JavaOne℠

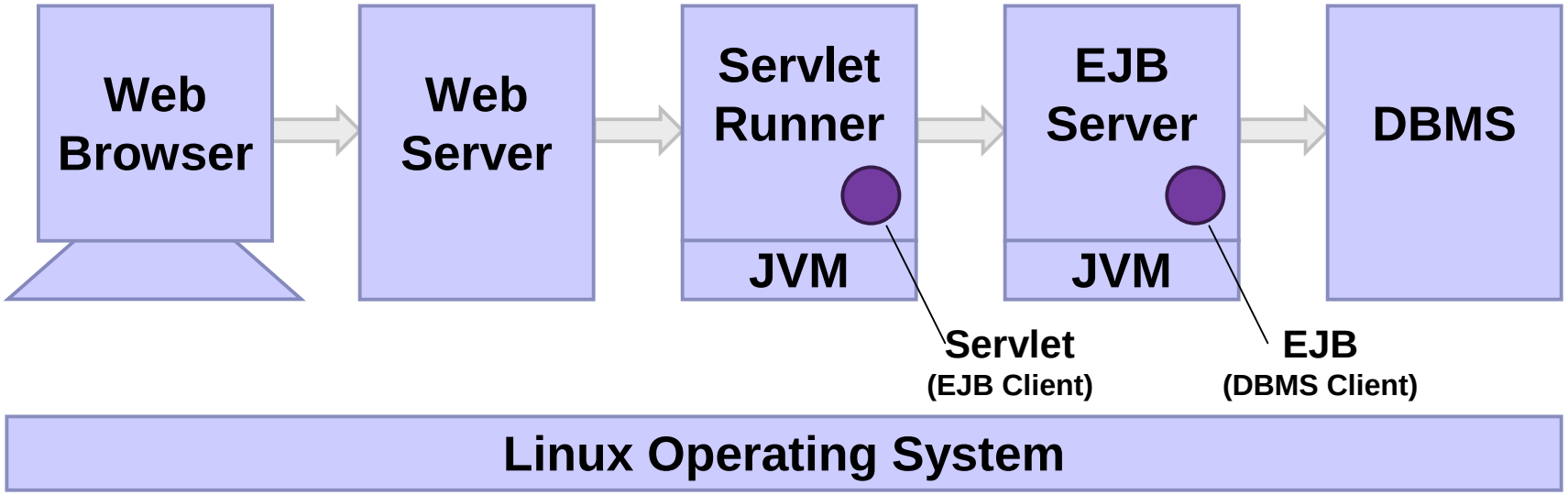
Sun's 2000 Worldwide Java Developer Conference℠

Enterprise Java™ Technology for Linux HOWTO

Lior Sharon
Chief Technology Advisor
Niragongo, Inc.

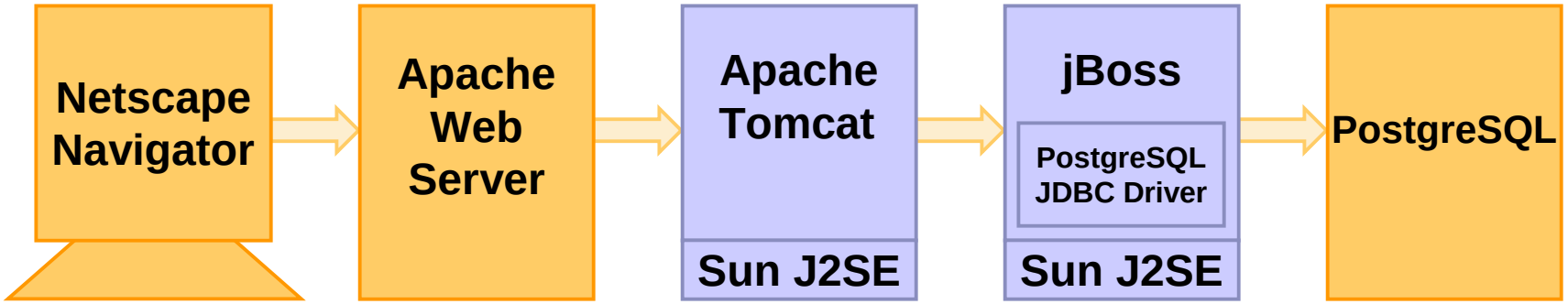
Presentation Objective

Demonstrate an end-to-end example of the key Enterprise Java APIs on the Linux platform



Presentation Objective

Demonstrate an end-to-end example of the key Enterprise Java APIs on the Linux platform



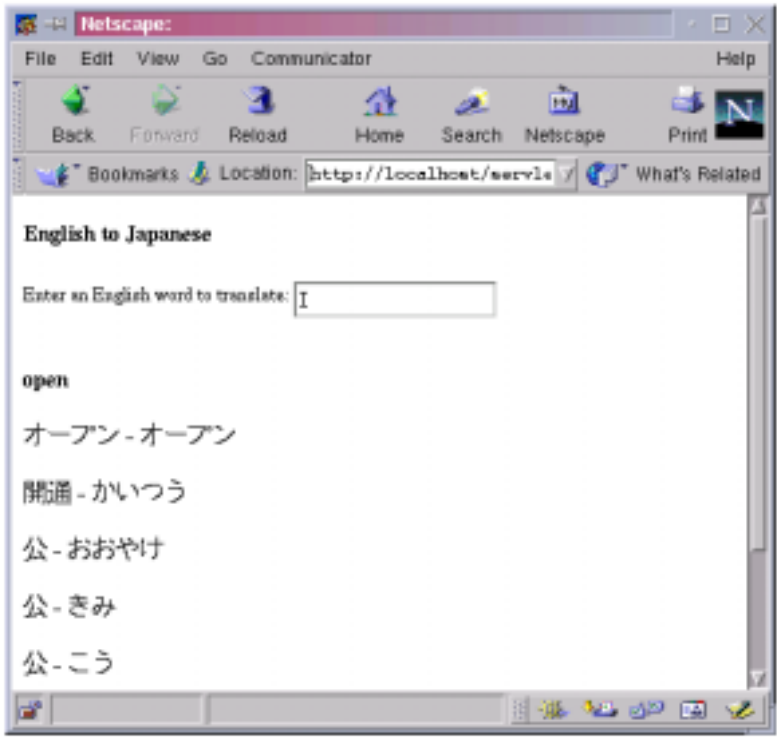
For Each:

- Background
- How to Download
- How to Install
- How to Setup the Environment
- How to Test with a Sample Application
- How to Obtain More Information

Legend

- Components provided with RedHat 6.2 distribution
- Components we'll primarily discuss

End-to-End Example Screenshots



- The Open Source EDICT Japanese/English dictionary has been loaded into PostgreSQL tables

- The example application is an English-to-Japanese dictionary server
- The whole implementation is about 150 lines of code
- Example illustrates the Java platform's built-in support for multi-byte characters through Unicode

“Enterprise Java”

- “Enterprise Java” is used to mean a certain set of APIs
  - APIs such as: Servlet, JSP™, JDBC™, EJB™ architecture
  - Others include: JNDI, JTS, JMS
- This presentation focuses on the most popular APIs
- “Enterprise” in this context should not simplistically be perceived to mean more scalable, reliable, manageable, secure, etc.

Caveats

- The examples in this presentation and the suggestions in the HOWTO are provided to help you get an Enterprise Java environment up and running on Linux as quickly as possible
- The examples are not product recommendations or endorsements
- It is up to you to determine what design and products are best for your particular purpose

RedHat 6.2 Base Installation

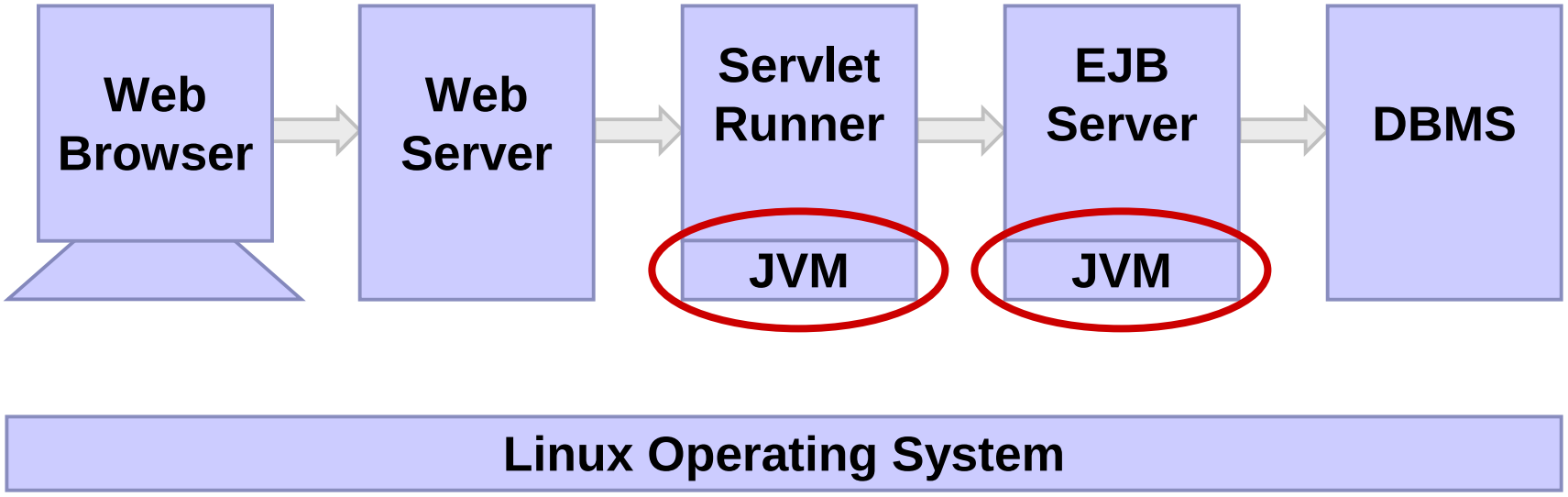
- The base installation used
  - RedHat 6.2
  - “Developer Workstation” installation option
  - Netscape, Apache, PostgreSQL, and Kaffe and various utilities are already installed
  - rpm -q
  - Examples use Bourne Again Shell (bash) syntax
  - echo $0

Linux for Windows Users

Linux	Windows Equivalent	Description
rpm	Similar the setup.exe and Install Shield but significantly more powerful	Software package installer used by RedHat and several other distributions. KDE kpackage is a GUI front end.
echo	echo	
mkdir	mkdir	
mv	move	
tar	zip and unzip	KDE ark is a GUI front end.
ls	dir	
which	No equivalent	Shows from where a command will be run.
export	set	Sets environment variables (bash, ash, sh, ksh).
/etc/rc.d/init.d directory	Services	Similar but more sophisticated (run levels and start order). KDE ksysv is a GUI front end.
cp	copy	
sh	cmd	Starts a Bourne new shell.
chmod	attrib	Changes file permissions. “dot”-prefixed files are hidden.

Java™ Development Kit

Software that provides the Java virtual machine and core Java software development tools



Java Development Kit Background

- Several JDK™ releases available for Linux
  - Blackdown
  - IBM
  - Kaffe
  - J2SE™ platform
  - J2EE™ platform
- Java 2 Platform, Standard Edition (J2SE™) for Linux made available December 1999

J2EE™ Platform for Linux

- Announced December 1999
- Made available May 2000
- Provides a complete reference implementation for all the Enterprise Java APIs
- An easier way to get started than the products described in this presentation
- Generally not recommended for a production environment

J2EE™ Platform Value Proposition

- Simplifies Enterprise Development
- Speeds Time to Market
- Eliminates Vendor Lock-In
- Allows to Connect to Existing Solutions

From Sun Java 2 Platform, Enterprise Edition Product Management

Java™ Technology and Linux

- Sun is committed to the Linux community
- J2EE™ platform availability on Linux due to user demand
- Linux port required porting one C-language library
- Full due diligence in QA on Linux

From Sun Java 2 Platform, Enterprise Edition Product Management

J2SE Platform Download

- The J2SE platform can be obtained from http://java.sun.com/linux/

J2SE Platform Installation

- As root, remove Kaffe, if pre-installed, to avoid confusion

```
rpm -e kaffe
```

- Install the J2SE (JDK) distribution tarball

```
cd /usr/local
tar zxvf jdk1_2_2-linux-i386.tar.gz
```

- You should now see the JDK directory

```
ls /usr/local/jdk1.2.2
```

J2SE Platform
Setting Up the Environment

- Environment variables

```
export JAVA_HOME=/usr/local/jdk1.2.2
export PATH=$JAVA_HOME/bin:$PATH
```

- Confirmation

```
which javac
which java
```

- Add exports to $HOME/.bash_profile
- Syntax will differ for C shell and other shells

J2SE Platform Test Program

- Create HelloWorld

```
public class HelloWorld {
  public static void main(String args[]) {
    System.out.println(”Hello, world!”);
  }
}
```

- Compile and run

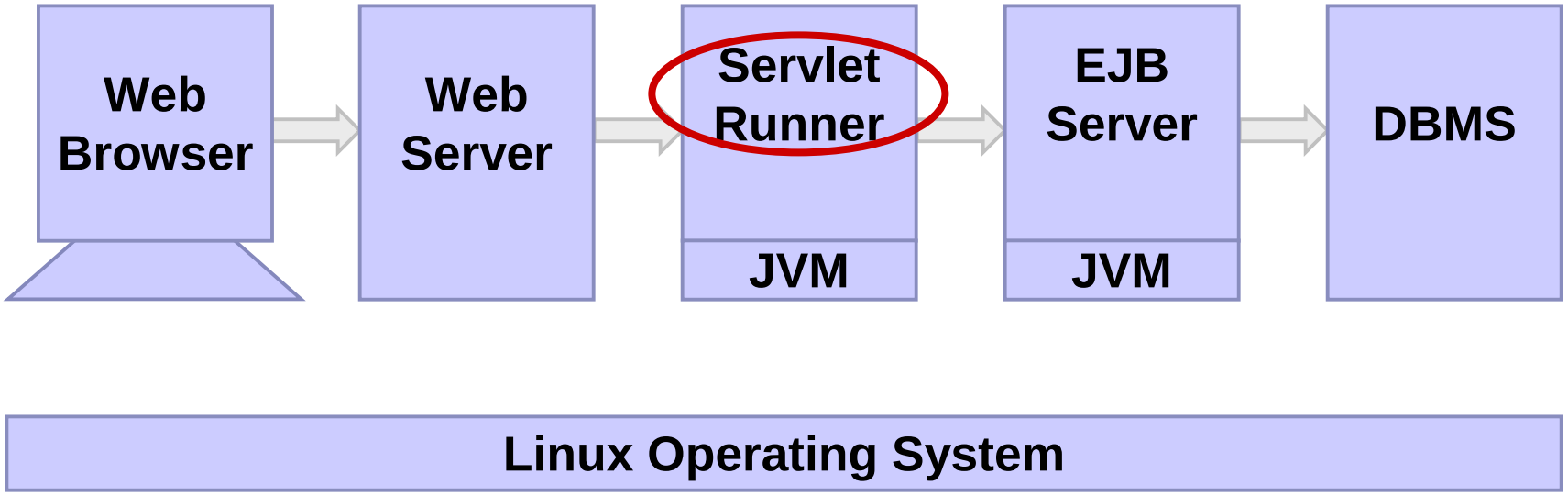
```
javac HelloWorld.java
java HelloWorld
```

- Confirmation

```
Hello, world!
```

Servlet Runner

Software that enables the web server to run Java technology-based server-side programs



Servlet Background

- Several Servlet Runners available for Linux
  - Allaire JRun
  - Apache Tomcat
  - BEA WebLogic
  - Caucho Resin
  - Lutris Enhydra
- Some packaged stand alone, others part of larger application server framework

Apache Tomcat Download

- Apache Tomcat can be obtained from
  http://jakarta.apache.org/tomcat/
- Full 100% Java code, no need to compile! Requires JRE/JDK installation to run.
- Can run as stand-alone server, or as integrated engine with known web servers.
- Apache web sever binary module available for i386 Linux distributions.

The Apache Software Foundation
http://www.apache.org/

JavaOne

Apache Tomcat Installation

- Install Tomcat by extracting its tarball into a new directory

```
cd /usr/local
tar zxvf jakarta-tomcat.tar.gz
```

- Run Tomcat using its startup script

```
cd /usr/local/tomcat/bin
./startup.sh
```

- Confirm Tomcat is running via Netscape

  http://localhost:8080/

The Apache Software Foundation
http://www.apache.org/

JavaOne

Apache Tomcat
Integrating with Apache server

- Install Apache's mod_jserv binary

```
mkdir /etc/httpd/libexec
mv mod_jserv.so /etc/httpd/libexec/
```

- Update /etc/httpd/conf/httpd.conf

```
Include /usr/local/tomcat/conf/tomcat-apache.conf
```

- As root, restart Apache to load mod_jserv

```
/etc/rc.d/init.d/httpd restart
```

- Confirm installation via Netscape

  http://localhost/examples/

The Apache Software Foundation
http://www.apache.org/

JavaOne

Apache Tomcat Test Program

- Create HelloWorldServlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorldServlet extends HttpServlet {
  public void service(HttpServletRequest request,
    HttpServletResponse respose) throws
    ServletException, IOException {
    response.setContentType(”text/html”);
    PrintWriter out = response.getWriter();
    out.println(”Hello, world!”);
  }
}
```

The Apache Software Foundation
http://www.apache.org/

JavaOne

Apache Tomcat Test Program (Cont.)

- Compile

```
javac HelloWorldServlet.java
```

- Put class file in Servlet deployment directory

```
cp HelloWorldServlet.class
   /usr/local/tomcat/webapps/ROOT/WEB-INF/classes
```

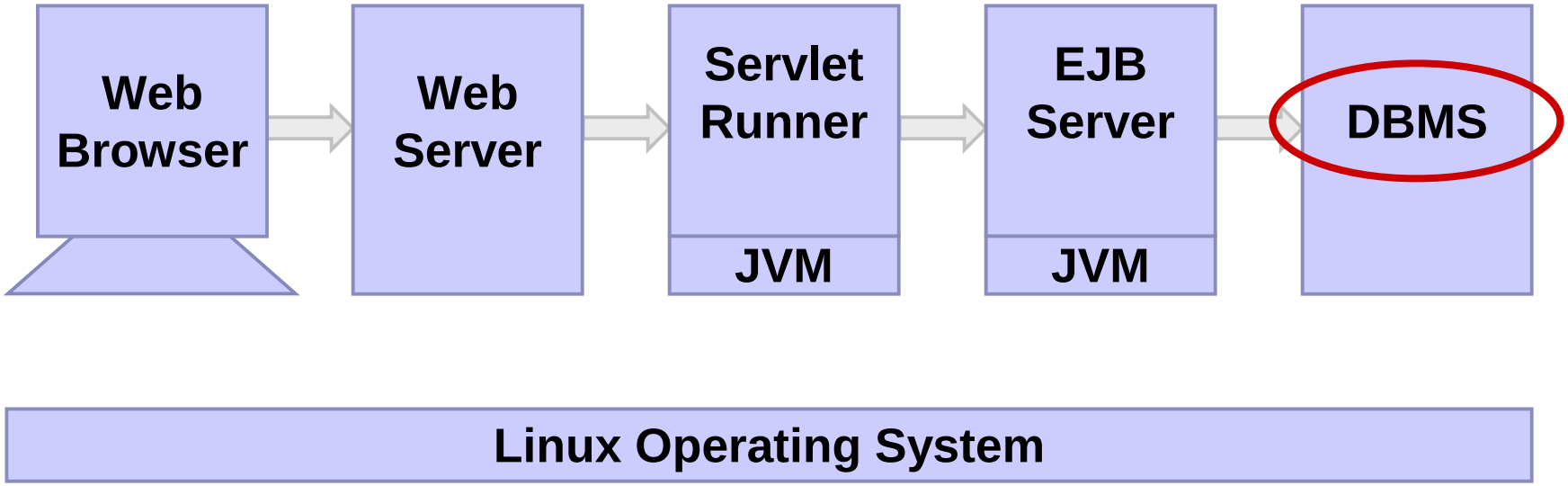
- Confirm via Netscape
  http://localhost/servlet/HelloWorldServlet

The Apache Software Foundation
http://www.apache.org/

JavaOne

The JDBC™ API

Software that enables Java technology-based programs to access databases



JDBC API Background

- Several DBMSs that support the JDBC API available for Linux
  - IBM DB2
  - MiniSQL
  - MySQL
  - Oracle
  - PostgreSQL
  - Sybase

PostgreSQL JDBC API Download

- PostgreSQL JDBC API-based driver included on RedHat 6.2 distribution, but may not have been installed
- Can be downloaded from

  http://www.postgresql.org/

PostgreSQL JDBC API Installation

- Review PostgreSQL packages installed

```
rpm -qa | grep postgres
```

- Install PostgreSQL RPM

```
rpm -i postgresql-jdbc-6.5.3-6.i386.rpm
```

PostgreSQL

JavaOne

PostgreSQL JDBC API Setting Up the Environment

- Make sure PostgreSQL is running

```
ps -f -u postgres
```

- As root, start database server if necessary

```
/etc/rc.d/init.d/postgresql start
```

- You need to set up your CLASSPATH

```
export CLASSPATH=$CLASSPATH:/usr/lib/pgsql/jdbc6.5-
  1.2.jar
```

PostgreSQL JDBC API Test Program

- Enable user to use PostgreSQL

```
su - postgres
createuser username
```

- Create a test database

```
createdb javatest
```

- Create a test table with one row in it

```
psql javatest
create table test (col1 varchar(255));
insert into test (col1) values
  ('Hello, from PostgreSQL!');
\q
```

PostgreSQL JDBC API Test Program (Cont.)

- Create HelloPostgreSQL

```
import java.sql.*;
public class HelloPostgreSQL {
  public static void main(String[] args) {
    try {
      Class.forName(”postgresql.Driver”);
      String url = ”jdbc:postgresql:javatest”;
      Connection conn = DriverManager.getConnection(
                          url, ”username”, ””);
      Statement stmt = conn.createStatement()
      ResultSet rset = stmt.executQuery(
                         ”select col1 from test”);
      // ...
```

PostgreSQL JDBC API Test Program (Cont.)

- Create HelloPostgreSQL

```
// ...
        rset.next();
        System.out.println(rset.getString(1));
      } catch (Exception e) {
        System.out.println(”Exception!”);
        e.printStackTrace();
      }
    }
}
```

PostgreSQL

JavaOne

PostgreSQL JDBC API Test Program (Cont.)

- Compile and run

```
javac HelloPostgreSQL.java
java HelloPostgreSQL
```

- Confirmation

```
Hello, from PostgreSQL!
```

PostgreSQL JDBC API More Information

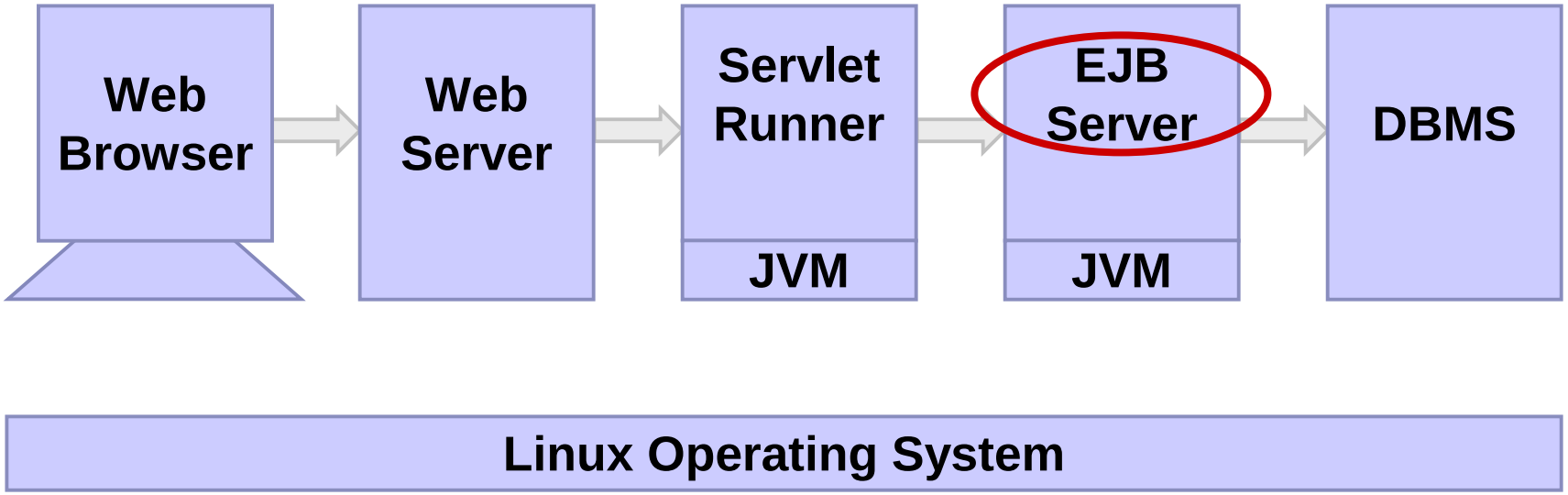
- Web

  http://www.postgresql.org/

  http://metalab.unc.edu/mdw/HOWTO/PostgreSQL-HOWTO.html

Enterprise JavaBeans™ Architecture (EJB™)

An easy-to-use Java™ technology-based application server framework



EJB Architecture Background

- Several EJB technology-based servers available for Linux
  - BEA WebLogic
  - jBoss (p.k.a. “EJBoss”)
  - JOnAS
- jBoss is an open source project about a year old that supports session and entity EJB technology-based components as well as bean and container managed persistence

jBoss Download

- jBoss 2.0
  - Uses Java™ 2 platform, JDK v1.3
  - Limited documentation
  - Support mostly in the form of mailing list
  - High potential
  - Easier installation using Java-based installed
- jBoss can be downloaded from http://www.ejboss.org/
- jBoss 1.0
  - Uses Java™ 2 platform, JDK v1.2.2
  - Must compile from source

jBoss 1.0 Installation

- Compile source

```
cd build; sh build.sh
```

- Put the run-time directory structure in the jBoss home directory

```
cd $HOME/bin; mv ejboss-1.0PR1-jdk1.2.2 $HOME
```

- Let other users see the JARs and generated proxy stubs

```
cd $HOME/ejboss-1.0PR1-jdk1.2.2
chmod 755 .. . beans beans/* beans/generated/* lib
  lib/*
```

jBoss 1.0
Setting Up the Environment

- Lots of JARs, see server.sh and client.sh as examples
- As ejboss user, start jBoss server

```
./server.sh
```

- JARs are dynamically loaded and proxy code generated and compiled
- As ejboss in another window, run the regression test client

```
./client.sh
```

jBoss Test Program

- Test program requires writing both a client and a server
- Minimum classes and interfaces
  - Home Interface
  - Remote Interface
  - EJBObject Implementation
  - Test Client
- Simple example is a stateless session bean with one simple method

jBoss Test Server Program

- Define the home interface

```
import javax.ejb.*;
import java.rmi.*;
public interface TestHome extends EJBHome {
  public Test create() throws CreateException,
    RemoteException;
}
```

- Define the remote interface

```
import javax.ejb.*;
import java.rmi.*;
public interface Test extends Remote, EJBObject {
  public String sayHello() throws RemoteException;
}
```

jBoss Test Server Program (Cont.)

- Create the bean implementation

```
import javax.ejb.*;
import java.rmi.*;
public class TestImpl implements SessionBean {

  // lifecycle methods omitted
  // session context methods omitted

  public String sayHello() throws RemoteException {
    return ”Hello, from EJBoss!”;
  }
}
```

jBoss Test Server Program (Cont.)

- Compile

```
javac *.java
```

- Create XML deployment descriptor, META-INF/ejb-jar.xml
  - Gives the EJB its name, “TestEJB”
  - Ties the interface and implementation
- JAR up the class files and the deployment descriptor

```
jar cvf EJBTest.jar *.class META-INF/ejb-jar.xml
```

jBoss
Test Server Program Deployment

- Stop the server (server.sh)
- Copy the JAR file to the jBoss beans deployment directory

```
cp EJBTest.jar ~ejboss/ejboss-1.0PR1-jdk1.2.2/beans
```

- As ejboss, restart the jBoss server

```
sh server.sh
```

- Check standard output log
- Likely errors related to CLASSPATH and deployment descriptor file

jBoss Test Client Program

- Steps of the test client are to:
  - Set up properties for resolving an initial context
  - Resolve the initial context
  - Get the home interface (a factory)
  - Create the EJB interface using the home interface
  - Invoke the bean methods

jBoss Test Client Program (Cont.)

- Create TestClient

```
import java.util.*;
import javax.naming.*;
public class TestClient {
  public static void main(String[] args) {
    try {
      Properties properties = new Properties();
      properties.put(Context.INITIAL_CONTEXT_FACTORY,
        ”org.jnp.interfaces.NamingContextFactory”);
      properties.put(Context.URL_PKG_PREFIXES,
        ”org.jnp.interfaces”);
      properties.put(Context.PROVIDER_URL,
        ”127.0.0.1”);
      //...
```

jBoss Test Client Program (Cont.)

- Create TestClient (cont.)

```
        //...
        Context context = new
          InitialContext(properties);
        TestHome testHome
          (TestHome)context.lookup("TestEJB");
        Test test = testHome.create();
        System.out.println(test.sayHello());
      } catch (Exception e) {
        System.out.println(e);
      }
    }
  }
```

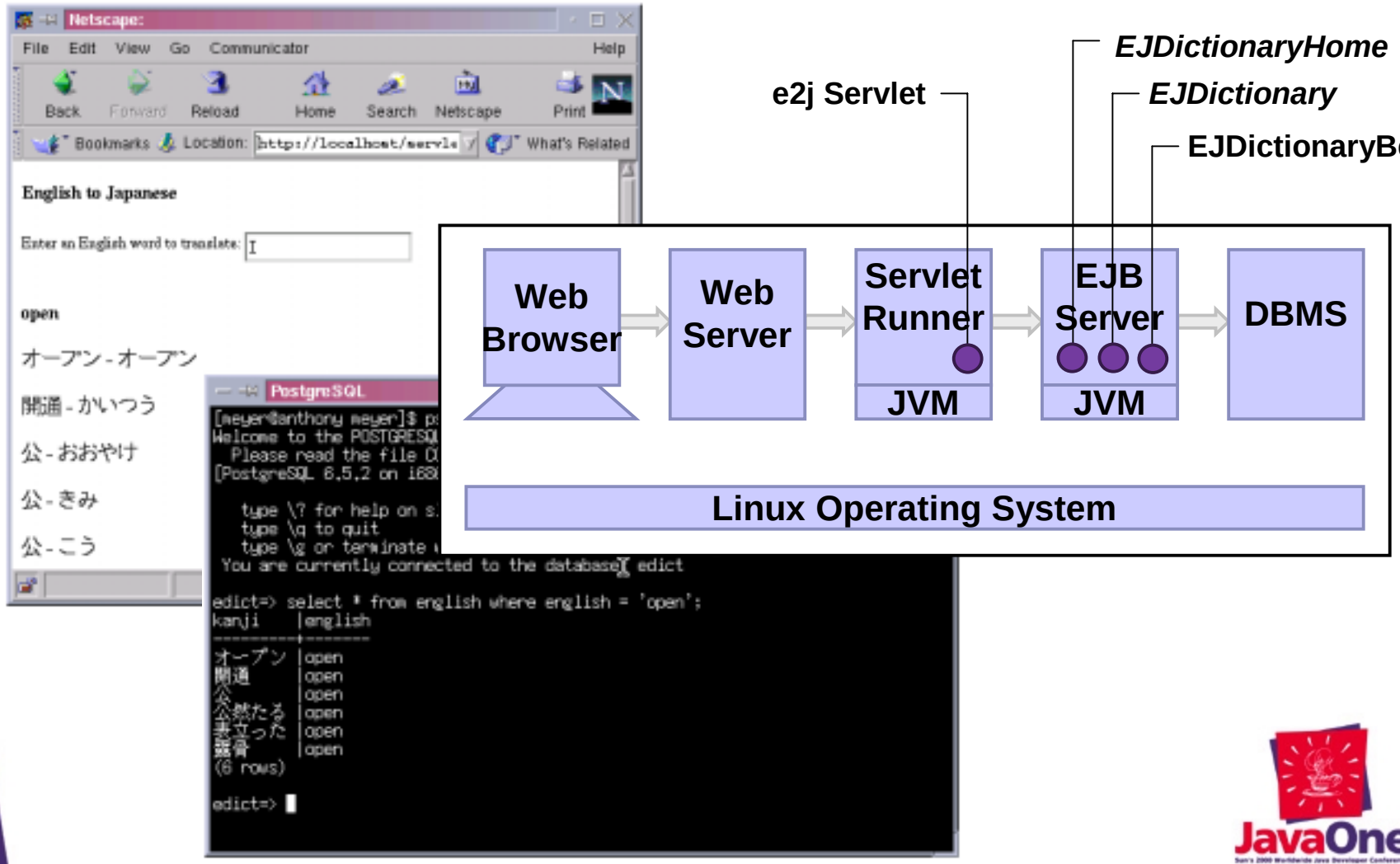
jBoss Test Client Program (Cont.)

- In a separate window from server.sh, compile and run

```
javac TestClient.java
java TestClient
```

- Review server.sh standard output for errors

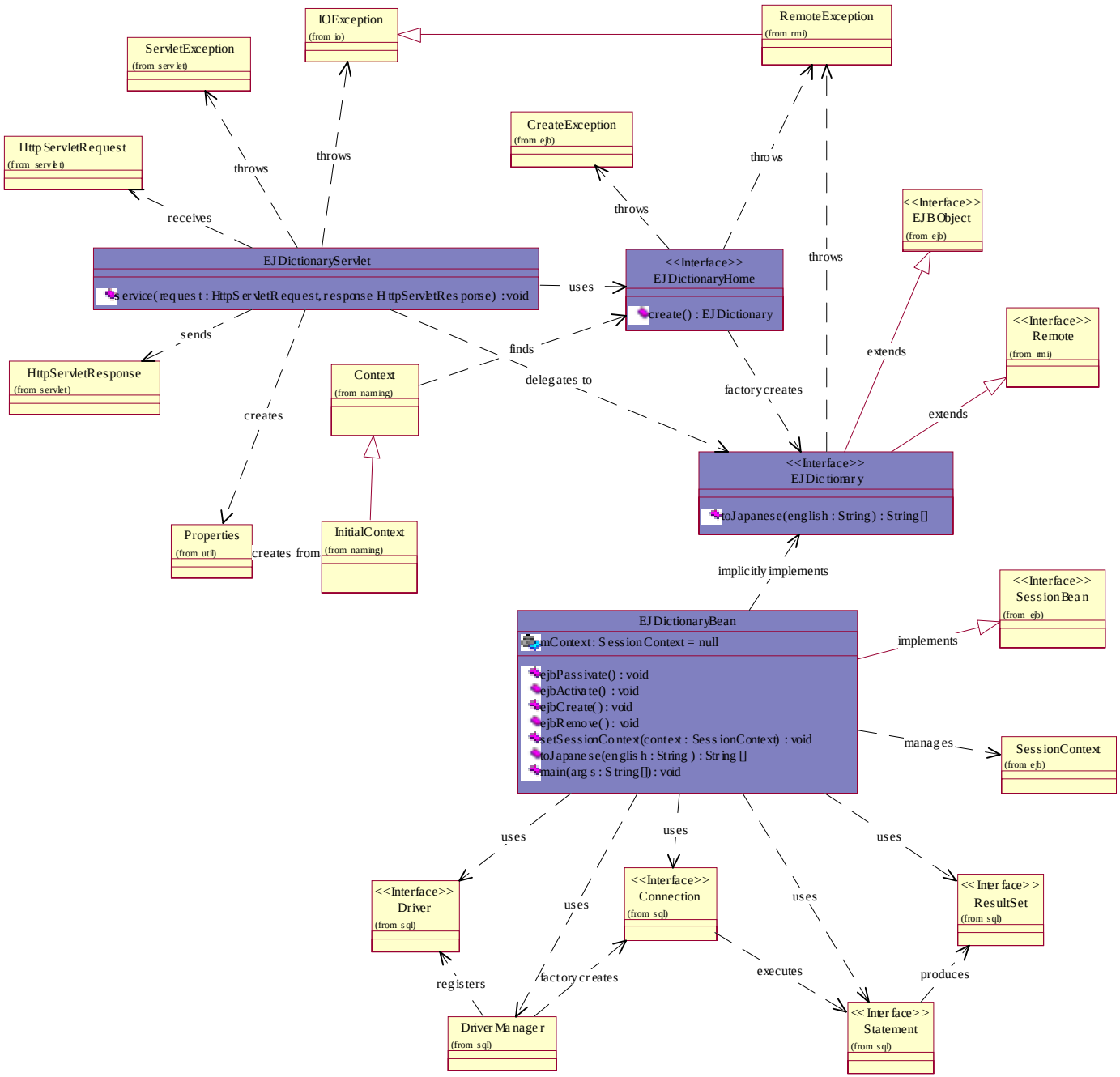
End-to-End Example

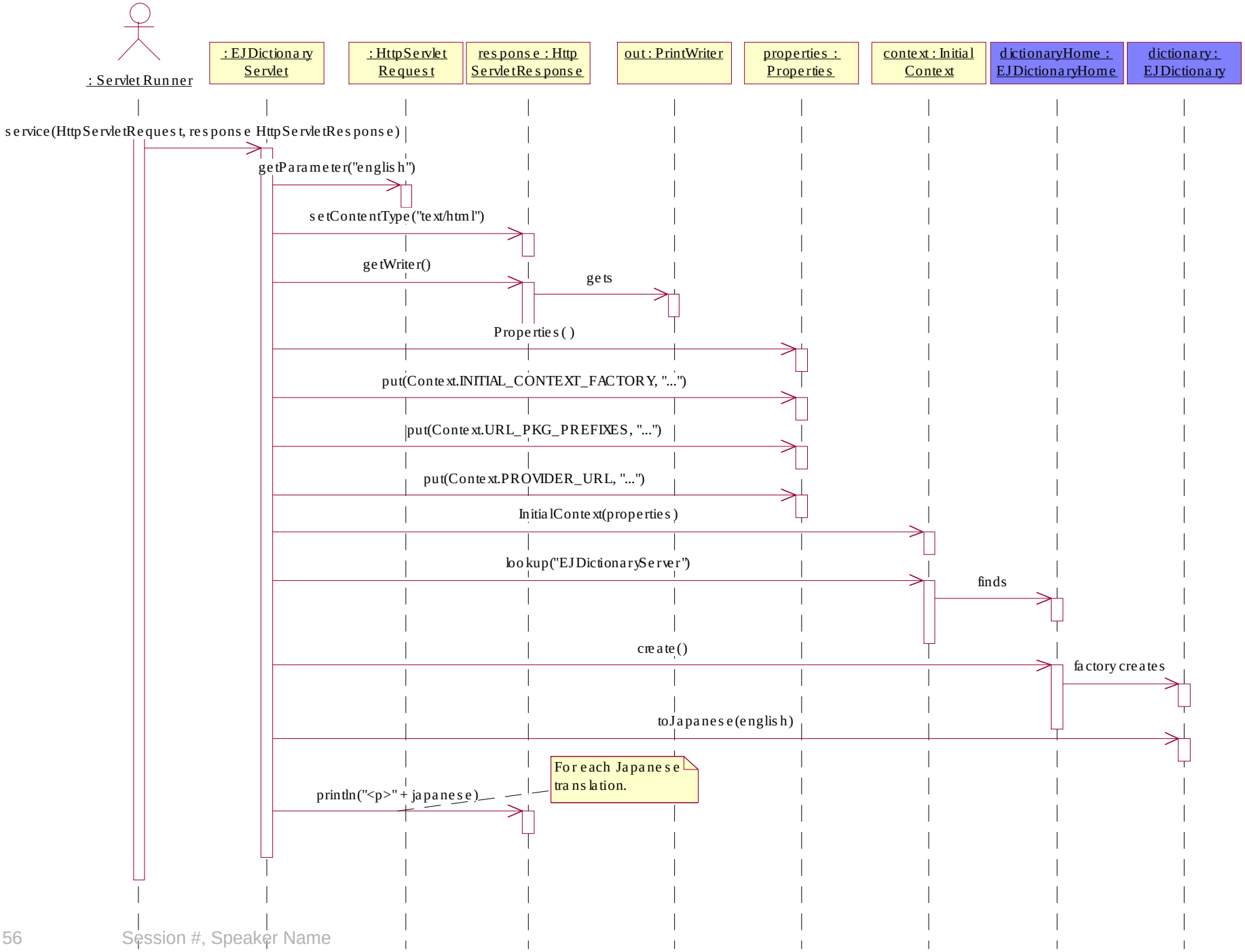


End-to-End Example Classes and Interfaces

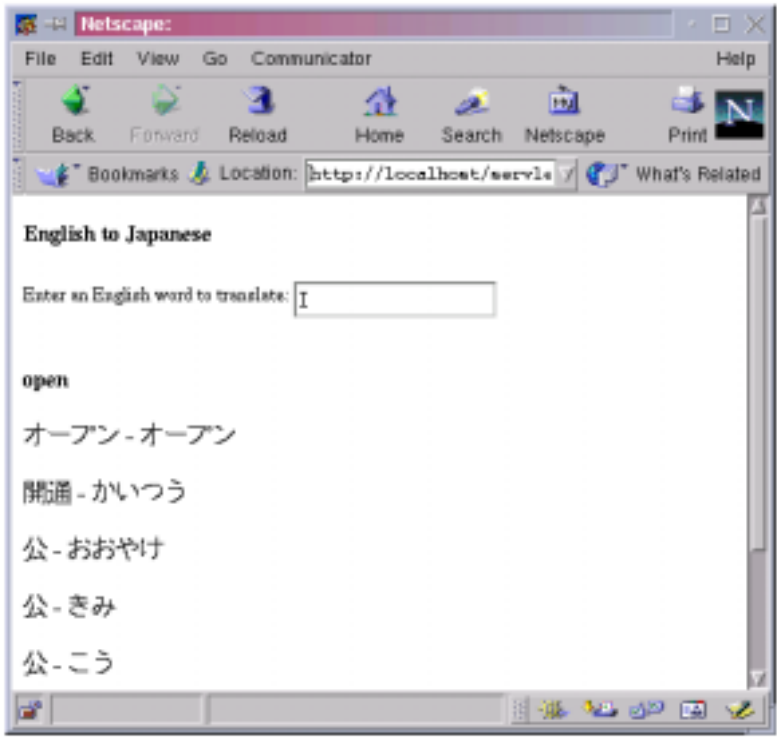
- e2j Servlet Class
  - Receives the HTTP POST
  - Delegates to the EJDictionaryBean
- EJDictionaryHome Interface
  - EJB factory interface with a simple create()
- EJDictionary Interface
  - Declares String[] toJapanese(String english);
- EJDictionaryBean Class
  - The EJBObject implementation
  - Does a SQL query to translate English to Japanese

Here “EJ” means “English to Japanese” not “Enterprise Java”


IOException
(from io)
RemoteException
(from rmi)
ServletException
(from servlet)
HttpServletRequest
(from servlet)
CreateException
(from ejb)
<<Interface>>
EJBObject
(from ejb)
throws
receives
EJDictionaryServlet
service(request : HttpServletRequest, response HttpServletResponse) : void
uses
<<Interface>>
EJDictionaryHome
create() : EJDictionary
sends
finds
extends
<<Interface>>
Remote
(from rmi)
HttpServletResponse
(from servlet)
Context
(from naming)
delegates to
factory creates
creates
<<Interface>>
EJDictionary
toJapanese(english : String) : String[]
Properties
(from util)
creates from
InitialContext
(from naming)
implicitly implements
<<Interface>>
SessionBean
(from ejb)
EJDictionaryBean
mContext: SessionContext = null
ejbPassivate() : void
ejbActivate() : void
ejbCreate() : void
ejbRemove() : void
setSessionContext(context : SessionContext) : void
toJapanese(english : String) : String []
main(args : String[]) : void
implements
manages
SessionContext
(from ejb)
uses
<<Interface>>
Driver
(from sql)
<<Interface>>
Connection
(from sql)
<<Interface>>
ResultSet
(from sql)
registers
factory creates
executes
produces
DriverManager
(from sql)
<<Interface>>
Statement
(from sql)


: Servlet Runner
: EJDictionary Servlet
: HttpServlet Request
response : Http ServletResponse
out : PrintWriter
properties : Properties
context : Initial Context
dictionaryHome : EJDictionaryHome
dictionary : EJDictionary
service(HttpServletRequest, response HttpServletResponse)
getParameter("english")
setContentType("text/html")
getWriter()
gets
Properties ( )
put(Context.INITIAL_CONTEXT_FACTORY, "...")
put(Context.URL_PKG_PREFIXES, "...")
put(Context.PROVIDER_URL, "...")
InitialContext(properties)
lookup("EJDictionaryServer")
finds
create()
factory creates
toJapanese(english)
For each Japanese translation.
println("<p>" + japanese)

Demonstration



$0 This presentation was done with 100% free and mostly open source software

- RedHat Linux 6.2 including:
  - Apache 1.3.9
  - Gnome 1.0
  - Netscape™ 4.72
  - PostgreSQL 6.5
- J2SE™ 1.2.2 platform
- Apache Tomcat 3.1
- jBoss (“EJBoss”) 1.0PRE1
- JBuilder Foundation 3.5
- Sun StarOffice™ 5.2 software
- EDICT Japanese/English Dictionary, 19 MAR 2000

Conclusions

- Java™ technology on Linux is really just getting started
- The Java technology/Linux landscape is rapidly changing
- Numerous popular commercial products are available
- Open source products also provide an alternative with a low entry cost
- Linux offers a viable alternative for Java technology-based development for the enterprise

More Information

http://JavaWiz.com/

- This presentation's slides, sample code and configuration files

http://gary.meyer.net/

- Enterprise Java™ Technology for Linux HOWTO

http://java.sun.com/linux/


JavaOne
SM
Sun's 2000 Worldwide Java Developer Conference