ercato J

cut J2EE project **complexity**

In your J2EE project,

- *development takes 42 percent longer than the worst estimate?*
- *progress has slowed down?*
- *hours of delay between coding and testing?*
- *builds are a nightmare?*
- *multiday transition to live systems?*
- *your architecture becomes obfuscated?*
- *business logic moves into JSPs?*

cut the Gordian knot

Living Pages Research

to **solve** and to **innovate**

At a glance: how ercatoJ can help

Key benefits:

. . for your project

- ✓ edit-compile-test in seconds
- ✓ hot deployment in production
- ✓ separation of concerns
- ✓ lines of code reduced
- ✓ iterative approach is encouraged
- ✓ quick response to requests for change

. . for your development work

- ✓ fewer J2EE pitfalls
- ✓ faster or no EJB deployment
- ✓ less or no SQL required
- ✓ no XML schema required
- ✓ focus on business processes

. . for your product

- ✓ possibly richer functionality
- ✓ configurable and safe queries
- ✓ modifications possible with no interruption of service

ercatoJ can reduce project complexity because it is built upon an amazingly simple yet powerful concept: ercatons.

Every ercaton encapsulates a business object or process and absorbs complexity.

ercatons live within the Java2 EnterpriseEdition (J2EE) container and are built from Java objects and XML documents. They are able to implement most patterns in enterprise software development.

Their integration into J2EE is based on a few Enterprise JavaBeans (EJBs) which need no redeployment when software changes. Users, transactions, sessions and the database are fully shared. ercatons therefore do not introduce new complexity into an existing J2EE application.

Every ercaton encapsulates a business object or process

The following analogy may help to understand: "*if J2EE is an operating system, then ercatons add to it what the many small /usr/bin binaries add to Unix: power and simplicity*".

This analogy holds true to the point where ercatons are not yet another framework to build this big, monolithic application – they offer an option to build a simpler one.

J2EE problems solved

J2EE is the best standardized application server architecture available. Still it has problems which can be addressed using ercatoJ:

The J2EE blueprints assume that all relevant details of the business model (as expressed at an UML level) are implemented as EJBs and that retrievable business data maps to persistent attributes.

If J2EE is an operating system, then ercatons add to it what the many small /usr/bin binaries add to Unix: power and simplicity

This assumption is fine for a project following the waterfall model: every detail is designed first, then implemented.

However, many successful projects follow the model of the Agile Process: start with a simple system and iterate until the project goal is reached. Such projects are left with two alternatives:

1. Frequently refactor the entire J2EE model
2. Let the J2EE model contain "generic parts"

The first leads to overly complex projects because even a simple change is a tedious and error-prone procedure. Some changes are close to impossible after the application went productive.

The second alternative leads to projects where most of the time is spent to create or use a technical framework to handle the genericity: the initial business model moves out of focus.

Both, J2EE frameworks and J2EE-aware IDEs only push the limit a bit without eliminating it.

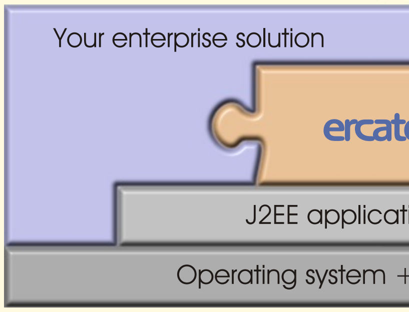
Therefore, in the typical large-scale J2EE project, most of the time is spent "to integrate, redeploy and address technical issues".

ercatoJ offers a way out by proposing a third option:

3. Keep the J2EE model small and focused.

XML problems solved

XML has grown very poular and the marketplace proposes several XML-based application servers or XML databases.



It is probably true that corporate data is best stored in the form of XML. However, there are four problems which can be addressed using ercatoJ:

1. Retrieval of XML is slow and not standardized
2. XML data is often restricted to a schema
3. Unlike J2EE, XML application servers are proprietary
4. Object-oriented modelling is not supported (only mapped)

Every er introduc com

ercatoJ offers an alternative because many of the advantages of XML are made available within a J2EE application.

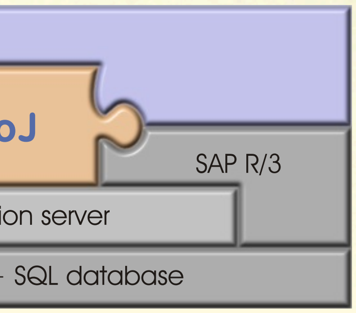
ercatons boldly go w

Usage examples

The ercato programming model encourages an iterative development style. Focus on one problem at a time. When one aspect is finished it is easily integrated into a changing environment. Examples?

Assume you want to **add business methods, data fields or constraints**. Then add some lines of XML to an ercaton and possibly add some Java code, too. Now just copying the changed ercaton and possibly the changed jar archive will do the job. It is that easy. The J2EE application needs no redeployment and no interruption of service is going to occurr (the copy operation is transaction-safe).

Assume you **require another column** (or table) **in your database scheme**, e.g., for a query operation. Then you only need to add an XML-tag to one (index-)ercaton and one XML-attribute to one (base-)ercaton. That is it. The content of the new column is now even going to reflect the production data which existed before and no interruption of service is going to occur.

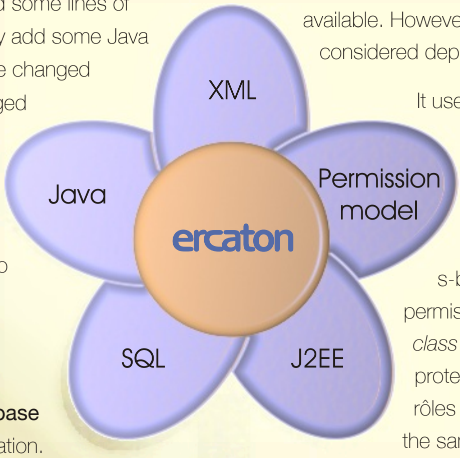


Assume you **frequently change the business logic** and do **not want to update the user interface**. Then alternatively you may create a style sheet or customize the ercatoJ standard web style sheet. You do this once forever. Possibly combined with your JSPs you end up with a user interface which stays in sync with the business logic and provides support for viewing, editing, navigation and complex search.

rcaton you
e absorbs
plexity

ercatoJ anatomy

ercatoJ uses bright ideas from several ancestors.



It uses XML to represent and persist rich structures, XSLT may express business logic and views.

It uses SQL as an accelerator for unsacrificed performance. Even inner and outer joins are available. However, SQL as a language is considered deprecated.

It uses and complements J2EE, e.g., transactions, sessions, authentication.

It uses an extended Unix file system semantics (incl. x- and s-bits) to express rôle-based permissions *by instance*, not *by class*. As a consequence, protection ensures that two different rôles may see different data from the same SQL query.

It uses Java and its object-oriented model. Business logic may be expressed in arbitrary Java.

It uses the Unix philosophy of many small building blocks which cooperate towards a whole which is more than the sum of its parts. ercatons are the parts, the J2EE application is the whole. The concept of several users conributing parts in a secure manner is implemented, too.

Keep the J2EE model small and focused

It uses the concept of prototype-based languages such as '*Self*' where objects are created without a class.

"Ercatons were easy to use and breathtakingly efficient. Once you get the idea you wonder how you ever worked without it."

Dr. Ralf Marsula
Clavis GmbH, Bremen Solution Partner SAP

How it all works

ercatons are the result of the ercato project to create a server environment which does not just *export* web services but is actually fully programmable *using* web services. In this context, many shortcomings of web services have been addressed and a decent security model has been created, to start with.

In all traditional J2EE frameworks and J2EE itself, XML plays a helper role, as carrier of data from model to view, or from one application to another, or as a configuration language. In contrast, ercatoJ treats XML as what it is: a first class citizen. It is the first and only J2EE-based product which does so: XML documents contain and store all the business data *plus* as much business logic as is required.

Every XML
is an o

But why is the ercato programming model that powerful? Because it exactly starts from the simple idea we mentioned above: **treat *every* XML document as *being* an object** (and *call* it an ercaton). Inheritance, polymorphism, encapsulation, persistence, methods, appearance etc. may all be expressed at the declarative level of XML. This includes some properties of real-world objects which software objects usually do not posses. Additionally, the full procedural power of Java and J2EE is made available as well.

Did you ever wonder why you should care about the distinction between object and class when creating an UML business model? ercatons make this distinction obsolete and in many cases, an ercaton just is *the* perfect representation of a business object or process. On the other hand, an ercaton looks as much as a Java object to Java code as you want it. There is nothing to sacrifice.

Consider the following scenario: "In some application, customer data is expressed as an XML document with actually no Java involved. At the same time, a customer's account is expressed as an entity bean delegating to an account class. Both, customer and account have been modelled as

here no ejb has gone

objects (ercaton and Java class)." We anticipate and fully support this situation. Of course, an SQL database will be involved, too. For maximum freedom, ercatoJ does not restrict ercatons to any XML schema (or XML at all...) and any Java class is able to interoperate. It is not very elegant but perfectly legal for some Java code to manipulate an ercaton at the XML-DOM level.

ercatons represent business data *and* business logic at the XML level. This may have introduced problems at the levels of orthogonality and redundancy, change management, or consistency. Yes, and ercatoJ solves all of them.

document
bject

It is impossible in this booklet to discuss the background of prototype-based languages or XML node-set algebra which are part of the scientific foundation of the ercato programming model. Rather, we just claim the following here: "*Once an example of a business object or a description of a business process is written down (in XML, maybe using a text editor), the implementation of this object or process as part of a J2EE application is already complete or very close to completion*". You probably need to see to believe.

Available ercatoJ modules:

- ✓ ercatoJ Base
- ✓ ercatoJ EJB support API
- ✓ ercatoJ TagLib (*)
- ✓ ercatoJ User manager
- ✓ ercatoX One standard extensions, level 1
- ✓ ercato Persistence layer
- ✓ ercato Versions advanced versioning
- ✓ ercato WebServices
- ✓ ercato InheritanceForXML
- ✓ ercato Query service
- ✓ ercato SecureRôles
- ✓ ercato WebView customizable web style
- ✓ ercato WebXP enhanced web experience
- ✓ ercato Edit+Forms
- ✓ ercato Backup+Replication
- ✓ ercato Shell remote login and scripting
- ✓ ercato WebDAV (*)

(*) in v1.2

(ercatoJ and ercato.NET modules share ercato and ercatoX modules)

ask for product details

before

ercatoJ Fact Sheet

J2EE integration

lightweight integration stub
(4 EJBs, 3 servlets)
shares users, transaction and
session contexts
standalone for EJB-free projects is available
code archives (JARs) not required in EAR
callable by / may call other EJBs

Deployment

deployment without EJB re-deployment
deployment and testing are scriptable
sub-deployments are supported

Component-based technology

business logic encapsulated
in small components
pool of useful components provided
extensible even in running application
stand-alone components (ercatons)
each component may define a web-service

Programming models

by Java API
by manipulation / transformation of XML
object-oriented at *both*, Java *and* XML level
declaration of actions / constraints / triggers
3rd-party XML mapping frameworks work

User interface

optional, customizable web interface
pre-configured edit cycle provided

Workflow

follow-up actions are supported

Security control

per action / method and instance
protected data not exposed by queries

Persistence technology

true XML is persisted, not just mapped
no schema required
transaction-isolation level
accessible to text and XML tools

Database

SQL used for full retrieval performance
database schema follows data on the fly
combines with existing tables
JDBC/JDO is enabled but not required

Requirements

JDK 1.4
Oracle 9iAS or Orion 2.0
IBM Websphere 5.0 (*)
BEA Weblogic 8.1 (*)
JBoss 3.2 (*)
Oracle 9i
MySQL MaxDB 7.4 (*)

Availability

v1.0: now
v1.1: 4th quarter 2003
by Living Pages Research GmbH,
Munich, Germany

(*): supported, yet to be certified for production use

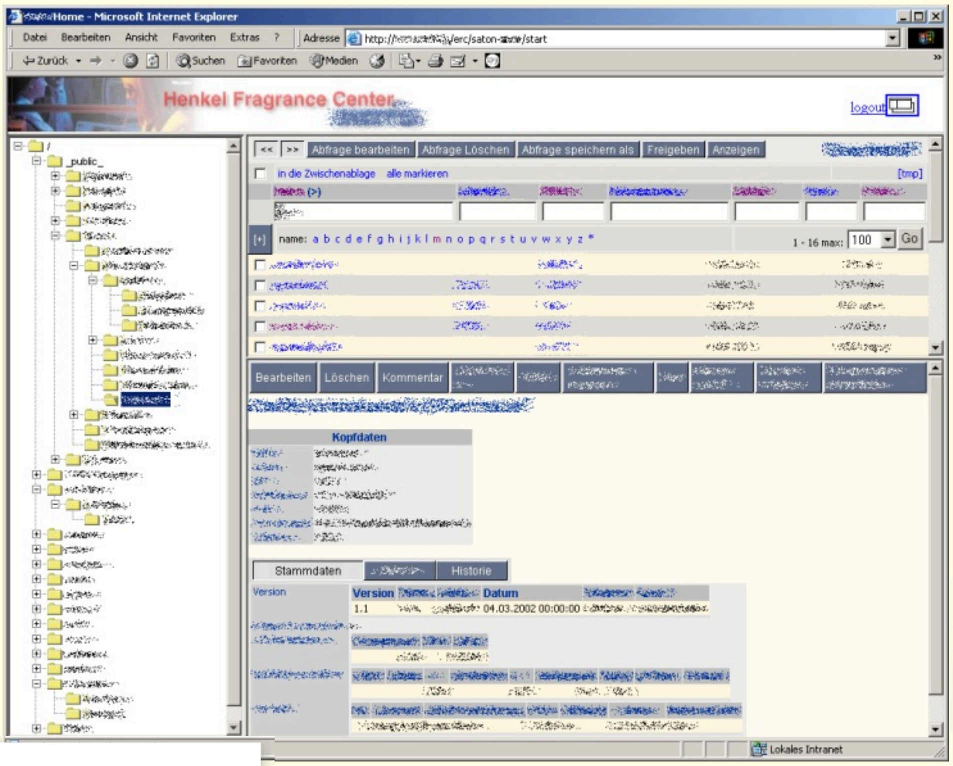
ercatoJ

ercatoJ at Henkel Group: a selected customer reference

The Henkel Group (Henkel KGaA, Düsseldorf, Germany) operates in three strategic business areas - Home Care, Personal Care, and Adhesives, Sealants and Surface Treatment. The Company is represented in over 75 countries. 48,515 employees work for the Henkel Group worldwide.

Henkel Fragrance Center GmbH (HFC) develops and produces perfume oils which are indispensable components for many products of the group.

Living Pages has developed and deployed the new central software system at HFC. This mission-critical enterprise solution supports every single business process of perfume development and is fully productive.


Home - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Adresse http://…/erc/saton-…/start
Zurück Suchen Favoriten Medien
Henkel Fragrance Center
logout
public
Abfrage bearbeiten Abfrage Löschen Abfrage speichern als Freigeben Anzeigen
in die Zwischenablage alle markieren
[tmp]
name: a b c d e f g h i j k l m n o p q r s t u v w x y z *
1 - 16 max: 100 Go
Bearbeiten Löschen Kommentar
Kopfdaten
Stammdaten Historie
Version Datum
1.1 04.03.2002 00:00:00
Lokales Intranet

“Ungeachtet der komplexen Materie wurde das Problem hervorragend gelöst sowie Zeitrahmen und Budget eingehalten. Bei der Lösung unseres geschäftskritischen Problems wurden unsere Erwartungen voll erfüllt. Mit der eingesetzten Softwaretechnologie gelingt es offenbar gut, auch komplexere Probleme in den Griff zu bekommen.”

[The complex issue notwithstanding the problem has been solved outstandingly well and still within time and budget. The solution of our mission-critical problem has met all our expectations. It is obvious that the deployed software technology is well suited to address problems of a more complex nature, too.]

Dr. Alexander Boeck
Geschäftsführer [Managing Director]
Henkel Fragrance Center GmbH

Henkel

The software is a state-of-the-art J2EE intranet solution and utilzes the **ercatoJ** technology in all its modules.

The system interfaces with several isolated applications and is fully integrated into the corporate environment of SAP/R3 servers.

It is fair to say that this solution could not have been realized without **ercatoJ**. The savings in man power and elapsed time have been crucial for success.

Get more information:

Please contact Dr. Falk Langhammer at
Living Pages Research GmbH
Kolosseumstraße 1a
D-80469 München
Germany
Ph.: +49 (89) 189207-20
Fax: +49 (89) 189207-29
info@living-pages.de

www.living-pages.de/ercatoj

Living Pages Research