



**Product
Information**

*CocoBase® Delivers "TOP TEN"
Enterprise Persistence Features For
JPA Development!*

CocoBase® Provides A Complete Enterprise Solution
For JPA Based Development.

CocoBase® PURE POJO - Technical Paper

CocoBase® Delivers "TOP TEN" Enterprise Persistence Features For JPA Development Not Available From Other JPA Providers.

CocoBase's® long-term focus on meeting the challenging requirements of enterprise-level applications has resulted in a persistence architecture that can easily handle even the most complex requirements. These TOP TEN features provide companies a highly flexible and powerful solution for building JPA Persistence standard based enterprise applications. JPA is the Java Persistence Architecture coding standard as defined in the EJB 3 specification for use in programming the persisting of data on the Java software platform.

TOP TEN Enterprise-Level Persistence Features For JPA Development Not Available From Other JPA Tools	SUPPORTED BY COCOBASE	SUPPORTED BY OTHER JPA TOOL
Full Support For Legacy & Unstructured Database Models	CocoBase	NONE
Dynamic & Customizable SQL Mapping based Querying	CocoBase	NONE
High "Raw Data" Performance	CocoBase	NONE
Fast "Data Relationship Complexity" Scalability	CocoBase	NONE
High Productivity User Interfaces (GUI)	CocoBase	NONE
Classless Mapping	CocoBase	NONE
Custom Repository Implementations	CocoBase	NONE
Fully Distributed Persistence	CocoBase	NONE
Modifiable Standards Based Runtime	CocoBase	NONE
No Class, Code Or Byte-Code Manipulation	CocoBase	NONE

Full Support For Legacy & Unstructured Database Models

CocoBase doesn't impose any constraints or requirements on existing database models before those can be mapped as a set of POJO / JPA objects. One example of such is CocoBase's ability to map a complex object inheritance hierarchy without requiring the developer to choose a specific inheritance mapping pattern. CocoBase not only allows different inheritance mapping strategies to be combined within the same object hierarchy, but it also allows the mapping of multiple inheritance scenarios (involving a list of super-interfaces, for example) with or without abstract classes.

And CocoBase inheritance mapping capabilities are not restricted to only those already known inheritance mapping patterns (single table, joined tables, separate tables). Instead, it will smartly determine what is the best strategy based simply on "where is the data". In other words, as developers specify which column in which table contains the data for a particular object field, CocoBase will properly analyze those mappings and determine the correct strategies for mapping the object hierarchy. Further, CocoBase allows inheritance discriminator "expressions" based on multiple existing columns anywhere in the model rather than only on one specific base table column.

Other examples that deal with legacy or unstructured table models include CocoBase capabilities of mapping 'virtual' columns and keys (that is, columns that are needed in the database model, but do not exist as fields in the object model), composite primary key mapping, and hard coding table and column names and values directly into compiled SQL maps, which among other things allow the mapping of derived attributes and complex data transformations at the SQL level. So, there is no reason to change a database model that's been there for years, working flawlessly, since you can simply map it with CocoBase.

Dynamic & Customizable SQL Mapping Based Querying

CocoBase is the right ORM tool for those that ever felt the need to have fine grain control on the SQL that is generated by the mapping layer. For example, an application wishing to utilize more specific select statements with proprietary function calls, syntax and optimization hints aiming to take advantage of specific database server features, is an example of a situation where it would be useful and productive to have the level of SQL generation control CocoBase provides. And CocoBase allows the developer to insert the specific SQL desired for any of the CRUD operations.

But controlling the SQL should not be mistaken as the same as 'hard-coded' static SQL that may alternately be incorporated as mappings. CocoBase supports the hard coded approach, and goes beyond that to provide a dynamic querying system based on a custom SQL generation template defined by the developer through the map. CocoBase generates SQL dynamically from this map, and it does so intelligently in order to optimize SQL calls whenever possible (for example, by not including a update SET statement for a column that hasn't changed in the object). So the developer gets dynamic SQL generation, combined with customizations and modifications where applicable. This provides an amazing level of flexibility while having the mapping layer generate custom and dynamic SQL on the fly. This is the main idea behind the dynamic Object to Relational mapping patented technology used by CocoBase.

"Raw Data" High Performance

Base level "raw data only with no data complexity" performance of CocoBase is anywhere from 30% to 200% faster than other JPA tools. Raw data performance is a measure of only processing data from the application to the database and back with little to no complexity of the relationships between the data. Another way to quantify this is that CocoBase, as a mapping layer, is often two to three times faster than running the same operations through a JDBC driver without CocoBase. Other ORM tools are typically 15% to 20% slower than just using a JDBC driver and doing the same operations.

"Data Relationship Complexity" High Volume Scalability

Data relationship complexity has to do with the complexity and number of relationships being persisted (i.e. complexity of the object model). CocoBase, once again, provides excellent scalability that is 30% to 1,000% faster than other JPA tools. The key to CocoBase's performance is its' ability to provide Linear Scalability which retains the same speed per object as the number and complexity of the objects increases. The typical corporate application has a medium level of data relationship complexity so linear scalability is very important for a JPA tool to provide.

High Productivity User Interfaces (GUI)

The CocoBase 5 GUI was designed mainly to be easy to use and provide a high productivity environment for Java developers. The main idea behind the new GUI is to free the developer from having to specify O/R mappings whenever possible. A complete tool set is available for that purpose. For example, the project wizard will let the developer create a new CocoBase 5 project from a set of database tables and Java classes, and then activate the CocoBase's Magic Mapper wizard to automatically create the mappings between both the table and class models. Editing the mappings and corresponding models is also a GUI supervised task, so that the O/R mapping specification is always consistent with object class and table models. A problem window lists all problems and warnings regarding the existing mappings, which helps the developer to quickly identify and fix any potential issues within the current CocoBase 5 project.

Another high productivity example is with mapping inheritance, if a subclass discriminator column or expression was not properly specified; the GUI will detect that problem and include it into the problem window. The problem remains listed until it is properly fixed by the developer.

Classless Mapping

During enterprise application development requirements, there are situations where table data needs to be available to the application, but there's no corresponding Java class to accommodate that data. This is especially true when dealing with legacy database models. CocoBase allows these tables to be mapped, either within a CocoBase 5 project as an abstract class, or directly through SQL maps. Once mapped, such "classless" table data can be queried and accessed from within the application using standard `java.util.Map` objects. In other words, even when the application object model lacks a corresponding Java class, there's no need to use raw JDBC or SQL to access existing table data. This helps the application object model to be kept small and intelligible when some of the data is only sporadically accessed and doesn't justify the need to an object class on the Java side.

Custom Repository Implementations

By default, CocoBase 5 stores project mappings as a set of xml files that are optimally managed by the system. However, there may be enterprise requirements regarding the data format used to store these mappings. CocoBase 5 provides a complete repository API and base implementation that allows developers to easily implement their own repository format.

For example, to create a repository implementation where the mapping data is stored as a set of serializable object binary files would be straightforward. Further, the CocoBase 5 GUI can be easily configured to load, edit and modify custom repository implementations.

This also means developers can easily create repository implementations that support their own mapping file format used by their legacy mapping system and instantly have all of the CocoBase 5 tool set at their disposal. CocoBase 5 also allows developers to create their repository and mappings directly within their Java code, that is, no mapping files are required. That is useful for enterprise scenarios where the mappings need to constantly change and such changes need to be directly managed by applications, or even for those that wish to simplify deployment and want to avoid the use of additional files.

Fully Distributed Persistence

What makes CocoBase PURE POJO stand out for providing JPA Persistence in a distributed and/or J2EE environment is its heritage as a distributed persistence platform. Unlike other JPA implementations that derive from local persistence engines with bytecode manipulation and instances rooted in a local JVM, CocoBase has a long history as a distributed oriented platform. As such, its support for the new distributed capabilities of JPA/EJB3 put it in a unique position to provide the best solution for managing that distributed support efficiently, effectively and with enterprise reliability, performance and scalability.

Modifiable Standards Based Runtime

CocoBase 5 is pure Java and works on top of any JDBC compatible driver. The CocoBase 5 runtime includes an implementation of the EJB3 JPA (Java Persistence Architecture), a standard API for object persistence. For those engineers who wish to implement their own custom persistence manager APIs, CocoBase provides the necessary facilities to accomplish that. Its runtime is implemented as a layered architecture and developers can create an entirely different persistence model on top of the CocoBase runtime if they find it necessary.

Also, there are plugin APIs available at each layer, so that custom behaviors can be easily incorporated into the runtime. This is actually how JPA is implemented in CocoBase. It is just a higher level API built on top of the already existing CocoBase runtime infrastructure.

No Class, Code Or Byte-Code Manipulation

CocoBase 5 is a non-invasive JPA implementation, that doesn't require either bytecode or source code manipulation to implement. This is how and why CocoBase 5 Pure POJO APIs can provide distributed persistence, and a variety of other advanced features not possible for other implementations. There's no lengthy startup time for the system to modify your class files, no special requirements for those files, and POJOs that aren't EJB3 beans can be used with CocoBase 5 JPA unchanged, and unmodified.

OTHER KEY IMPORTANT ADVANTAGES AND BENEFITS UNIQUE TO COCOBASE

JPA Persistence Functionality & Features

It has been analyzed and shown in detail that CocoBase is the only tool to cover the full range of persistence features and functionality from "desktop simple" to "external enterprise". The other ORM tools providing JPA support are restrictive in that they only cover features at the "desktop simple to "workgroup moderate" with limited features at the "internal/external enterprise" levels.

Easy To Use Technology

In the new version 5.0 of CocoBase the tool has been re-architected solely around making the technology easy to use. The advanced user interface greatly simplifies the detailed process of mapping the objects with the tables and fields, and then generating the persistence code. The flexibility of the technology eliminates the need to hand-code SQL, JDBC, Java, etc. that the other JPA tools require just to complete the application.

High ROI - Lowest Total Cost of Ownership

CocoBase provides a return of \$3 for every \$1 spent purchasing it. This is a ROI of 300% which essentially makes CocoBase better than free. The time savings from the combination of the technology, the high-productivity user interface, and that CocoBase writes all the SQL, JDBC, Java code for the user is an enormous time savings. Other JPA tools require the developer to actually write their own SQL, JDBC and Java code which greatly reduces productivity, increases errors and makes the code hard to manage.

Professional Technical Support and Mentoring Services

Thought Inc. provides the best in industry support for JPA Persistence that not only covers the use of CocoBase, it also covers key related technologies as well. The technical issues in dealing with enterprise persistence can be quite complex and the ability to deal with related issues on databases, JDBC drivers, application servers, etc. is an important part of supporting CocoBase users. Mentoring services are also offered as a way to directly impart the level of expertise within the CocoBase engineering team to the project team.

Mature Technology with Customer Applications in Use for 5 Years & More

CocoBase has been available for use since 1997 as the first 100% Java ORM tool and has key Fortune 1,000 customers using the technology to run their mission-critical applications. Companies like Expeditors International, CVS Pharmacy, MCI – Verizon, United Airlines and governmental agencies like U. S. Social Security and the State of Maine all use CocoBase. Most of these customers have been using CocoBase for at least five years and more proving that the technology is mature and successful.

Commercially Built Quality Technology

CocoBase was architected from its inception as a commercial-grade technology focused directly on the set of challenging requirements at the enterprise level. The engineering team behind CocoBase has decades of experience in dealing with this particular set of technological requirements. The result is a tool and framework that has a consistent level of expertise throughout the code-base and in how it deals with providing the enterprise JPA persistence. Other JPA tools are not commercially developed and while they may be free to use, the quality is not at the same level.

Patented Technology Protection

Thought Inc. innovated the Dynamic Mapping Repository approach to persisting data on the Java platform. The company has carefully documented and patented the technology in CocoBase to provide customers a clear right to its use. CocoBase has a landmark patent, #5857197, covering the technology with additional patents granted as well. Thought Inc. makes a priority of committing company resources into the continued research and development of the technology. Other JPA tools have at best little to no patent protection which leaves the user at risk for violating existing patents.

Request An Online Introduction To CocoBase® PURE POJO V5 Now!

Please go to Thought Inc.® website (www.thoughtinc.com) and request to evaluate CocoBase® PURE POJO version 5.0 today. See for yourself why CocoBase® is chosen for enterprise level development of JPA based applications.

Please contact sales@thoughtinc.com for more information and to setup an online introduction and then get an onsite demonstration scheduled.

We, at Thought Inc.®, look forward to working with you.

COMPANY INFORMATION

THOUGHT Inc.®, in business since 1993, has been shipping the CocoBase® Enterprise O/R product since early 1997 and is currently in its' 5th major release. THOUGHT Inc.® invented and patented repository based Dynamic Object to Relational Mapping™. CocoBase® is by far, the most mature Java based O/R Mapping tool available and leads the industry in technological innovation. This is why so many of our customers rely on CocoBase®! For more information please see the website at www.thoughtinc.com.

LEGAL NOTICES

Copyrights

Copyright 2007, THOUGHT Inc., 5 Third Street suite 815, San Francisco, CA 94103 USA. All rights reserved. Copyright in these (any and all contained herein) documents is owned by THOUGHT Inc. This material is for personal use only. Republication and re-dissemination, including posting to news groups, is expressly prohibited without the prior written consent of THOUGHT Inc. This publication is provided "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement, to also include any and all technical inaccuracies or typographical errors.

Patents

CocoBase® is a patented product under patent #5857197 as well as patents pending for object caching, object navigation, dynamic object querying technologies, etc.

Trademarks

CocoBase®, THOUGHT Inc.®, Dynamic O/R Mapping™, Dynamic Transparent Persistence™, Pure POJO™ and others are all trademarks of THOUGHT Inc.® All other trademarks are owned by their respective owners.