

**codefutures**

# **CodeFutures FireStorm/DAO™ Technical Overview**

**CodeFutures™ FireStorm/DAO™ Technical Overview (July 2005)**

Copyright © 2005 Code Futures Software Ltd., including this documentation, all demonstrations, and all software. All rights reserved. The document is not intended for production and is furnished as is without warranty of any kind. All warranties on this document are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.

CODEFUTURES™ AND FIRESTORM/DAO™ ARE TRADEMARKS OF CODE FUTURES SOFTWARE, LTD.

JAVA AND ALL JAVA-BASED MARKS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF SUN MICROSYSTEMS, INC. IN THE U.S. AND OTHER COUNTRIES.

OTHER COMPANY, PRODUCT, AND SERVICE NAMES MENTIONED IN THIS DOCUMENT MAY BE TRADEMARKS OR SERVICE MARKS OF OTHERS.

## CONTENTS

1	FireStorm/DAO Introduction .....	1
1.1	FireStorm/DAO Overview .....	1
2	Data Access Object Overview .....	3
2.1	Data Access Object Design Pattern.....	3
2.2	The Benefits of Data Access Objects.....	3
2.3	Minimizing the Impact of Moving to DAO.....	4
2.4	DAO Code Generation.....	4
2.5	DAO Types Supported by FireStorm/DAO.....	4
3	FireStorm/DAO Technical Architecture.....	6
3.1	Schema Import Framework .....	6
3.2	FireStorm/DAO Project & Configuration Meta-Data .....	6
3.3	Code Generation Framework .....	7
3.4	FireStorm/DAO Ant Tasks .....	8
3.5	FireStorm/DAO™ Command Line .....	8
3.6	FireStorm/DAO Studio .....	8
4	FireStorm/DAO Product Editions .....	9
4.1	FireStorm/DAO Enterprise Edition .....	9
4.2	FireStorm/DAO Architect Edition .....	10
4.3	FireStorm/DAO OEM Edition .....	11
5	Appendix A: FireStorm/DAO Features.....	12
5.1	Database Support .....	12
5.2	DAO Code Generation.....	13
5.3	JDBC Code Generation.....	13
5.4	J2EE Code Generation .....	14
5.5	JDO Code Generation .....	14
5.6	Web Application Generation .....	15
5.7	Custom Code Generation .....	15
6	Appendix B: Technical Specifications.....	16
6.1	Supported Databases .....	16
6.2	Supported J2EE Platforms .....	17
6.3	Supported JDO Platforms .....	17
6.4	Supported Operating Systems .....	17
7	Appendix C: Useful Reference Information.....	18
7.1	CodeFutures WebLogs .....	18
7.2	CodeFutures Technical Support.....	18
7.3	Further Reading .....	18

# 1 FIRESTORM/DAO INTRODUCTION

## 1.1 FireStorm/DAO Overview

FireStorm/DAO makes Java software developers more productive by automatically generating Java source code for accessing relational databases. The benefits provided by CodeFutures Java code generation approach are higher developer productivity, better software quality, reduced complexity, and lower maintenance costs.

### DAO and ORM Code Generation

FireStorm/DAO is a Java Code Generator that can import existing database schemas (from a SQL script or from a live JDBC connection) and can then generate a complete persistence tier based on any of the following Java persistence technologies:

- Java Database Connectivity (JDBC)
- Java Data Objects (JDO)
- Enterprise JavaBeans (EJBs)
- Hibernate

FireStorm/DAO generates Java source code and configuration files that developers would otherwise have to write by hand. FireStorm/DAO generates code that is compliant with the Data Access Object (DAO) design pattern (DAO is a core J2EE design pattern). FireStorm/DAO can also generate native persistence code for Object Relational Mapping (ORM) products (such as Hibernate or JDO-based products).

FireStorm/DAO adopts a pragmatic approach of generating Java source code for data persistence that is a direct mapping of a particular relational database schema. It is also possible to define complex multi-table queries and to leverage existing database logic contained within stored procedures.

FireStorm/DAO can generate code for standalone Java as well as for leading J2EE application servers, such as JBoss, BEA WebLogic, IBM WebSphere, and Apache Tomcat. The generated source code is well-written, consistent and contains documentation. Most importantly, the generated code is production quality and has been tested in literally hundreds of deployments world-wide.

FireStorm/DAO imports database schema definitions from SQL scripts or from live databases via JDBC and then generates a complete data-access tier based on the Data Access Object (DAO) design pattern. FireStorm/DAO is a powerful yet simple alternative to traditional object-relational mapping products. Rather than attempting to map arbitrary Java code to a database schema using complex rules, FireStorm/DAO adopts a more pragmatic approach of generating Java source code that is a one-to-one representation of a relational database schema as well as allowing Custom DAO objects to be defined to reflect complex queries and calls to existing stored procedure logic.

FireStorm/DAO generates a high-performance DAO tier based on JDBC and SQL calls. The generated code can be called from standalone Java applications as well as enterprise J2EE applications. All common JDBC data types are supported including BLOB and CLOB types. Auto-increment columns and auto-generated sequences are also supported.

FireStorm/DAO generates a complete Web application for interacting with the generated DAO tier. The generated application is based on the Apache Struts framework or plain JSPs and contains full functionality for inserting, updating, deleting, and searching records using the DAO tier. FireStorm/DAO can generate DAOs for partial database schemas by selecting the required views and tables.

FireStorm/DAO Architect Edition offers all of the features of the Enterprise Edition but also allows new custom code generation templates to be developed and integrated with the FireStorm/DAO Studio environment.

## 2 DATA ACCESS OBJECT OVERVIEW

### 2.1 Data Access Object Design Pattern

Business applications almost always need access to data from relational or object databases and the Java platform offers many techniques for accessing this data. The oldest technique is to use the Java Database Connectivity (JDBC) API, which provides the capability to execute SQL queries against a database and then fetch the results, one column at a time. Although this API provides everything a developer needs to access data and to persist application state, it is a cumbersome API to develop against and does not fit well within an object-oriented design.

Java 2 Enterprise Edition (J2EE) offers a newer persistence framework in the form of Entity Beans, a subset of the Enterprise JavaBean (EJB) framework. Many developers are now looking to alternative persistence frameworks, such as Java Data Objects (JDO) and Hibernate. CodeFutures is committed to providing support for all the major DAO design options.

### 2.2 The Benefits of Data Access Objects

The Data Access Object (DAO) design pattern provides a technique for separating object persistence and data access logic from any particular persistence mechanism or API. There are clear benefits to this approach from an architectural perspective. The DAO approach provides flexibility to change an application's persistence mechanism over time without the need to re-engineer application logic that interacts with the DAO tier. For example, there may be performance benefits in changing an application's performance mechanism from using Entity Beans to using direct JDBC calls from a session bean, or even a move to an alternative persistence framework, such as JDO. Without a DAO tier in place, this sort of transition would require extensive re-engineering of existing code.

The DAO design pattern also provides a simple, consistent API for data access that does not require knowledge of JDBC, EJB, or JDO interfaces. A typical DAO interface is shown below.

```
public interface CustomerDAO
{
    public void insert(Customer customer)
        throws CustomerDAOException;

    public void update(CustomerPK pk, Customer customer)
        throws CustomerDAOException;

    public void delete(CustomerPK pk)
        throws CustomerDAOException;

    public Customer[] findAll()
        throws CustomerDAOException;

    public Customer findByPrimaryKey(String email)
        throws CustomerDAOException;

    public Customer[] findByCompany(int companyId)
        throws CustomerDAOException;
}
```

It is important to note that DAO does not just apply to simple mappings of one object to one relational table, but also allows complex queries to be performed

and allows for stored procedures and database views to be mapped into Java data structures.

## 2.3 Minimizing the Impact of Moving to DAO

Moving to the DAO design pattern using manual line-by-line coding requires a significant amount of repetitive source code to be produced for no immediate advantage over using JDBC, EJB, or JDO directly. For many developers, this disadvantage is good enough reason to ignore the long-term benefits of using a architecturally superior approach, especially where there are strict project deadlines. Without the code generation advantages of FireStorm/DAO, it is not easy to justify to a project manager or project sponsor the extra time and cost of manually writing DAO code, regardless of any future benefits.

FireStorm/DAO makes it the transition to the DAO design pattern easy to justify.

## 2.4 DAO Code Generation

CodeFutures' solution to the manual coding problem is to automate the production of a DAO tier, as well as automating the actual implementation logic for whichever persistence framework is deemed appropriate for an application. This approach is easy to adopt because almost all databases use a standard language for defining their structure (SQL).

## 2.5 DAO Types Supported by FireStorm/DAO

FireStorm/DAO supports three types of Data Access Object (DAO): Table, View, and Custom.

### 2.5.1 Table DAOs

The Table DAO is the most widely used Data Access Object. Each Table DAO represents a single table and contains methods to insert, update, and delete single rows using the following signatures (using as an example a Customer table):

```
public void insert(Customer customer) throws CustomerDaoException;
```

```
public void update(CustomerPk pk, Customer customer) throws  
CustomerDaoException;
```

```
public void delete(CustomerPk pk) throws CustomerDaoException;
```

In addition to this, there are also numerous 'finder' methods generated. FireStorm creates default finders when you first import a schema but these are fully customizable. Some examples of finder methods:

```
public Customer findByPrimaryKey(CustomerPk pk) throws  
CustomerDaoException;
```

```
public Customer[] findWhereLastNameEquals(String lastName) throws  
CustomerDaoException;
```

```
public Customer[] findByCountry(int countryId) throws  
CustomerDaoException;
```

### 2.5.2 View DAOs

The View DAO is generated for each view in the database. This DAO offers the same finder methods as the Table DAO but obviously does not provide the insert, update, and delete operations because views are read-only.

### 2.5.3 Custom DAOs

Custom DAOs are used when you have a complex SQL query that goes beyond the simple CRUD (create, update, delete) operations on a single table. Examples include SQL queries that perform a join between several tables, queries that perform aggregation using the GROUP BY operator, and bulk update or delete queries.

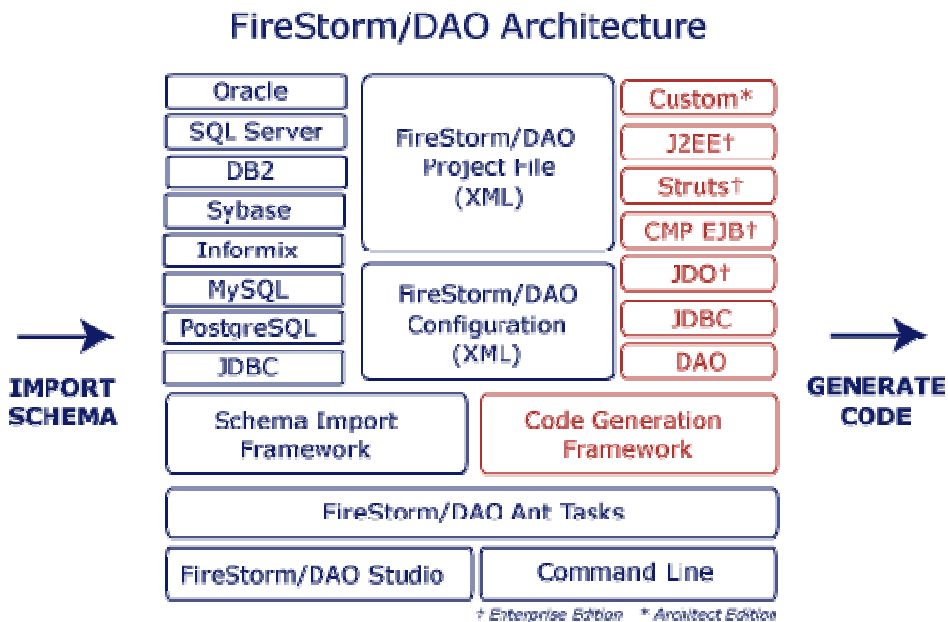
Example queries that the Custom DAO feature can support:

```
SELECT a.*, b.* FROM a, b WHERE a.id = b.id AND b.install_date  
between ? and ?
```

```
DELETE customer WHERE status = ? AND create_date = ?
```

```
SELECT product, count(*) FROM product WHERE download_date > ? GROUP  
BY product
```

## 3 FIRESTORM/DAO TECHNICAL ARCHITECTURE



### 3.1 Schema Import Framework

FireStorm/DAO can import relational database schemas from Database Definition Language (DDL) scripts and can also import schemas from a JDBC connection to a live database.

#### 3.1.1 Importing from supported databases

FireStorm/DAO contains specific support for the DDL syntax and data types used by Oracle, DB2, Microsoft SQL Server, Sybase, Informix, MySQL, InnoDB, PostgreSQL. FireStorm/DAO also has intelligent support for auto-increment columns and sequences used by MySQL, Oracle, and DB2.

#### 3.1.2 Importing with generic databases

FireStorm/DAO can import schemas from any relational database that supports ANSI-SQL or has a JDBC driver available.

### 3.2 FireStorm/DAO Project & Configuration Meta-Data

#### 3.2.1 Project Meta-Data

FireStorm/DAO projects are stored in an easy-to-understand XML format making it easy to share projects with other developers and store them in source control

systems such as CVS. FireStorm/DAO ships with an XML Schema Definition (XSD) file making it easy to manipulate the project meta-data using external tools.

### **3.2.2 Configuration Meta-Data**

FireStorm/DAO stores all of its configuration data in XML format.

## **3.3 Code Generation Framework**

FireStorm/DAO has a powerful code generation framework allowing custom code generation modules to be developed (Architect Edition only). The API provides access to all of FireStorm/DAO's configuration and project meta-data in a simple Java object-model removing the need to parse the underlying XML files. FireStorm/DAO's code generation framework also provides a highly object-oriented approach to creating Java source files.

### **3.3.1 Custom Code Generation Module**

This Architect Edition-only module allows the DAO code generation templates to be customized to:

- Optimize performance based on specific project requirements
- Integrate more closely with existing build environments, such as internal coding standards
- Add environment-specific extra custom features such as caching, security, auditing, and clustering

The code generation templates are written in standard Java so there is no need to learn a proprietary code generation template language as with other solutions. There is no limit on the number custom of DAO design templates that can be developed – possibly using different DAO templates for different parts of the same application. Additional non-DAO custom code generation templates can also be developed and used with FireStorm/DAO. The custom code generation templates can be used with the Enterprise Edition.

### **3.3.2 J2EE Code Generation Module**

The J2EE Code Generation Module creates JSP pages for each DAO allowing them to be tested very easily. Each DAO is also wrapped in a Stateless Session Bean façade making it easy to publish the DAO objects as transactional components as part of a Service-Oriented Architecture (SOA). J2EE deployment descriptors are also generated for the target platform (JBoss, WebLogic, WebSphere). When generating code for J2EE it is possible to choose between JDBC, CMP EJB, or JDO for the DAO implementation code.

### **3.3.3 Apache Struts JSP Code Generation Module**

Generates a complete Web application for interacting with the generated DAO tier. The generated application is based on the Apache Struts framework and contains full functionality for inserting, updating, deleting, and searching records using the DAO tier. There is also the option for generating plain JSPs without using the Struts framework.

### **3.3.4 CMP EJB Code Generation Module**

Generates Entity Bean source code for each table as well as CMP deployment descriptors for the target platform.

### **3.3.5 JDO Code Generation Module**

Generates JDO classes and DAO implementation based on JDO. Generates JDO deployment descriptors.

### **3.3.6 JDBC Code Generation Module**

Generates DAO implementation using plain JDBC/SQL calls.

### **3.3.7 DAO Code Generation Module**

Generates Data Access Object (DAO) interfaces, Data Transfer Object (DTO) classes, and DAO exception classes. The JDBC, EJB CMP, and JDO code generation modules generate concrete implementations of the DAO interfaces.

## **3.4 FireStorm/DAO Ant Tasks**

FireStorm/DAO contains Ant tasks for importing schemas and generating code, making it easy to incorporate FireStorm/DAO into automated build systems.

## **3.5 FireStorm/DAO Command Line**

The command line interface is ideal when running FireStorm/DAO on remote Unix servers where only a console interface is available.

## **3.6 FireStorm/DAO Studio**

FireStorm/DAO Studio is an easy-to-use graphical user interface for importing schemas, manipulating meta-data, and generating code.

## 4 FIRESTORM/DAO PRODUCT EDITIONS

### 4.1 FireStorm/DAO Enterprise Edition

FireStorm/DAO Enterprise Edition offers the choice of generating a DAO tier using either JDBC, Java Data Objects (JDO), or Enterprise JavaBeans (EJB). The Enterprise Edition generates a complete Apache Struts JSP Web application for interacting with the generated DAO tier.



#### 4.1.1 Enterprise Edition Features

- Import database schemas from a JDBC connection
- Import database schemas from SQL/DDDL scripts
- Reverse-engineer stored procedures
- Round-trip engineering (import modifications made to a schema)
- Create new schemas using a simple user interface
- Imported schemas can be modified using a simple user interface
- Generates Data Access Object (DAO) interfaces
- Generates Data Access Object (DAO) exception classes
- Generates Data Transfer Object (DTO) classes
- Generates Data Access Object (DAO) classes for JDBC
- Generates Data Access Object (DAO) classes for Java Data Objects (JDO)
- Generates Data Access Object (DAO) classes for Enterprise JavaBeans (EJB)
- Generates Session Bean facade for each DAO interface
- Generates Web pages based on Apache Struts framework
- Generates Web pages based on plain Java Server Pages
- Dynamic Update Method
- Interactive SQL Tool
- Database Schema Migration
- Optimized JDBC Code
- Import Generation for Partial Schema
- Generates JDO meta-data and JDO properties file
- Generates CMP deployment descriptors
- Generates J2EE deployment descriptors
- Generates ANT build scripts

The FireStorm/DAO Enterprise Edition generates a complete JSP Web application for interacting with the generated DAO tier. The generated application is based on plan JSPs or the Apache Struts framework and contains full functionality for inserting, updating, deleting, and searching records using the DAO tier. For each table, view, and custom DAO, the following artifacts are generated:

- Struts Actions for insert, update, and delete operations
- Struts Actions for each defined finder method
- ActionForm bean to represent DTO
- Relevant entries in struts-config.xml and tiles-defs.xml
- JSP pages for viewing and editing each DAO
- JSP pages for displaying input form and results page for each finder method

## 4.2 FireStorm/DAO Architect Edition

FireStorm/DAO 2 Architect Edition builds on the power of the Enterprise Edition by allowing custom code generation modules to be built to meet any bespoke requirements. The Architect Edition also ships with full source code for the JDBC, JDO, and EJB code generators from the Enterprise Edition.



FireStorm/DAO Architect Edition offers all of the features of the Enterprise Edition but also allows new custom code generation templates to be developed and integrated with the FireStorm/DAO Studio environment.

The code generation templates are written in standard Java so there is no need to learn a proprietary code generation template language as with other solutions. There is no limit on the number custom DAO design templates can be developed – possibly using different DAO templates for different parts of the same application. Additional non-DAO custom code generation templates can also be developed and used with FireStorm/DAO. The custom code generation templates can be used with the Enterprise Edition.

FireStorm/DAO Architect Edition includes the source code for the each of the DAOs (JDBC, EJB CMP, and JDO). This allows FireStorm/DAO to be customized:

- Optimize performance based on specific project requirements
- Integrate more closely with existing development procedures and build environments, such as internal coding standards
- Add environment-specific extra custom features such as caching, security, auditing, and clustering

The maximum benefits of FireStorm/DAO are usually derived by a software development group using the Architect Edition to customize the DAO design template for its environment.

An experienced software developer within a software development team customizes the DAO template. Then any other programmers that want to develop a database persistence tier use the Enterprise Edition and the custom DAO template provided by the DAO architect.

### 4.2.1 Architect Edition Features

- Contains all functionality of Enterprise Edition
- Provides API allowing custom code generation modules to be developed
- FireStorm API provides access to FireStorm project file meta-data
- Contains full source code for JDBC, JDO, Hibernate and J2EE code generation modules
- Contains sample source code for code generation customization
- Contains previews of new features not generally available

### 4.3 FireStorm/DAO OEM Edition

FireStorm/DAO OEM Edition is specifically aimed at software product vendors. The functionality is the same as FireStorm/DAO Architect Edition, with additional, normally unavailable, internal technical documentation and optionally access to all source code. CodeFutures' commitment to flexible licensing arrangement and product packages ensures that ISVs of any size can be accommodated. FireStorm/DAO is highly portable and easy to integrate with third-party software.



## 5 APPENDIX A: FIRESTORM/DAO FEATURES

### 5.1 Database Support

Feature	Enterprise	Architect
<ul style="list-style-type: none"> <li>Import relational database schemas via JDBC. Any database with a standard-compliant JDBC driver can be supported.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Import relational database schemas from any ANSI-SQL compliant Database Definition Language (DDL) script.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>In addition to exposing tables, views, and custom SQL statements as Data Access Objects, FireStorm/DAO exposes Stored Procedures as DAOs. Available for Sybase, Oracle, and SQL Server.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Round trip engineering: A synchronization wizard makes it easy to import new versions of a database (JDBC or SQL) schema without losing modifications made to a FireStorm/DAO project file. Database synchronization works by automatically importing new tables, views, and columns, without losing modifications made to existing objects.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Automatically recognizes MySQL auto-increment columns on import.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Support for DB2 and Oracle sequence types for auto-generated primary-keys.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Support for all standard JDBC data types including BLOB and CLOB.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Dynamic Update Method: Firestorm/DAO users can choose to generate static update methods that update every column every time, or a new dynamic update method that only updates columns that have been modified.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>By default, BLOB types are automatically mapped to byte arrays and CLOB types are mapped to Strings. The implementation code for reading BLOB and CLOB columns is contained in a generated base class that can be customised if required.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Exposes all DAO types as Stateless Session Beans making it easier to integrate databases with SOA environments. If the JDBC DAO implementation strategy is chosen then the dynamic finder methods are also exposed for maximum flexibility.</li> </ul>	Yes	Yes

<ul style="list-style-type: none"> <li>Partial schema support: specify a list of tables and views to import from the database where DAOs are only required for a subset of the database. FireStorm/DAO Studio provides a graphical user interface for selecting the database elements for which a DAO tier is generated. The developer specifies a list of tables and views to import from the database.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Custom Data Access Objects with dynamic SQL. Effectively this means that the same DTO and DAO mapping can now be used with multiple SQL statements. A typical use-case for using Dynamic SQL statements with a Custom DAO is where a multi-table SELECT is implemented as a Custom DAO but there is a need for multiple "finder" methods.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Interactive SQL Tool allows queries to be tested directly from FireStorm/DAO</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Database Schema Migration Tool makes it possible to change the target database in a FireStorm/DAO project file and any SQL data types not supported by the new target database is automatically change to supported types, making it easy to migrate a schema to a new database platform.</li> </ul>	Yes	Yes

## 5.2 DAO Code Generation

Feature	Enterprise	Architect
<ul style="list-style-type: none"> <li>Generate Data Access Object (DAO) interfaces, Data Transfer Object (DTO) classes, and Data Access Object (DAO) exception classes for each DAO defined in the project file.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Option to generate logging calls based on System.out.println() or log4j.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Generates complete Ant build script to compile and package the generated code.</li> </ul>	Yes	Yes

## 5.3 JDBC Code Generation

Feature	Enterprise	Architect
<ul style="list-style-type: none"> <li>Generate Data Access Object (DAO) implementation that uses plain JDBC calls and SQL statements.</li> </ul>	Yes	Yes

<ul style="list-style-type: none"> <li>Support for custom SQL statements, including joins, bulk updates and deletes when generating DAO implementation based on JDBC.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Generates JDBC DAO code that is concise and is closer to human-authored code. Variable names are only prefixed with underscores if there is the possibility of a name clash with method parameters, making the code more readable. Generated finder methods contain a single line of code that delegates to a dynamic finder method rather than containing dozens of lines of repetitive code.</li> </ul>	Yes	Yes

## 5.4 J2EE Code Generation

Feature	Enterprise	Architect
<ul style="list-style-type: none"> <li>Generate Container-Managed Persistence (CMP) Entity Beans and CMP deployment descriptors for each Table DAO that has a primary-key.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Generate fully working Web application, based on Struts and Tiles, for immediate testing of the generated DAO persistence and query service.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Generate Stateless Session Bean facade to enable Data Access Object (DAO) classes to be published as transactional services as part of a Service-Oriented Architecture (SOA).</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Generate J2EE deployment descriptors for IBM WebSphere, BEA WebLogic, and JBoss, and any standards-based J2EE application Server.</li> </ul>	Yes	Yes

## 5.5 JDO Code Generation

Feature	Enterprise	Architect
<ul style="list-style-type: none"> <li>Generate Java Data Object (JDO) classes and DAO implementation for each Table and View DAO.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Generate Java Data Object (JDO) meta-data files for the JDO reference implementation.</li> </ul>	Yes	Yes

## 5.6 Web Application Generation

Feature	Enterprise	Architect
<ul style="list-style-type: none"> <li>Generate Web applications for deployment onto standalone web containers (for example, Tomcat 4.x) or to full J2EE platforms. The generated Web application contains JSP pages, Struts Actions, and a DOA tier based on JDBC or CMP.</li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Generate a complete JSP Struts Web application for interacting with the generated DAO tier. The generated application is based on the Apache Struts framework and contains full functionality for inserting, updating, deleting, and searching records using the DAO tier. For each table, view and custom DAO, the following artifacts are generated: <ul style="list-style-type: none"> <li>- Struts Actions for insert, update, and delete operations</li> <li>- Struts Actions for each defined finder method</li> <li>- ActionForm bean to represent DTO</li> <li>- Relevant entries in struts-config.xml and tiles-defs.xml</li> <li>- JSP pages for viewing and editing each DAO JSP pages for displaying input form and results page for each finder method</li> </ul> </li> </ul>	Yes	Yes
<ul style="list-style-type: none"> <li>Generate plain JSP Web application (not using the Struts framework).</li> </ul>	Yes	Yes

## 5.7 Custom Code Generation

Feature	Enterprise	Architect
<ul style="list-style-type: none"> <li>Write custom code generators in Java with full access to the FireStorm/DAO code generation API and project meta-data. No proprietary scripting or template language to learn. The code generation templates are Java code.</li> </ul>	No	Yes
<ul style="list-style-type: none"> <li>Full access to the source code for the JSP, Struts JSP, JDBC DAO, JDO DAO, Hibernate DAO and EJB CMP DAO code generation modules allowing them to be completely customized.</li> </ul>	No	Yes
<ul style="list-style-type: none"> <li>Preview source code for new features in upcoming product releases.</li> </ul>	No	Yes
<ul style="list-style-type: none"> <li>Source code samples for customization and optimization of code generation templates.</li> </ul>	No	Yes

## 6 APPENDIX B: TECHNICAL SPECIFICATIONS

### 6.1 Supported Databases

FireStorm/DAO should work with any JDBC-compliant database. The database that CodeFutures or CodeFutures' customer have used FireStorm/DAO with include:

- DB2 7.1, 7.2, 8.1
- MySQL 3.23, 4.0, 4.1
- PostgreSQL 7.1.2, 7.2, 7.3, 7.4
- Oracle 8i, 9i, 10g
- Sybase 12.5 (JConnect 5.5)
- HypersonicSQL 1.61, 1.7.0, 1.7.2, 1.8
- Microsoft SQL Server 2000
- SAP DB 7.3
- Max DB
- Informix
- Ingres
- FrontBase
- Mckoi SQL
- Firebird
- DBMaker
- Interbase
- Progress 9
- Pointbase Embedded 4.3
- HP NonStop SQL/MX 2.0
- IBM Cloudscape
- Microsoft Access
- Apache Derby
- InnoDB
- Adabase D
- MimerSQL
- seeMore
- Hypersonic Database Engine
- Axion
- DaffodilDB

## 6.2 Supported J2EE Platforms

FireStorm/DAO generates code for the following J2EE platforms:

- BEA WebLogic 6.1
- BEA WebLogic 7
- BEA WebLogic 8.1
- IBM WebSphere 4
- IBM WebSphere 5
- JBoss 3.2
- Tomcat
- Sun Java System Application Server
- Any standards-compliant J2EE Application Server

## 6.3 Supported JDO Platforms

FireStorm/DAO generates code for the following JDO platforms:

- JDO Reference Implementation 1.0.1

## 6.4 Supported Operating Systems

All editions of FireStorm/DAO run on the following operating systems:

- Microsoft Windows 2000, Windows NT4, Windows XP
- Solaris
- Linux
- Mac OS X
- Any OS supporting Java 2 Standard Edition 1.4 or greater

## 7 APPENDIX C: USEFUL REFERENCE INFORMATION

### 7.1 CodeFutures WebLogs

Corporate WebLog:

<http://www.codefutures.com/Weblog/codefutures/>

Andy Grove's (CodeFutures CTO) WebLog:

<http://www.codefutures.com/Weblog/andygrove/>

### 7.2 CodeFutures Technical Support

CodeFutures offers Premium Support: includes 12 months of email support and all available upgrades, including all new major and minor product releases, during the support period. Premium Support is 20% of the software license price.

For support on FireStorm/DAO, please email: [support@codefutures.com](mailto:support@codefutures.com)

### 7.3 Further Reading

Product material on the Document Library on the CodeFutures Web site:

- CodeFutures Corporate Overview
- FireStorm/DAO Product Overview
- FireStorm/DAO Architect Edition Overview
- FireStorm/DAO Product Pricing Sheet
- DSNet Technical Case Study
- Trend S.p.a. Customer Profile

Additional documents are available on request from [sales@codefutures.com](mailto:sales@codefutures.com):

- CodeFutures Customer Testimonials
- FireStorm/DAO Product Roadmap
- CodeFutures Manager's Guide
- CodeFutures Unique Product Offering
- CodeFutures Corporate Overview (PowerPoint Presentation)
- CodeFutures Technology Vision (PowerPoint Presentation)