

From 4GL to JEE Taking a Bold Step

Gary Ben-Israel
Company Name

Session C16
Wednesday, April 30, 2008 • 2:10 p.m. – 3:10 p.m.

2008 IIUG Informix Conference



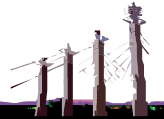
Agenda

- Considerations
- Alternatives
- The Building Bricks
- Performance
- Batch operations and reporting
- Securing your application
- Real life example



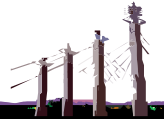
Considerations

- General considerations
- UI considerations
- Maintenance & Future development



General considerations

- Client Server 
- Separating your business logic
- Adding new functionality
- Connection pooling
 - Temporary tables
 - Database level security
- Performance



In a typical web environment the application runs on a client connected to an application server that connects to the database server.

Normal 4GL programs run directly on the database server.

In 4GL there is no separation between the UI, the SQL and the business logic.

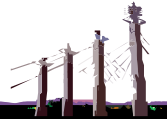
In JEE they are separate. This will imply a design change which is much dipper the just changing the UI.

Using connection pooling means that the same connection to the database is not closed when handed back by the application but returned to the pool and handed to the next application requesting a connection. As temporary tables and database security are attached to a connection you cannot rely on them

Adding TCP/IP communication and working in a three tire model adds new performance issues.

UI considerations

- Taking advantage of the WEB UI capabilities
- Client side and Server side Validations
- Changing the work flow
- Background tasks
- Graphic design
- Supporting different browsers



Moving from a character based to a graphical system introduces new capabilities such as displaying images.

Validations can take place at the browser level the application server level or both. Client side validations are faster but less secure and written in JavaScript which is not as powerful as Java. Server side validations are slower and impact the server. The solution is to use both.

Working with a browser requires a change in screen navigation, page navigation and operations.

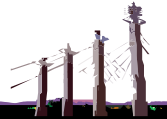
Background tasks are detached. They are unaware of the existence of the browser hence closing the browser will not affect them.

Graphic design is highly important but is time and cost consuming.

JavaScript and even HTML differs between browsers and browser versions.

Maintenance & Future development

- Tools
 - IDE – Integrated Development Environment
 - Version control
 - Build manager
- Hiring trained professionals
- The next step

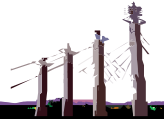


Before embarking on your JEE adventure you need to set up your environment. This will have a big impact on speed and quality.

The open source offers a huge diversity. Before selecting your products make sure that there are enough trained professionals.

Alternatives

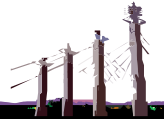
- Tools that work on your 4GL code
 - 4J's Genero
- Automatic conversion to a different language
 - EGL
- Developing from scratch



Using your existing code or an automatic migration is tempting but will not take advantage of the new possibilities.

The Building Bricks

- UI Solutions
- Frameworks
- The EJB container

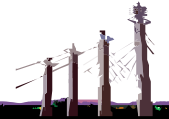


UI Solutions

- Rich Client (Desktop Applications)
- Thin Client (Classic web Applications)

Both address similar missions:

- Event handling
- Data Binding
- Navigation flow
- Serialization
- Layout
- Input validation
- I18N (Internationalization)
- L10N (Localization)



Rich Client Vs. Thin Client

☺ **Asynchronous**

- Uninterrupted user interaction

☺ **Client-side logic may be implemented in Java**

☹ **Requires Installation**

- Multiple client deployments

☹ **Synchronous**

- UI reloaded and rendered with each request

☹ **Client-side logic may be implemented in JavaScript**

☺ **Runs anywhere**

- With a browser

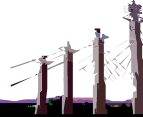


AJAX Basics

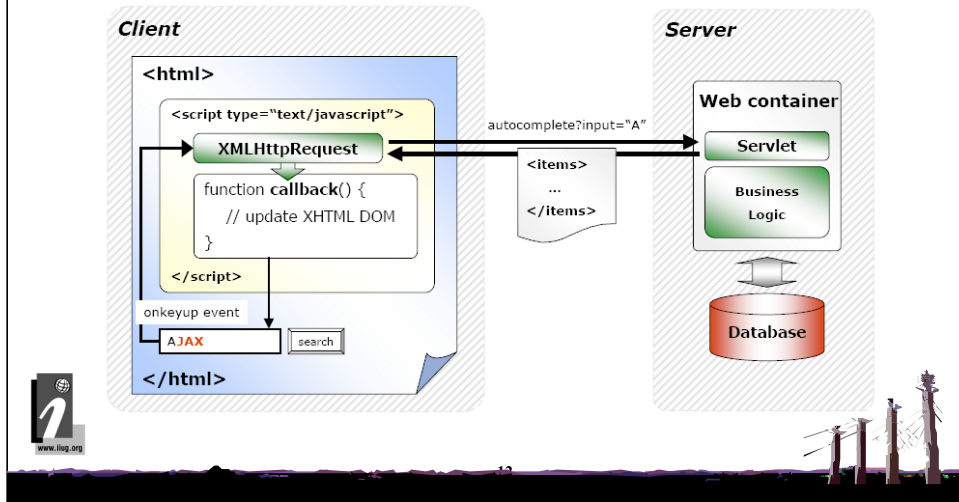
Asynchronous Javascript + XML(HttpRequest)

Not a single technology but a methodology

- Built on two standards:
 - XMLHttpRequest
 - DOM (XHTML)
- Relies on Javascript availability
- It's mostly great **but**.. May increase server load



AJAX: to the server and back again



The most noticeable advantage of AJAX is that only a part of the page is rendered. behind the scenes it offers asynchronous communication between the browser and the application server.

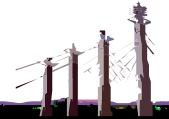
The framework universe is vastly hugely mind-bogglingly big

Wicket	WebObjects	Chrysalis	OpenEmcee	FacesFreeway
Struts	JPublish	(OpenLaszlo)	Scope	JSPWidget
JSF	Stripes	VRaptor	Warfare	Caramba
WebWork	ThinkCap JX	Swinglets	JWAA	Xopolon
Spring MVC	OXF	wingWeb	Helma	
Tapestry	Canyamo	Verge	Jacuard	
Cocoon	Anvil	XUI	Macaw	
Turbine	Jaffa	Japple	JOSSO	
Expresso	Millstone	ZK	XAMJ	
Maverick	Jucas	Bishop	Smile	
Echo2	(FreeMarker)	SwingWeb	Chiba	
Sofia	(Velocity)	Weblets	JBanana	
JATO	Common Controls	Baracuda	Jenius	
Rialto	Folium	Action Framework	JWarp	
Dinamica	Click	Shocks	Genie	
EchoPoint	JAT	TeaServlet	Avatar	
SWF	MyFaces	wingS	Dovetail	
Hijax	Facelets	Bento	Cameleon	
OpenXava	WebOnSwing	jStateMachine	JFormulator	
RIFE	DWR	jZonic	Melati	
Trails	Thinlet	qooXdoo	Struts Ti	

And
many
More...

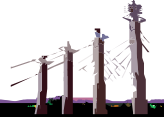
Why so many?

- Different needs
- A bad case of NIH syndrome
- It's fun writing
- The de-facto standard isn't good enough
- No framework is good enough (we got to have more!)



Framework categories

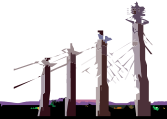
- Action-Based frameworks
 - Struts, WebWork (Struts 2), Spring-MVC
- Component oriented frameworks
 - Wicket, Tapestry, Facelets (JSF), Echo2, ZK, Zimbra, Millstone
- Template Engines integration
 - FreeMarker, Velocity, JSP, Tiles, Tapestry, Facelets (JSF)
- Full Stack frameworks
 - RIFE
- CRUD applications frameworks (Influenced by Ruby on Rails)
 - Trails, FacesFreeway, Struts Ti, RIFE-CRUD



Some frameworks cross categories.

The EJB Container

- Entity beans
- Session beans
- Message Driven Beans
- Resource Management and Primary Services



Entity Beans

Entity beans in Java Persistence 1.0 specification are available only as plain old java objects (POJOs). To implement an entity bean you need to define a bean class and decide what field you will use as the identifier (primary key) of that class.

In Java persistence, entity beans are not components like their EJB 2.1 counterparts. Application code works directly with the bean.

So how is an entity queried? How is it stored? Is some magic involved?

NO. Interaction with entity beans is done via a new service called the EntityManager.



No Magic. No byte code manipulation. No special proxies.

Just Plain Java.



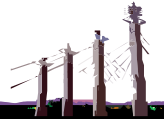
Unlike other EJB types entities can be allocated, serialized and sent across the network like any other POJO.

Entity beans model real world objects. These objects are usually persistent records in some kind of database.

In Java persistence application code works directly with the entity bean and does not interact through a component interface as you would with an EJB Session bean.

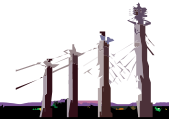
Entity bean example

```
import javax.persistence;  
@Entity  
@Table(name="CABIN")  
public class Cabin {  
    private int id;  
    private String name;  
    @Id  
    @GeneratedValue  
    @Column(name="CABIN_ID")  
    public int getId() { return id; }  
    public void setId(int pk) { this.id = pk; }  
    @Column(name="CABIN_NAME")  
    public String getName() { return name; }  
    public void setName(String str) { this.name = str; }  
}
```



Object Relational Mapping (ORM) Solving the Paradigm mismatch

- The problem of granularity
- The problem of subtypes
- The problem of Identity
- Problems related to associations
- Problem of object graph navigation



The problem of granularity

Granularity refers to the relative size of the objects you're working with.

As an example let's look at the address fields in a RDBMS table user. In Java we will have an address class. We can use UDTs but they are poorly supported by SQL and not portable between different databases.

The standard solution will be to map the address class to:

```
create table user(
  name varchar(50),
  address_street varchar(50),
  address_city varchar(15),
  address_state varchar(15),
  address_zipcode varchar(15)
)
```

The problem of subtypes

In Java we implement inheritance using a super and sub class.

In an SQL database we can't declare that credit_card_details

table is a subtype of billing_details by:

```
create table credit_card_details extends billing_details.
```

Hence it is not surprising that SQL will not support polymorphism

The problem of Identity

Java objects define two different notations of sameness:

Object identity (Checked with a==b)

Equality as determined by the implementation of the equals() method.

On the other hand identity of a database row is expressed as the primary key value.

Neither equals() nor == is naturally equivalent to the primary key value.

Problems related to associations

Object-oriented languages represent association using object references and collections of object references.

In the relational world, an association is represented by a foreign key column.

Object references are inherently directional, from one object to another. If you need both directions you must define the association twice. Once in each class.

Java associations can be many to many. If you have a many to many relationship in a relational database using a link table, this table doesn't appear anywhere in the object model.

Problem of object graph navigation

In Java when you access the billing information of a user you call a User.getBillingDetails().getAccountNumber()

This is often described as walking the object graph.

Unfortunately this is not an efficient way to retrieve data from an SQL database.

In the database you would want to use a join to minimize the number of SQL queries.

This mismatch in the way we access objects in Java and in a relational database is perhaps the single most common source of **performance problems** in Java applications.

Why ORM

- Productivity

Persistence-related code can be the most tedious code in a java application. The ORM eliminates much of the grunt work and lets you concentrate on the business problem.

- Maintainability

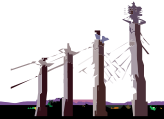
Fewer lines of code makes the system more understandable and easier to re-factor.

ORM provides a buffer between the two models, allowing more elegant use of object orientation on the Java side, and isolating each model from minor changes to the other.

- Performance



- Vendor independence



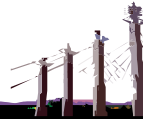
Since O/R mapping is mandated by the specification it makes EJB applications much more portable between vendors

A none-benefit

A supposed advantage of ORM is that it “shields” developers from “messy” SQL. This view holds that object-oriented developers can’t be expected to understand SQL and find it somehow offensive.

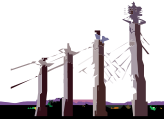
On the contrary, Java developers must have a sufficient level of familiarity with, and appreciation of, SQL in order to work with ORM.

To use an ORM effectively, you must be able to view and interpret the SQL statements it issues and understand the implications for performance.



Performance

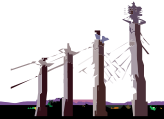
- Client side validations
- Caching highly used tables
- Optimizing message traffic
- Using “first” and “skip” in the select statement
- Using Stored procedures and UDRs
- Running ESQL C or Java programs in the server



There is common belief that Java performs poorly. This is not true. Written correctly and tuned properly you can expect the same performance as 4GL.

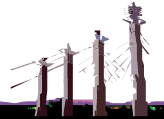
Batch operations and reporting

- Using a batch framework
- Communicating with batch programs
 - Maintaining connection after ending the client
 - Aborting the job
 - Viewing the job status and progress
 - Error reporting
- Report generators
- Report output formats



Securing your Application

- Authentication
- Authorization
- The Security Framework
- Database level security



The typical user will not have a Unix login so the need to be authenticated by the application.

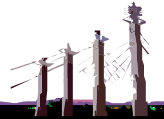
There are issues trying to use Informix encryption in Java.

In a graphical UI You need to consider component level security in addition to page and service.

With connection pooling you cannot use database level security because the same connection is shared between different users.

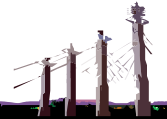
Real life example

- NITE is in the process of migrating 3 million lines of 4GL to Java EJB3
- We started October 2006 and are planning to finish in Q2 2010.
- Our first module shipped in December 2007.
- We are developing in house. Our development team is composed of 7 NITE and AlphaCSP Java programmers.

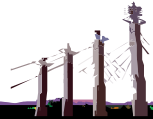
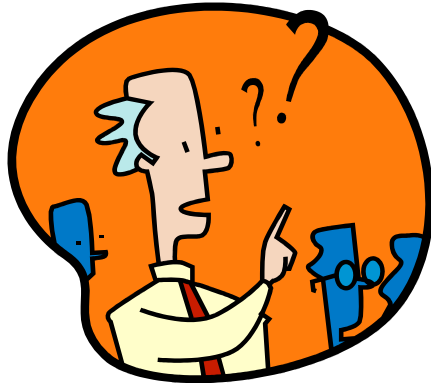


What did we choose

- IDE – IntelliJ
- Version control – SVN
- Build manager – Maven
- UI Framework – Wicket
- Job Framework Quartz
- Security Framework Acegi
- Application Server – Jboss
- Persistence - Hibernate
- Database - **Informix**



Questions



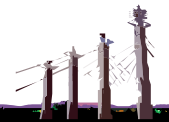
Session C16

From 4GL to JEE Taking a Bold Step

Gary Ben-Israel

National Institute for Testing & Evaluation

gary@nite.org.il



JEE three tier model

