должны представляться лишь “несравнимые” заявки (то есть если одна из заявок по всем параметрам “хуже” другой, то она должна автоматически отсеиваться). Аналогичная процедура должна производиться и приложением клиента перед выдачей предложений дистрибьюторов заказчикам.
4) организация взаимодействия клиента и дистрибьютора: клиент выставляет заявку, с ограниченным сроком действия, содержащую его требования (параметры заявки), а также несколько полей, модифицируемые дистрибьютором – общее количество предложений и сами предложения в стандартном формате (они могут, например, состоять всего из трех полей Цена/Срок поставки/ID дистрибьютора); после окончания срока подачи предложений клиент забирает заявку, анализирует ее и представляет пользователю несколько вариантов покупки.

Все приложения должны быть реализованы на Java. Складские рабочие места – в виде апплетов. Серверы, с которых считываются апплеты отличаются от тех, на которых поддерживаются складские БД (таким образом, в соответствии с правилами безопасности Java, апплет-администратор БД не может самостоятельно устанавливать сетевые соединения с хостом БД).

Предлагается следующая схема взаимодействия компонент такой распределенной системы, использующая технологии JavaSpaces и RMI:

- *Приложения типа Client.* Задачами этого приложения будут реализация GUI и формализация запросов заказчика. Общение с приложениями дистрибьюторов осуществляется через JavaSpaces. Client выставляет свою заявку (объект с ограниченным сроком действия) в своем Space-е. Каждый из дистрибьюторов, работающих с этим клиентом читает заявку и формирует свои предложения. Затем дистрибьютор модифицирует заявку, добавляя свои предложения и

увеличивая счетчик предложений[5]. После окончания срока подачи предложений Client забирает заявку и анализирует предложения.

- *Приложения типа Distributor*. Они считывают заявки заказчиков и формируют на их основе предложения для складов. Взаимодействие со складами также осуществляются через Space-ы. Дистрибьюторы выставляют свои заявки (помещают соответствующие объекты в специальный Space), а складские приложения читают их, анализируют и размещают свои предложения в том же Space-е. Затем, после истечения срока подачи, дистрибьюторы перебирают все объекты – предложения складов, отбрасывая худшие и формируют предложение дистрибьютора. В завершение, дистрибьюторы забирают (Take) из клиентского Space заявку клиента, модифицируют ее и записывают обратно.
- *Приложения типа Warehouse*. Они считывают заявки заказчиков дистрибьюторов и формируют на их основе предложения (см. выше).
- *Апплеты Warehouse – Administrator* должны осуществлять модификацию складских БД. Апплеты реализуют GUI, формируют запросы по модификации и вызывают методы rmi-сервера для их выполнения.
- *Приложение Rmi-сервер.* Его задачами является.
  - установка соединения с базой данных
  - выполнение запросов апплетов Administrator, по модификации складских баз. Эти методы Rmi-сервера и должны удаленно вызываться клиентами.
- *Базы данных складов, реализуемые в виде таблиц Oracle 8i или 9i..*

Средством разработки может быть любая версия JDK, начиная с 1.1. Возможно использование визуальных сред типа Visual Cafe, Visual J++, JBuilder или Visual Age. Необходимая документация и различные версии JDK присутствуют на сервере фирмы Sun (http://java.sun.com). Пользовательский интерфейс можно разработать с помощью JFC/Swing.

В отчет по заданию должны входить:

- откомпилированный работающий код;
- исходные тексты программ с комментариями;
- пользовательская документация - руководство пользователя, включающее назначение системы, основные характеристики, требование к окружению, инструкция по использованию, инструкция по установке, возможные отказы, их причины и способы устранения;

[5] Для обеспечения устойчивости системы необходимо реализовать весь набор операций с одной заявкой в виде единой транзакции (тогда, в случае сбоя в работе одного из Distributor-ов при модификации объекта – заявки, вся операция не будет заблокирована, а вернется к исходному состоянию).

- документация для разработчика - руководство разработчика, включающее назначение системы, функциональную спецификацию, подробное описание архитектуры, описание входов, выходов и интерфейсов, описание структуры данных, описание особенностей реализации, в т.ч. схемы взаимодействия классов и их описание, основные события и реакции на них; требования к окружению и их объяснение, характеристики и результаты тестов производительности, возможные отказы, их причины и способы устранения, история изменений.

5 Приложение. Установка и настройка программного обеспечения

5.1 *Установка сервера приложения JBoss*

Для установки *JBoss 4.0* необходимо иметь предустановленную *Java 1.4* или *JAVA 5.0*. *JAVA 5.0* требуется для поддержки спецификации *EJB3*. Определяем версию *JAVA*. Это можно сделать при помощи команды:

```
java -version
```

Если установлена *JAVA 1.4*, то в этом случае сервер приложений *JBoss* не будет поддерживать спецификацию *EJB 3.0*. Если необходима поддержка этой спецификации, то следует скачать с сайта http://java.sun.com/ последнюю версию *JAVA 5.0* и установить ее.

Сервер приложений *JBoss* доступен по адресу http://www.jboss.org/. На момент редактирования этого документа последней версией был пакет jboss-4.0.4.GA-Patch1-installer.jar. Копируем скачанный архив с *JBoss* в папку, из которой будет производиться инсталляция, и выполняем команду:

```
java -jar jboss-4.0.4.GA-Patch1-installer.jar
```

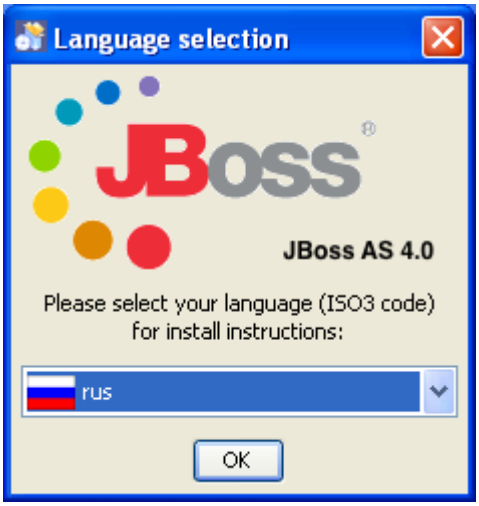

Рис. 1. Выбор языка инсталляции

В качестве последнего аргумента введите название скачанного архива. После этого начинается процесс инсталляции (рис. 1). В качестве языка инсталляции можно выбрать русский. После нажатия кнопки OK начинается непосредственно сам процесс инсталляции. Сначала Вам будет показано окно приветствия, затем описаны возможности *JBoss*,

после этого предложат согласиться с условиями лицензии *GNU Lesser General Public License*. Затем необходимо выбрать каталог установки.

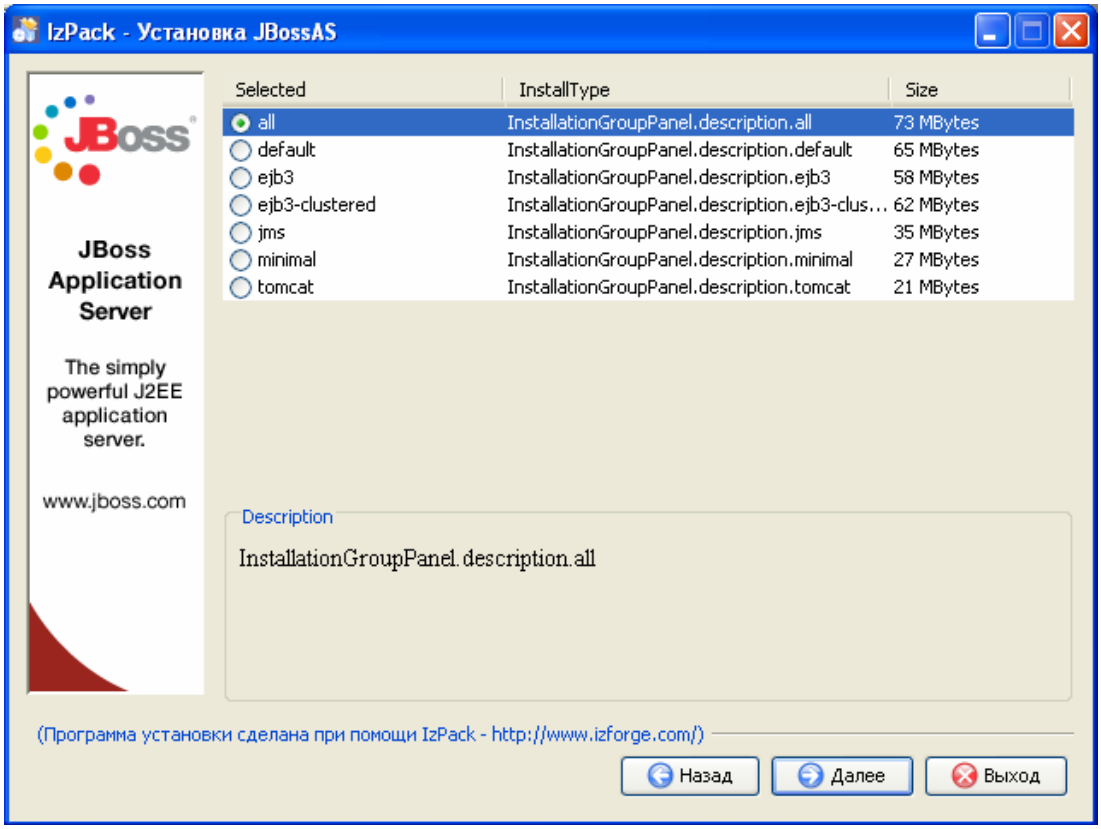

Рис. 2. Устанавливаемые компоненты

Важным окном является окно, изображенное на рис. 2. Здесь можно выбрать конфигурацию устанавливаемого сервера. Можно, не стесняясь, выбрал пункт *all* – в настоящем методическом пособии представлены примеры работы как с *EJB 2.1*, так и с более современным *EJB 3.0*. Стоит отметить, что в полную конфигурацию входит также и контейнер *JSP Tomcat* – так что устанавливать его дополнительно не обязательно.

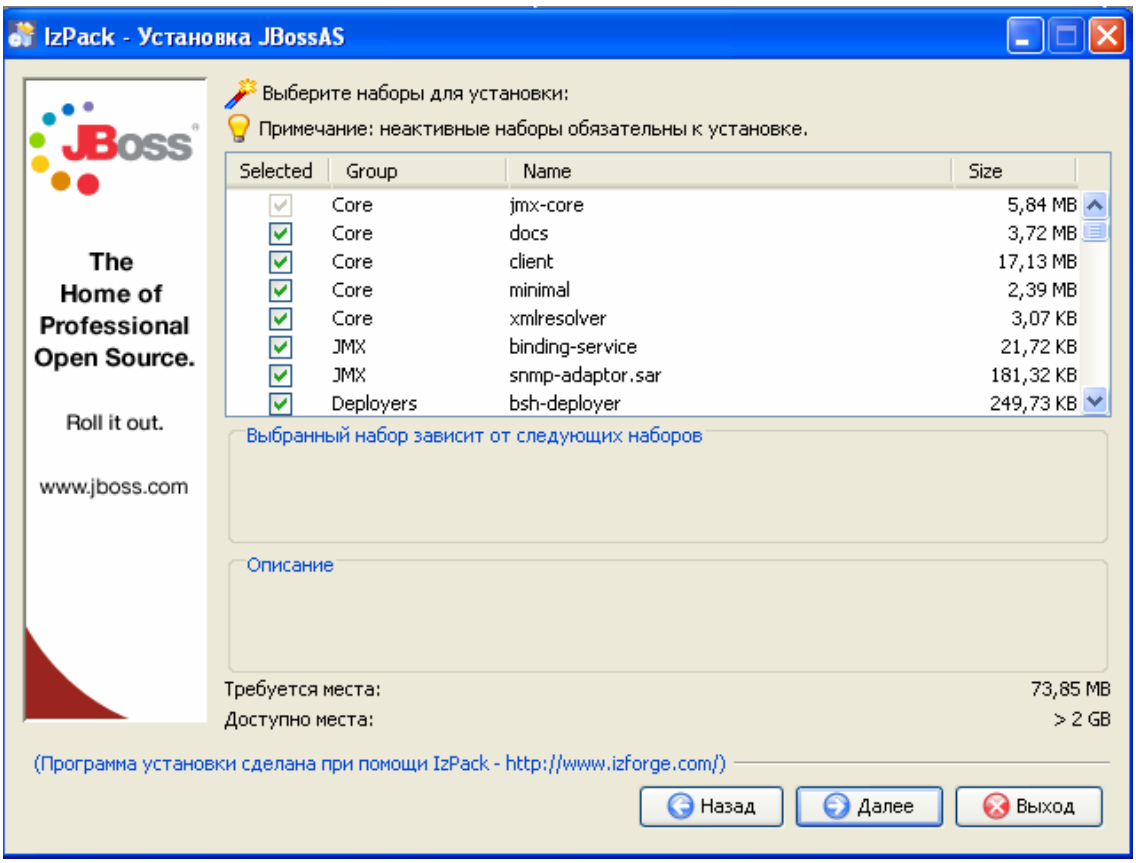

Рис. 3. Модули для установки

После этого будет предложено выбрать модули для установки (рис. 3) – тут также имеет смысл выбрать все модули и продолжил.

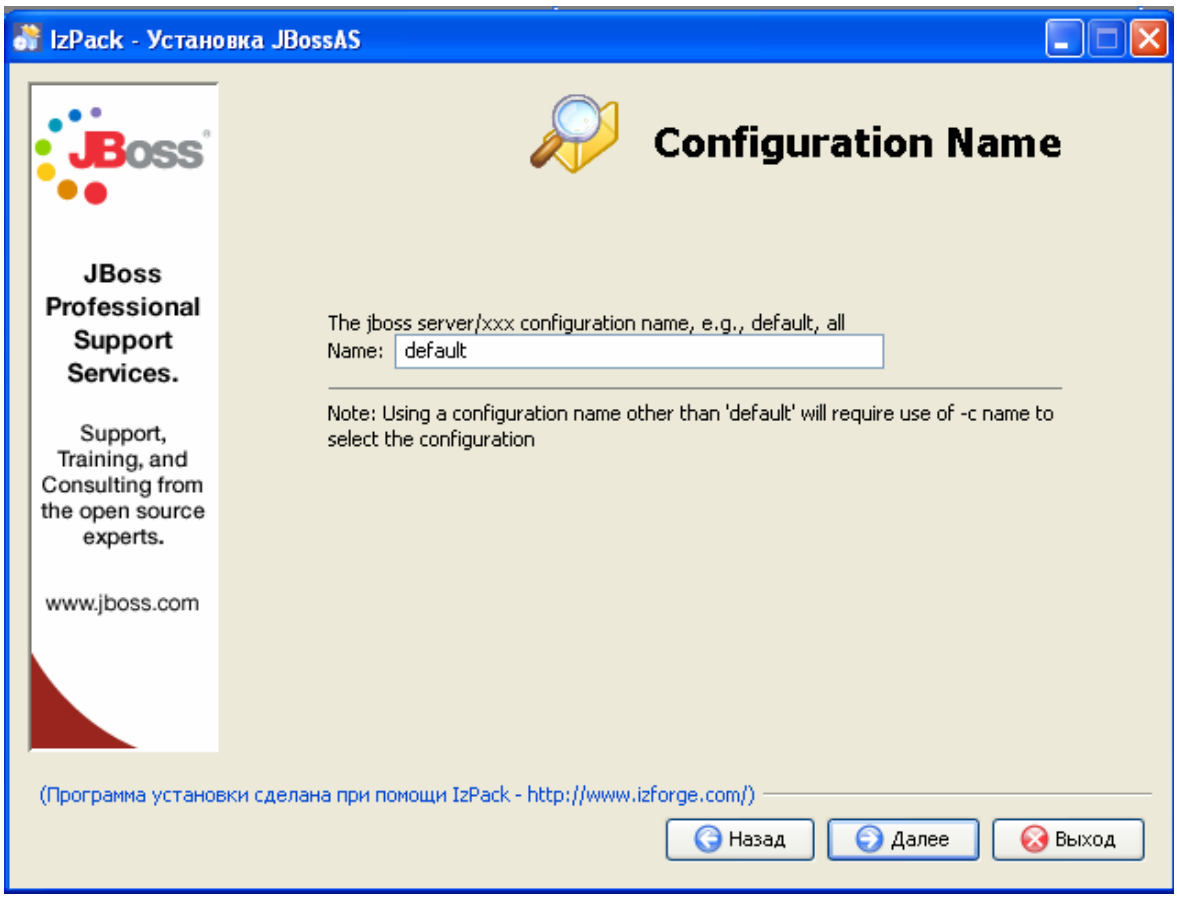

Рис. 4. Название конфигурации

В следующем окне (рис. 4) следует выбрать название конфигурации. Это название будет соответствовать названию каталога, находящегося в каталоге server установочного каталога *JBoss*. В этот каталог впоследствии будут помещаться файлы компонентов *EJB*. Если выбрать для него отличное от default название, то для запуска с этой конфигурацией необходимо будет добавлять параметр –c name к команде запуска *JBoss*, где name соответствует выбранному названию конфигурации.

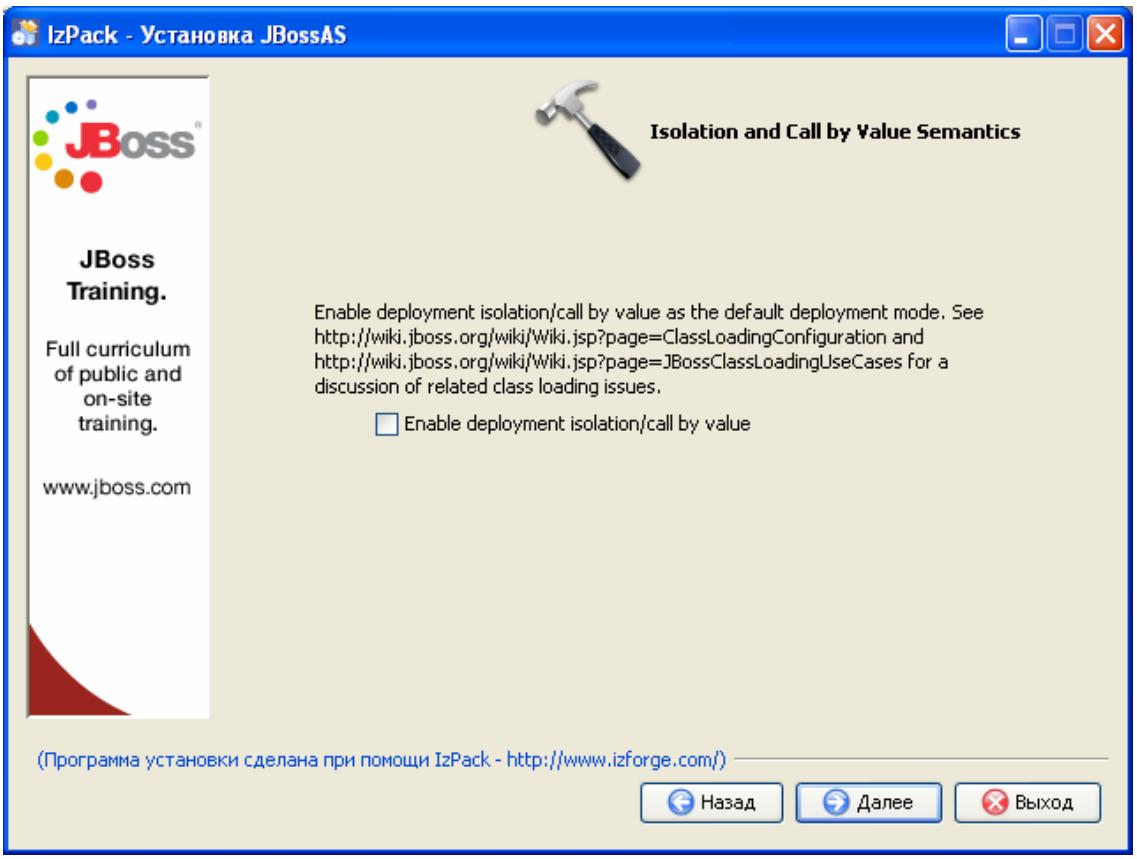

Рис. 5. Изоляция

Далее следует установка параметра isolation (изоляция). В старых версиях *JBoss* использовалась плоская модель загрузки файлов – классы каждого компонента *EJB* и веб-приложения загружались в общее пространство классов. В связи с этим могли возникать проблемы связанные с тем, что разные приложения определяют по-разному класс с одним и тем же именем. Этих проблем можно избежать, если использовать для размещения класса соответствующие приложению пакеты. Но иногда все-таки возникают проблемы – например, необходимо на одном *JBoss* сервере запустить старую и новую версию одного веб-приложения. Изоляция классов помогает “изолировать” классы в рамках одного приложения. У нас таких проблем не должно возникнуть, поэтому не ставим галочку.

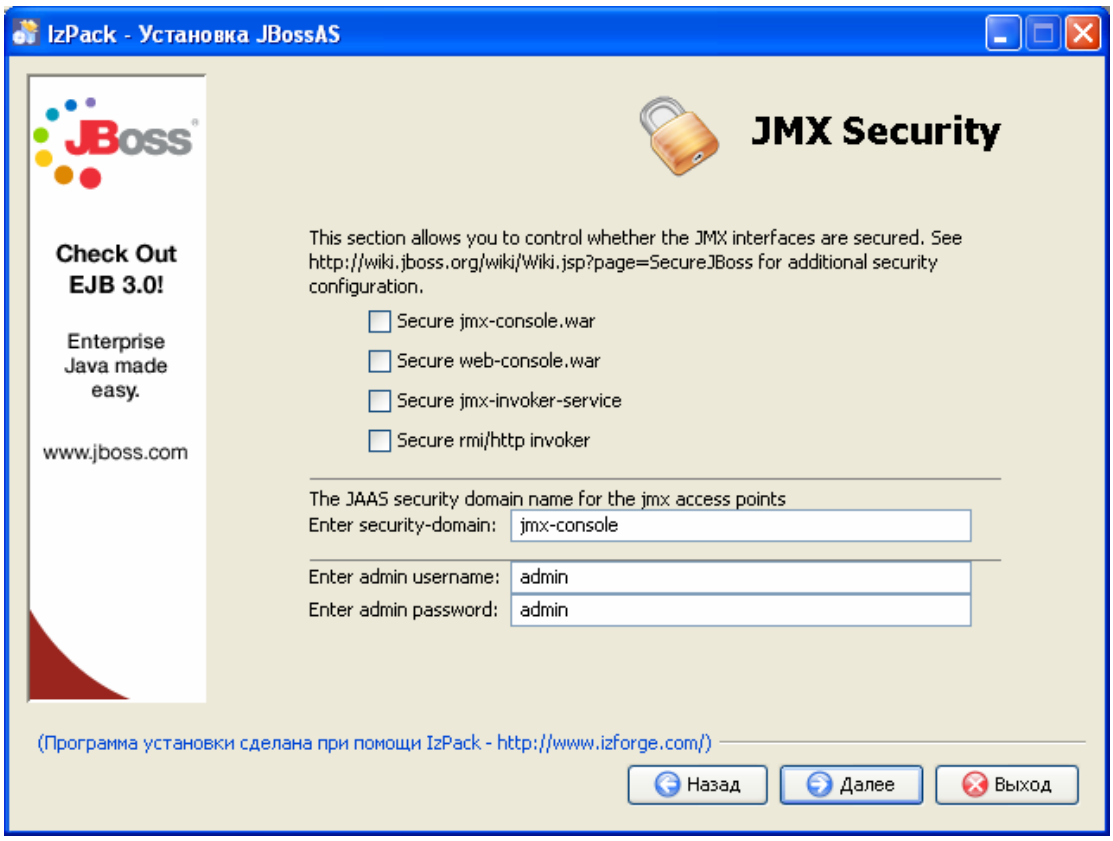

Рис. 6. Параметры безопасности

Далее следуют настройки параметров безопасности. По умолчанию они все отключены в *JBoss*. Как говорилось в известной книжке “UNIX Руководство системного администратора”, **чем безопаснее система, тем сложнее в ней работать**. Поэтому для учебных целей вполне можно ни одну галочку в предложенном списке опций безопасности не ставить. Тем, кому будет интересно настроить различные защищенные интерфейсы *JBoss*, разработчики предлагают обратиться по следующему адресу – http://wiki.jboss.org/wiki/Wiki.jsp?page=SecureJBoss.

После этого больше никаких важных опций нет. Через экран, на котором описаны все параметры установки, начинается непосредственно сам процесс инсталляции. После инсталляции вам будет предложено создать *XML*-сценарий автоматической установки *JBoss* в данной конфигурации. Если это сделать, то в корневой каталог установки *JBoss* будет помещен XML файл, описывающий только что проведенный сценарий установки. Его можно потом будет использовать при следующей установке:

```
java -jar jboss-4.0.4.GA-Patch1-installer.jar autoscen.xml
```

5.2 *Установка базы данных Oracle 9i*

Для того, чтобы установить базу данных *Oracle*, необходимо сначала скачать архив с сайта http://www.oracle.com/. В настоящем

пособии использована версия 9.2.0.1.0. Для некоммерческого использования она бесплатна. Затем необходимо запустить инсталлятор. После выбора установочного каталога, следует выбрать устанавливаемый продукт – *Oracle Database 9.2.0.1.0*.

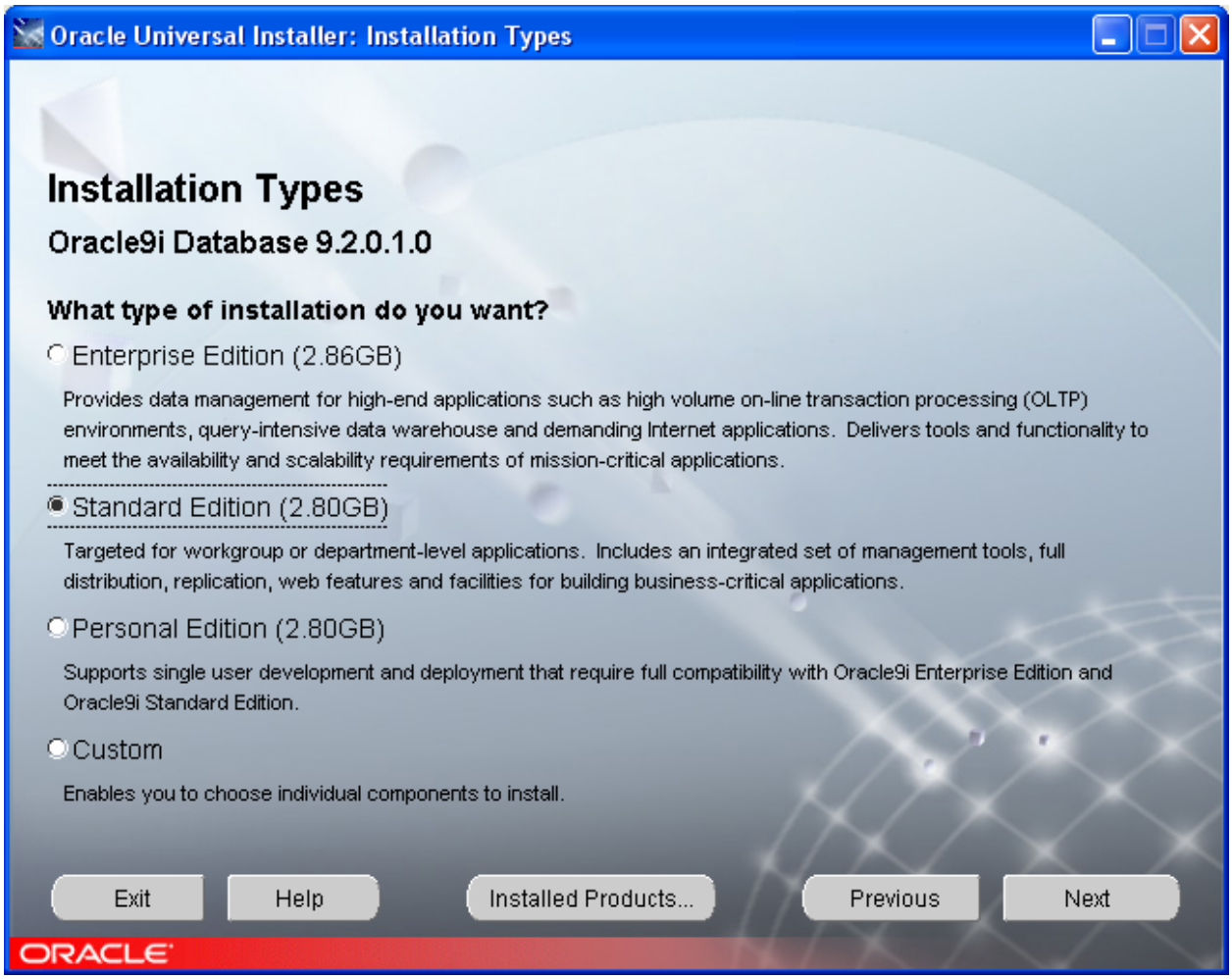

Рис. 7. Тип установки

Затем Вам будет предложено выбрать тип установки (рис. 7) – как бы не был велик соблазн выбрать *Enterprise Edition*, делать этого не рекомендуется. В *Enterprise Edition* входит помимо всего прочего поддержка кластеризации базы данных. Опыт показал, что в этом варианте могут возникать проблемы с созданием после установки самой базы данных из-за часто возникающих исключительные ситуации. Поэтому выбираем *Standart Edition* и продолжаем. Далее в качестве конфигурации базы данных выбираем пункт *General Purpose* (общего назначения).

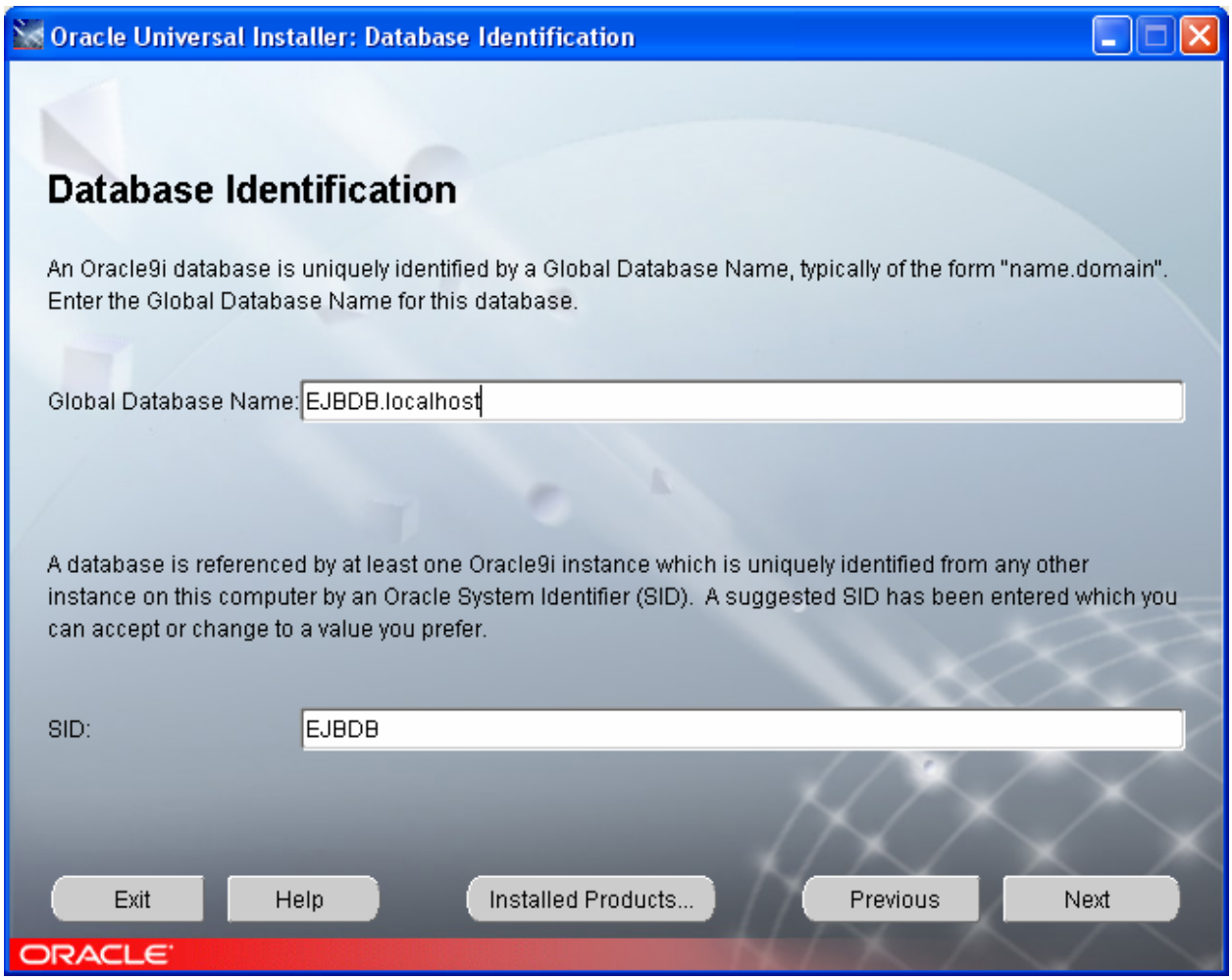

Рис. 8. Название базы данных

Название для базы данных состоит из двух частей – из *SID* (идентификатора) базы данных и доменного имени сервера, где она располагается. В нашем случае это *EJBDB.localhost*, где *EJBDB* – это *SID*, а *localhost* – это доменное имя. Далее нам будет предложено выбрать каталог на диске, где будут располагаться файлы базы данных. В качестве кодовой таблицы можно выбрать стандартную кодировку *Windows* (на русифицированной *Windows* – это *cp1251*). Далее начинается сам процесс инсталляции.

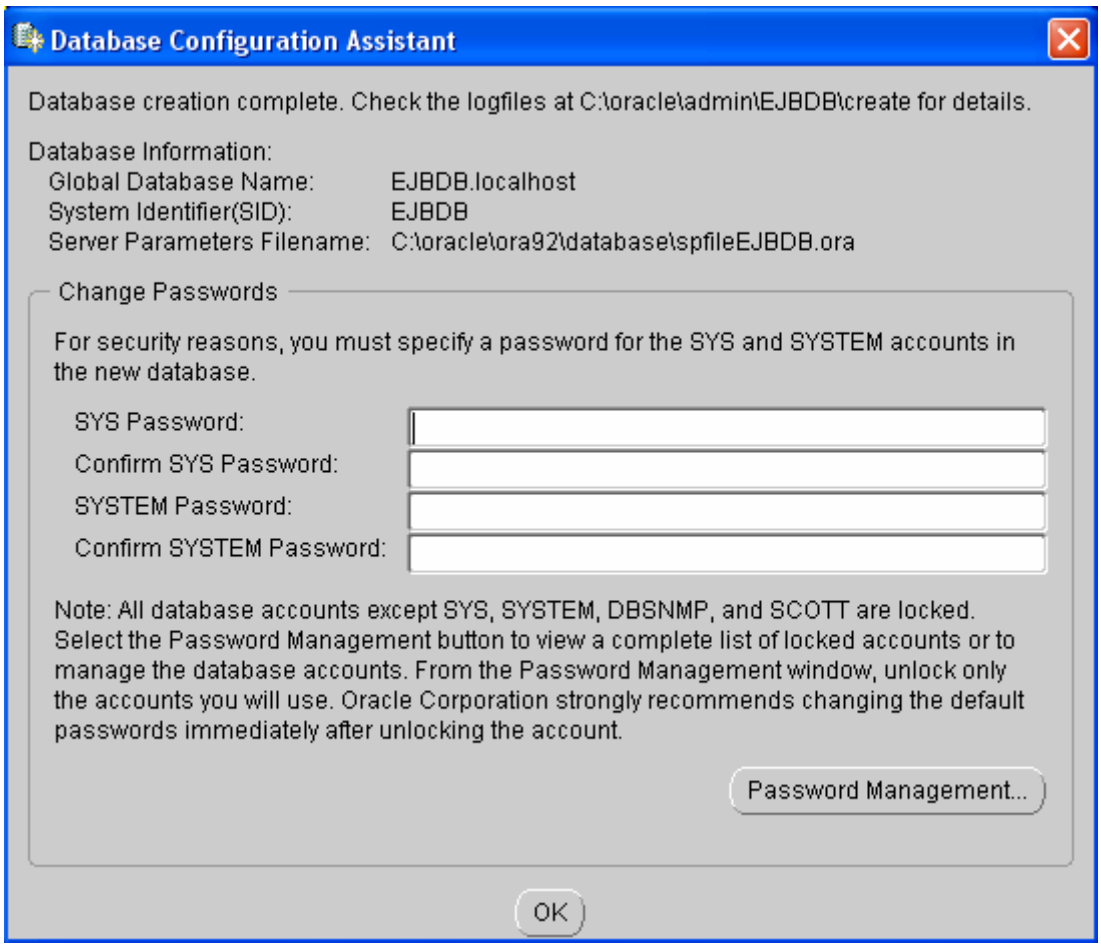

Рис. 9. Системные пароли

В конце инсталляции будет предложено выбрать системные пароли (для пользователей *SYS* и *SYSTEM*) для созданной базы данных. Данные два пользователя несколько отличаются привилегиями по управлению базой данных.

Установка закончена. После этого следует сделать еще несколько операций. Во-первых, открыть *Enterprise Management Console* из меню Пуск. Выбираем в дереве слева только что созданную базу данных.

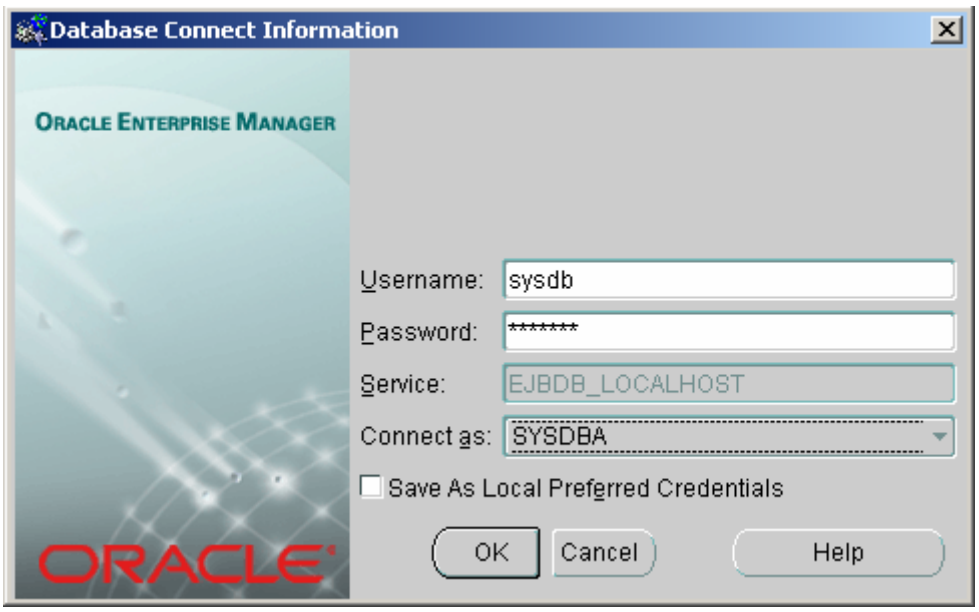

Рис. 10. Авторизация

Выполняем вход в менеджер управления базой данных под именем *SYSDB*. Тип подключения – *SYSDBA*. Необходимо сделать две вещи – создать пользователя, через которого будут выполняться подключения к базе данных из компонент *EJB*, а также поменять порт *XML* базы данных с 8080 на какой-нибудь другой незанятый порт. Просто порт 8080 используется по умолчанию как стандартный порт для контейнера *JSP Tomcat*. А он уже поставлен вместе с *JBoss*. И в этом случае запустить *JBoss* не удастся – он выдаст сообщение об ошибке, что этот порт уже занят. Но даже если поменять порт *Tomcat* на какой-нибудь другой, запустить *JBoss* все равно не удастся. Возможно это ошибка в самом сервере *JBoss*.

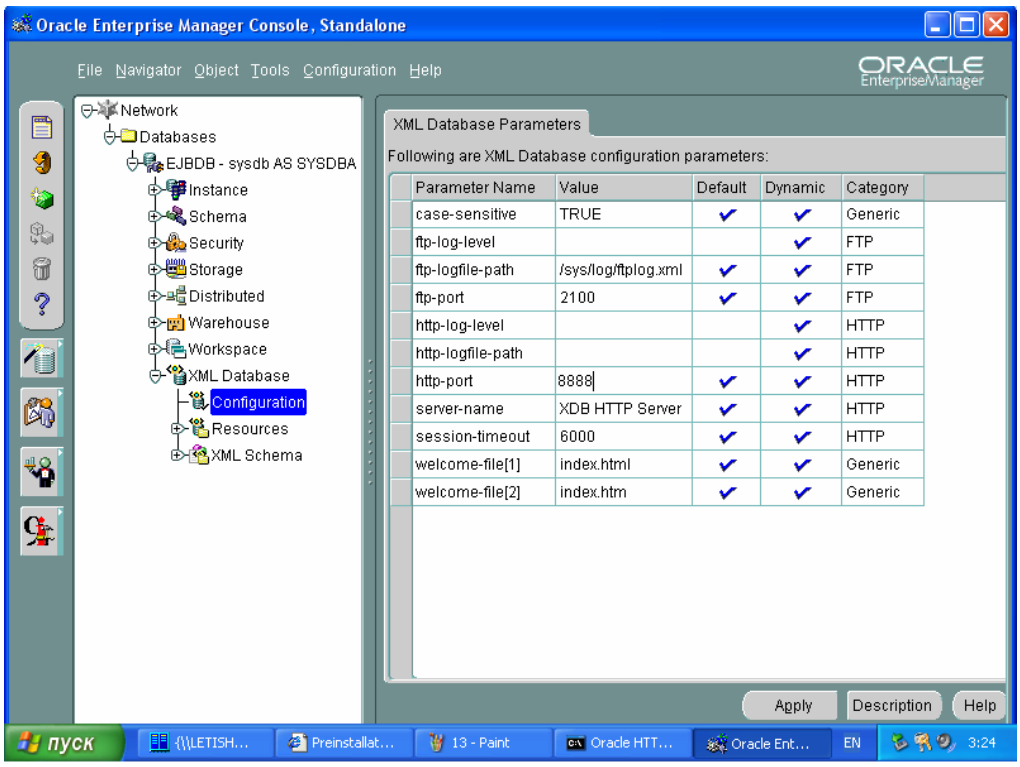

Рис. 11. Смена порта

Выбираем *http*-порт базы данных и меняем его значение с 8080 на какое-нибудь другое – к примеру, 8888.

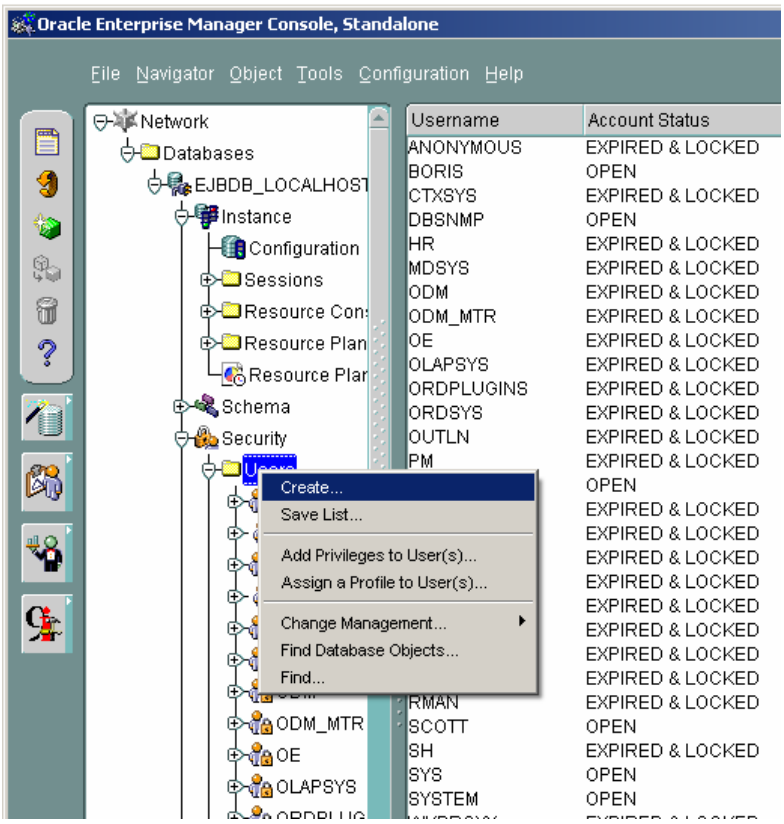

Рис. 12. Создание пользователя

Далее необходимо создать пользователя СУБД для того, чтобы подключаться к базе данных (рис. 12).

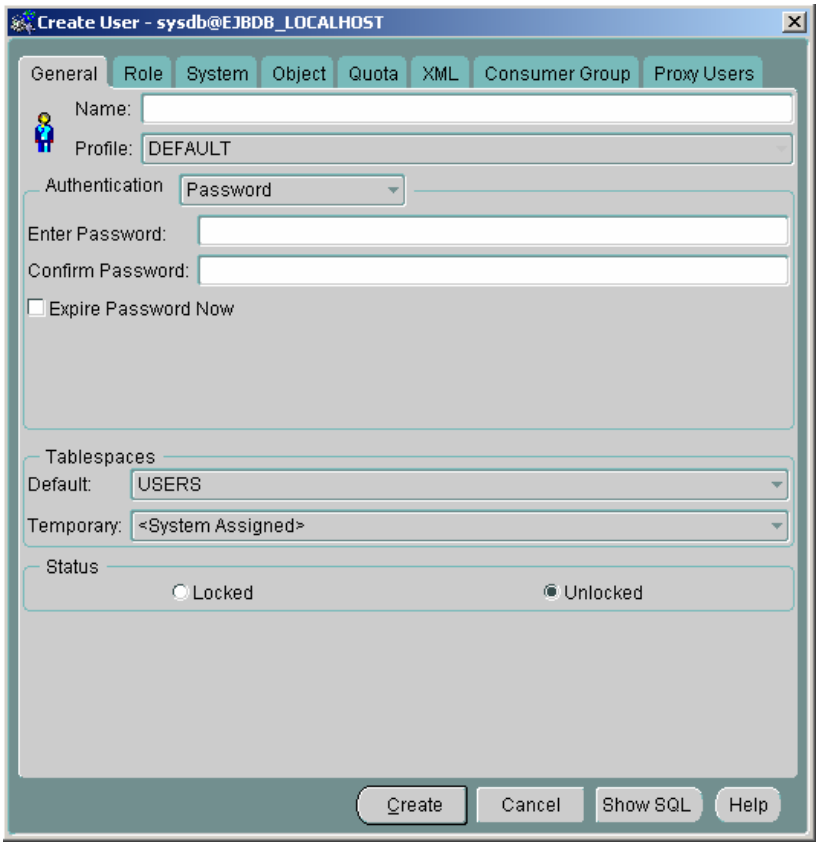

Рис. 13. Параметры пользователя

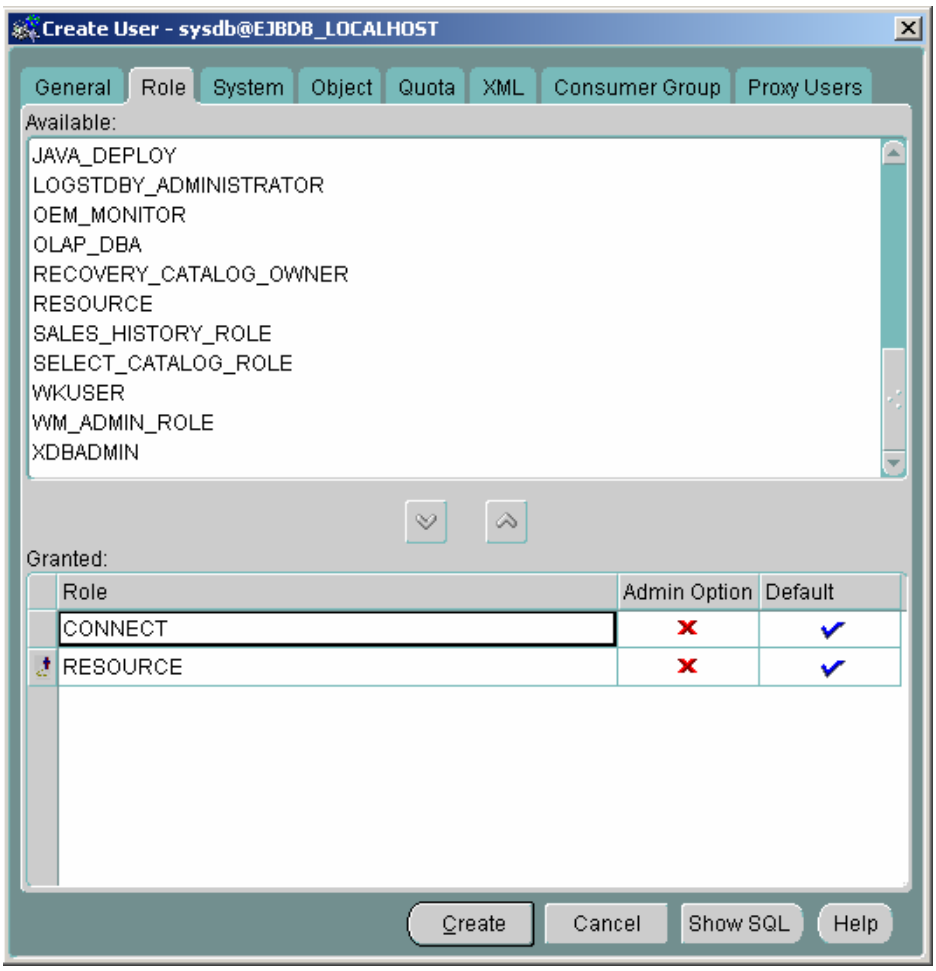

Рис. 14. Добавление роли

Для пользователя устанавливаем имя, пароль и наделяем его ролью *RESOURCE*, для того, чтобы он мог создавать таблицы и выполнять запросы (рис. 13, рис. 14).

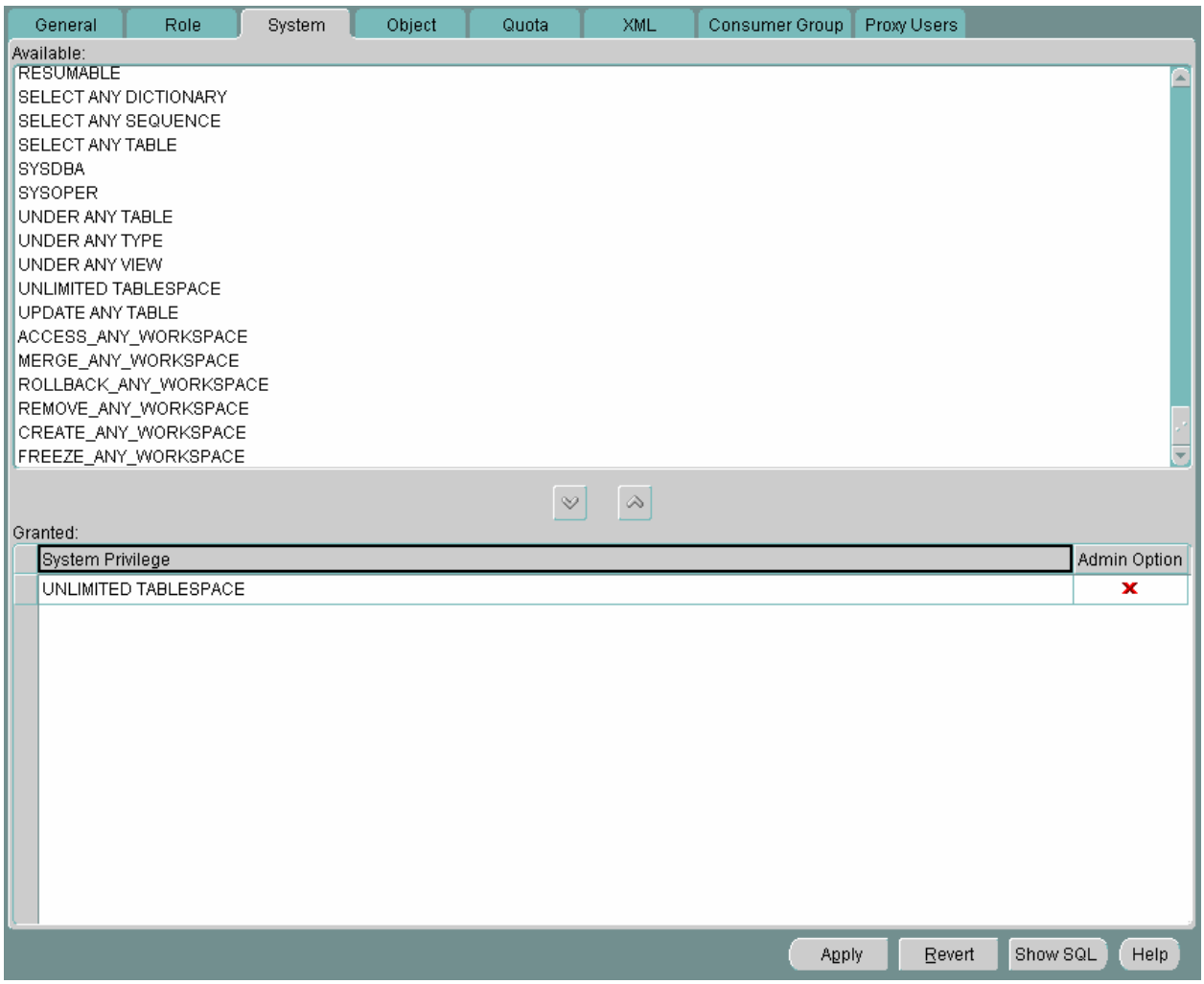

Рис. 15. Выделение неограниченной памяти для таблиц

После этого осталось дать пользователю неограниченное пространство для размещения таблиц. Это делается во вкладке System (рис. 15).

5.3 *Установка Eclipse-WTP*

Для того, чтобы установить *Eclipse Web Tools Platform*, необходимо вначале скачать архив с сайта http://www.eclipse.org/. Архив использованной в данном пособии версии wtp-all-in-one-sdk-R-1.5.2-200610261841-win32.zip. Никакого инсталлятора у него нет – для установки достаточно распаковать архив в соответствующую папку. В нашем случае случае это была папка C:\ejb\eclipse-wtp\. Стоит отметить, что если к моменту установки *Eclipse* уже была поставлена *Oracle database 9i*, то запустить *Eclipse* не удастся. Дело в том, что для работы *Oracle* устанавливает *JAVA* версии 1.3. В результате чего переменные окружения PATH и JAVA_HOME меняются, и *Eclipse*, не найдя по указанным путям *JAVA 1.4* либо *JAVA 5.0*, отказывается запускаться.

Проблема может быть решена следующим образом – создаем файл eclipse.bat, из которого запускается *Eclipse*.

Вот его содержимое:

```
set PATH=C:\Program Files\Java\jdk1.5.0_02\bin\
set JAVA_HOME=C:\Program Files\Java\jdk1.5.0_02
eclipse
```

Переменным нужно установить значения, соответствующие путям, куда соответствующая версия *JAVA* была установлена изначально.

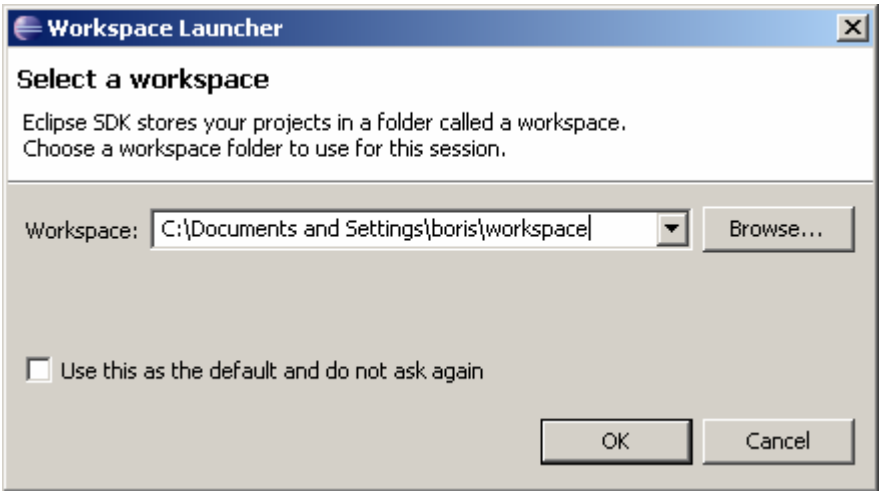

Рис. 16. Выбор места для проектов

При первом запуске вам будет предложено выбрать каталог на диске, в котором будут храниться проекты *Eclipse* (рис. 16).

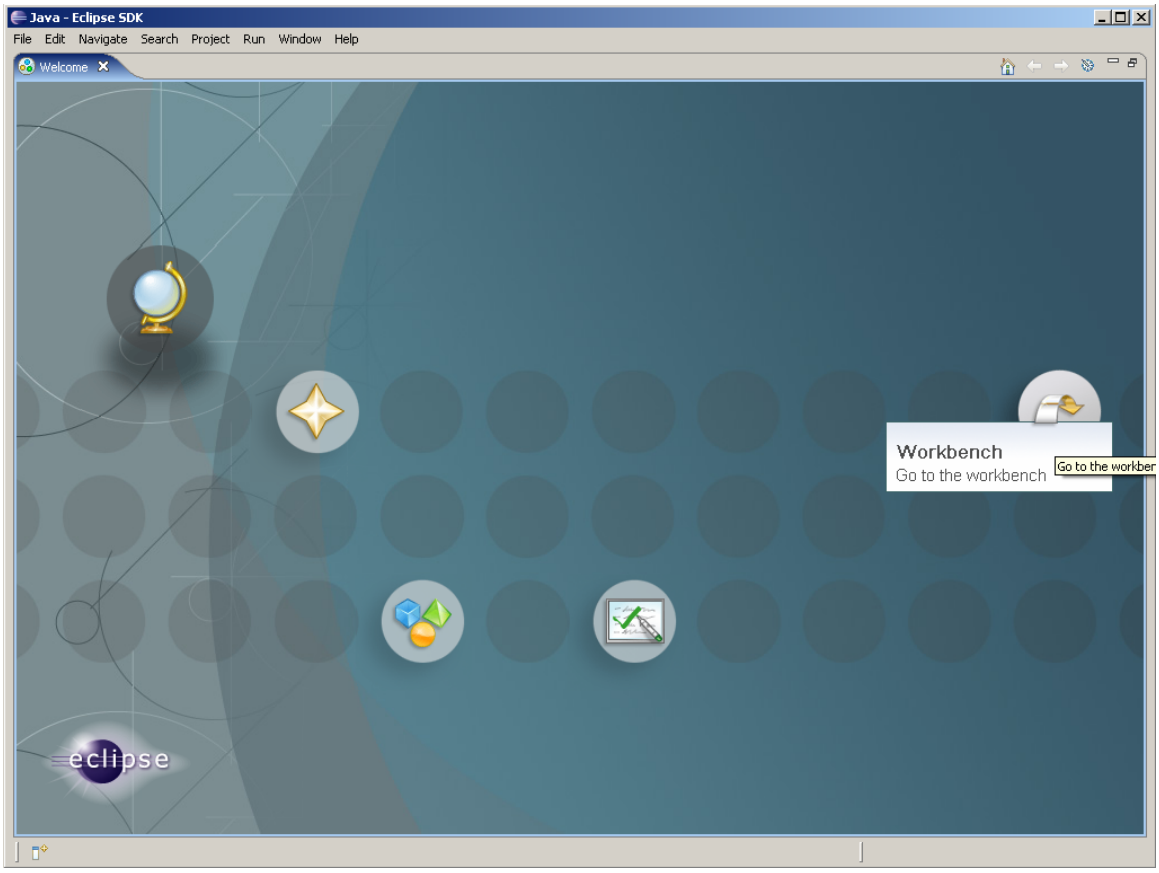

Рис. 17. Eclipse

При первом запуске отображается красивое приветственное окно (рис. 17). Можно выбрать закладку Tutorials – и изучить некоторые из возможностей *Eclipse*. Для того чтобы начать работать, следует выбрать закладку *Workbench*.

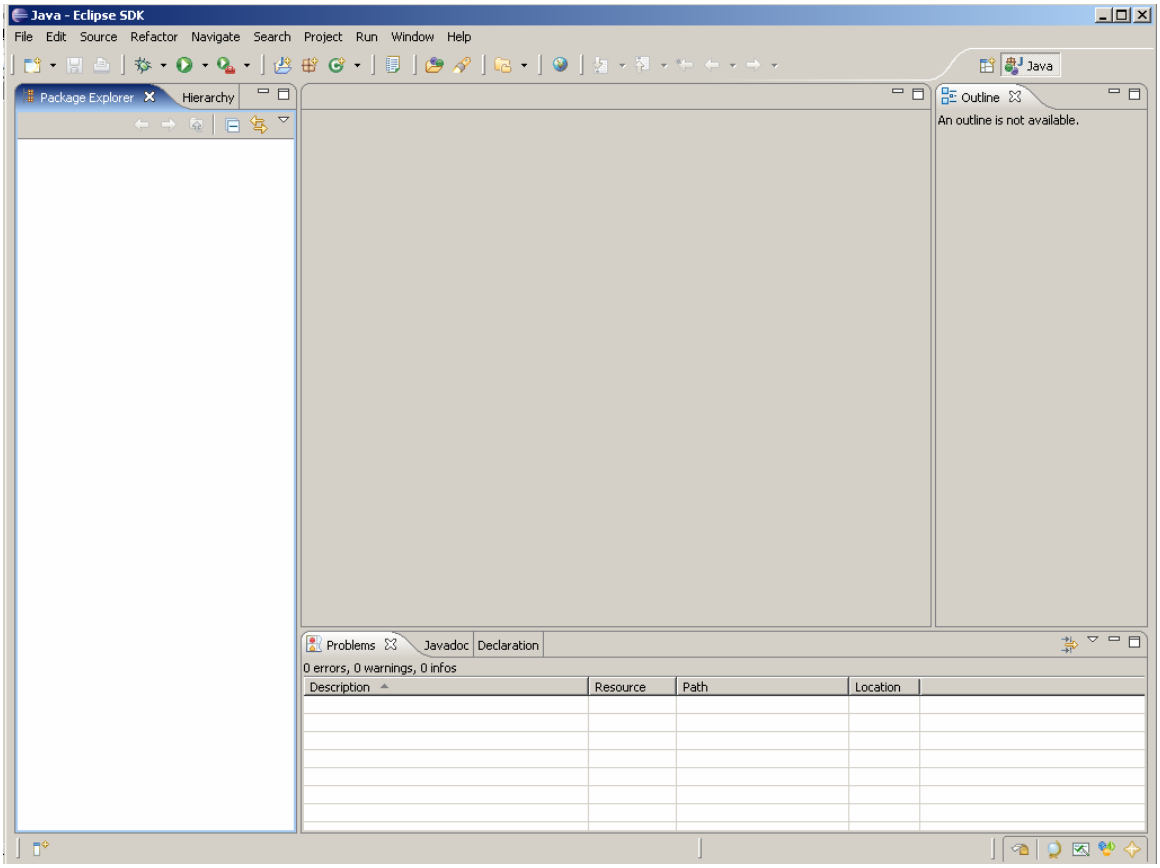

Рис. 18. “Мастерская” Eclipse

После этого можно непосредственно начинать работать с *Eclipse-WTP*. Данное методическое пособие не покрывает описание процесса работы с *Eclipse*. Однако все необходимые операции для разработки *EJB* подробно рассмотрены при рассмотрении различных примеров.

5.4 *Установка XDoclet*

Пакет *XDoclet* доступен по адресу http://xdoclet.sourceforge.net/. С его помощью *Eclipse-WTP* развертывает компоненты *EJB* в различных базах данных. Последней версией *XDoclet* является версия 1.2.3. Для установки достаточно распаковать его в один из каталогов на диске.