NEFS Enterprise Console

Tour

Netspective

Palmer Business Park
4550 Forbes Blvd, Suite 320
Lanham, MD 20706

(301) 879-3321

http://www.netspective.com
info@netspective.com

Copyright

Disclaimer and limitation of liability

Customer Support

Customer support is available through e-mail via support@netspective.com

NEFS Enterprise Console Tour

Table of Contents

1. Netspective Enterprise Console

The NEF and all application components built with it can be viewed with the *Netspective Enterprise Console* (NEC). The Console is a Sparx servlet that provides a browser-based administrative interface to all of the dynamic Netspective Enterprise Frameworks Suite (NEFS) components and objects associated with your application, enabling your development team to collectively manage the application development process.

The Console provides a "window into your application". It serves as an invaluable debugging aid, diagnostic tool, project documentation management system, and process artifact collection utility in one thin client application. No other framework provides such a visual representation of different aspects for your application.

And, because it's a standard NEFS application, it's a great demonstration and sample application.

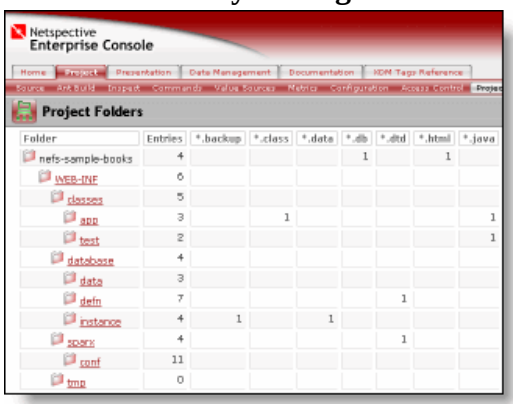
The Enterprise Console is automatically available to all Sparx-based applications during development and can be optionally turned off in production (for security).

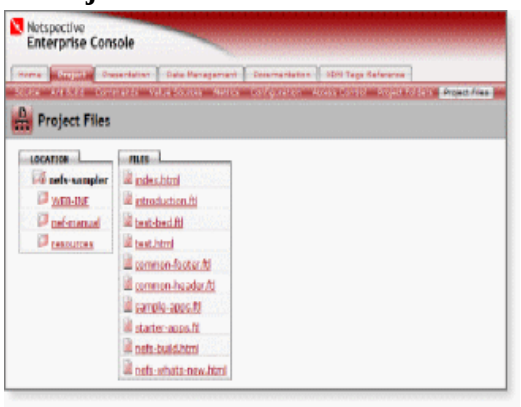
2. Central Management and Visualization of Your Project

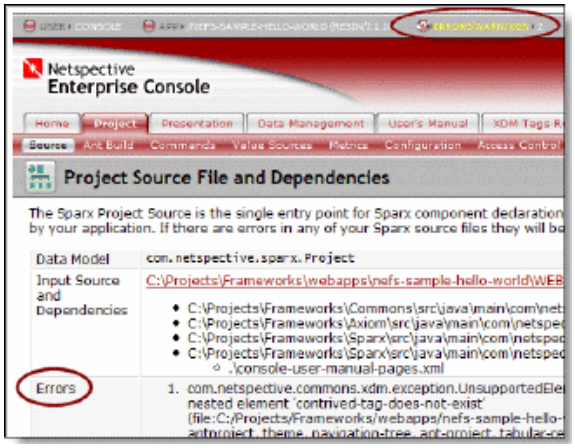
2.1. Project

The Console serves as the central repository for all your project components. Unlike other existing frameworks, NEFS Console provides a window into your application. It enables you to manage and visualize all the static and dynamic components using one thin client.

Table 1. Centrally Manage and Visualize Your Project







Using the Console you can access your project folders, all the project files and the error and warning messages. This is regardless of which IDE you are using as the Console is a thin-client.

3. Automatic Implementation Documentation

The Console provides a centralized location for all project documentation for any application. Instead of storing application code and programmer documentation separately, Console brings tag documentation, javadocs and other project documents into a single easily accessible place. Managers will no longer need to hunt for documents.

3.1. Functional Specs

Instead of having to create functional specifications and other implementation documentation manually, Console automatically documents (using the XML definitions and XSLT stylesheets) all the pages, forms/dialogs, sql statements, schema objects, and other programming artifacts. The HTML based design documentation and functional specs help your team visualize various application components.

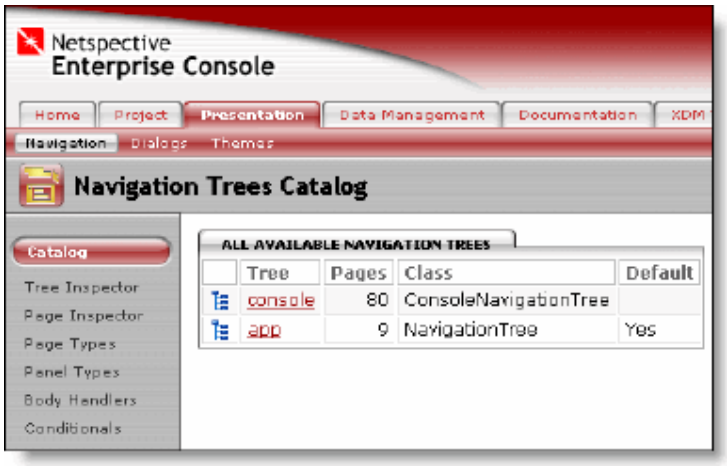
Note

Also, using simple stylesheets developers can create docbook or MS word versions of their application components (navigation, validation, dialogs, fields, etc).

3.2. Navigation

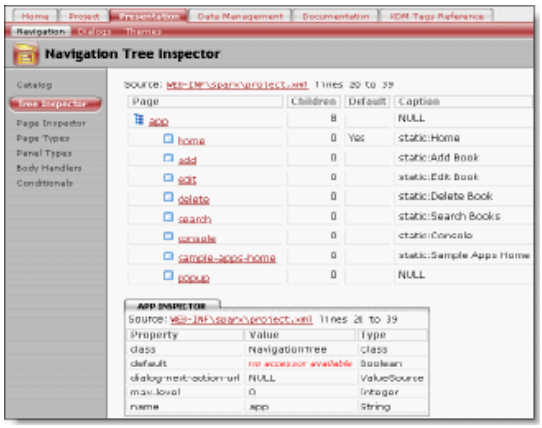
The Console provides a complete HTML based representation of the workflow of your application. Using the Console's Navigation Tree Inspector you can easily.

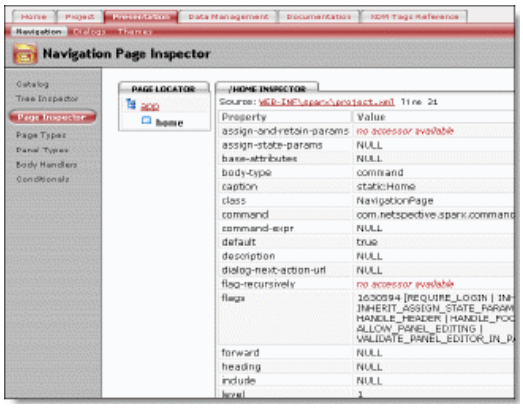
All of the navigation trees that are externalized in XML files appear in the Navigation Catalog within Console. You can use this view to review all the pages that appear in an application.



You may further analyze a navigation tree on the page level and view complete functional description for each page within the navigation tree.

Table 2. Viewing Page-Level Functional Description



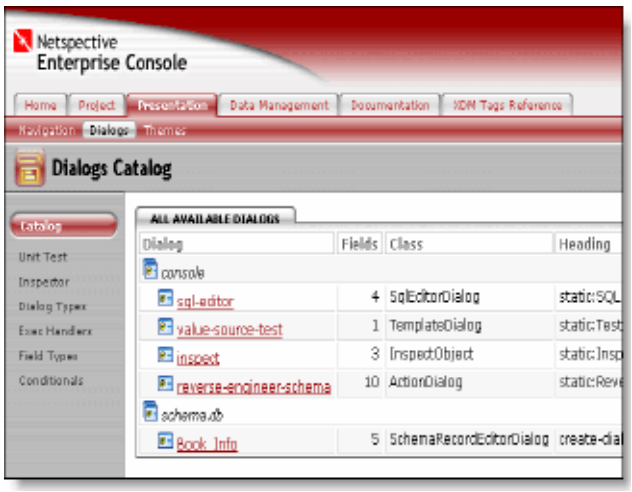


3.3. Dialogs

Almost all aspects of forms/dialogs are automatically captured and documented within Console. This includes a list of all the forms, any classes that are extended, all fields, field types, labels, conditionals, and many other features.

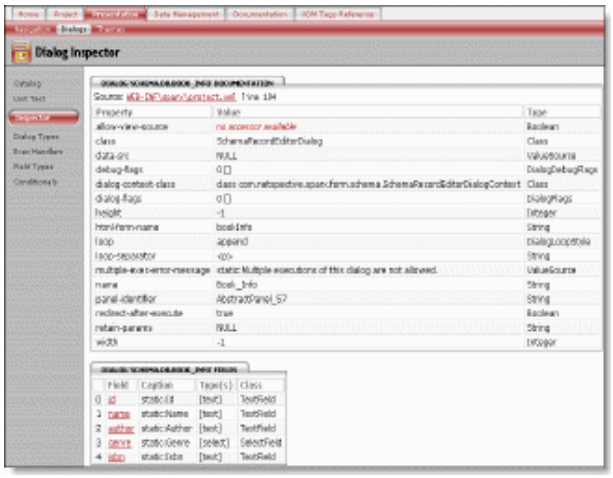
All of the dialogs that are externalized in XML files appear under the Dialogs Catalog in Console. Analysts or customers can use this view to review all the dialogs that appear in an application.

Figure 1. Managing Dialogs



Each dialog may be further analyzed by reviewing a user- friendly functional specification view. This view may be shared with analysts or clients to ensure requirements completeness.

Figure 2. Viewing Functional Specs of a Dialog



3.4. Database

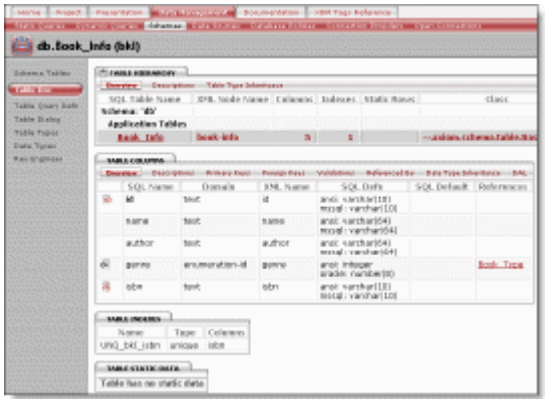
3.4.1. Schema Doc

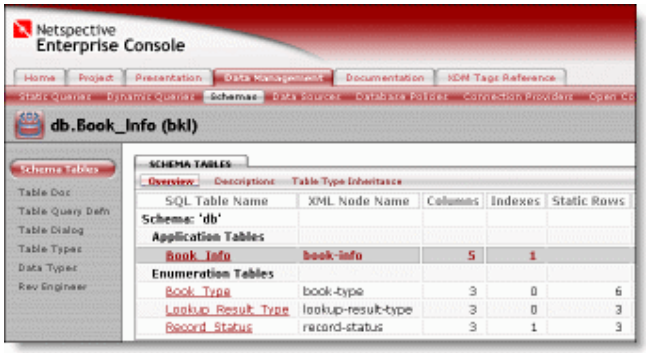
The NEFS allows you to define the structures (tables, columns, relationships) needed to store your data using XML file. Schema tags support full relational integrity and may actually be used to define meta data to support Java-based relational integrity for existing and legacy databases that may not be built with fully relational integrity.

Now you don't need experienced DBAs to create consistent, high-quality SQL DDL during the design and construction phases of your application. Also, because almost all schema resources are defined in XML, Sparx allows for re-use of Schemas across applications and different database vendors. This continued re-use provides standard behaviors for columns and tables across your enterprise, ensuring stable applications.

Another benefit of this approach is that by using the XML-based schema definition, all the database tables, columns, indexes, SQL DDL, and Java data access can be automatically documented in the Console.

Table 3. Automatically Generate Schema Doc



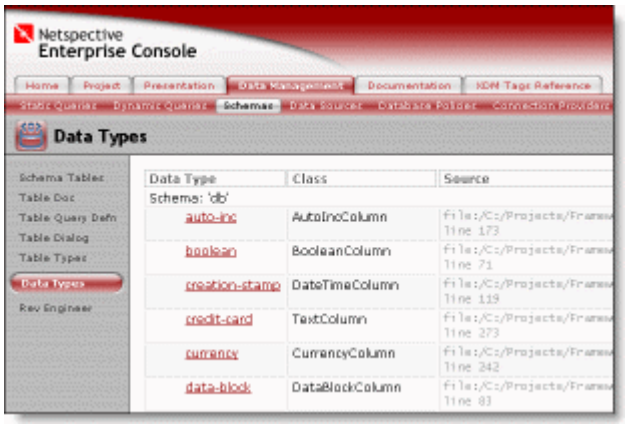


3.4.2. Data Types Dictionary

NEFS provides packaged database components through its Axiom framework. This results in significantly reduced time to deployment of data-driven applications. Axiom provides ready-to-use e-business database components such as Datatypes, Tabletypes, and Indextypes which form the basis of an XML-based data dictionary of Tables, Columns, Indexes and Data.

Datatypes serve as “column templates” that allow a programmer to specify a column type. They may be inherited from other datatypes, allowing better reuse and object-orientation in relational databases. These datatypes can be easily defined using XML tags. This RDBMS-neutral and consistent data dictionary can be viewed from within the Console.

Figure 3. Manage DataTypes Dictionary

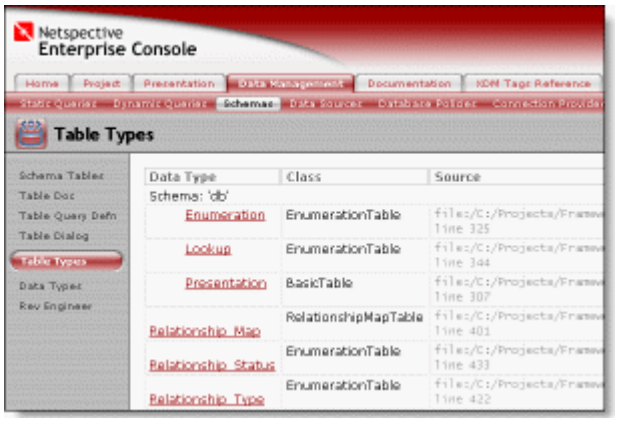


3.4.3. Table Types Dictionary

The Axiom Table Types dictionary contains definitions for generic tables and behaviors that can be inherited by real tables. Table types work as “table templates” that allow a programmer to specify a table type. They may be inherited from other table types, allowing better reuse and object-orientation in relational databases.

You can view the complete documentation for the built-in as well as custom table types through the Console. Note that this documentation is automatically generated when you define a new table type.

Figure 4. Avoid Rewriting Common Schema Elements Using Dictionaries



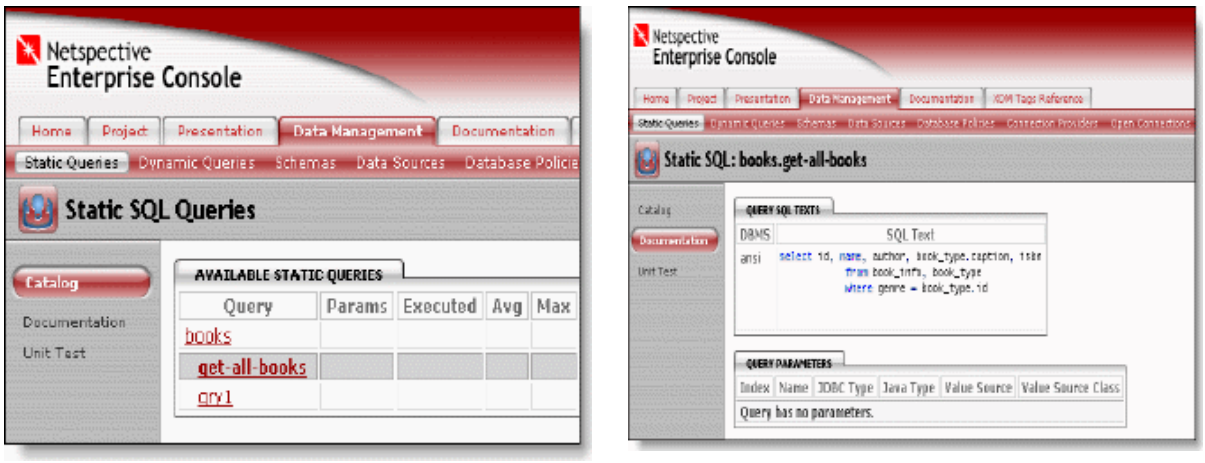
The dictionaries shield your database programmers from rewriting common schema elements for every new application.

3.4.4. SQL statement libraries

Almost all components of a SQL statement are automatically captured and documented within the Console. This includes a list of all the SQL statements used throughout the application, the parameters they require, and any views that may be attached to them.

NEFS allows all SQL statements and dynamic parameters used in a project to be specified in one or more SQL files using XML. All of the SQL statements that are externalized in these XML files appear in a single Console page.

Table 4. Manage Static SQL Queries

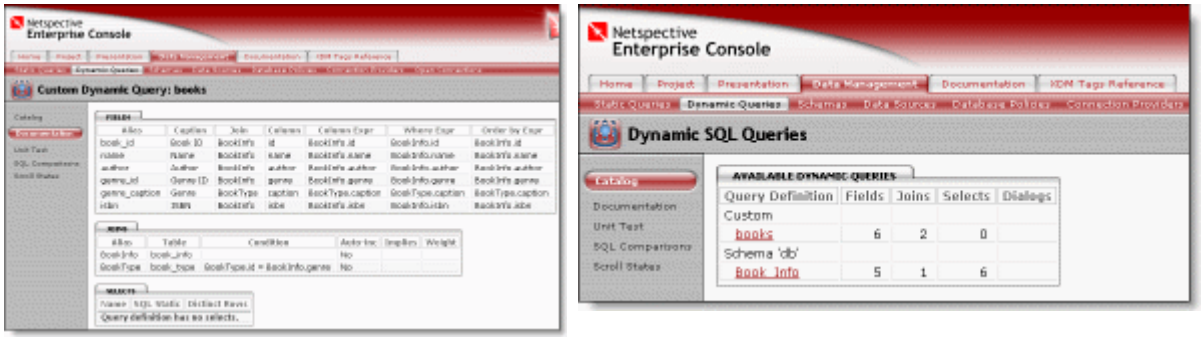


Programmers and manager can use this view to review all the SQL statements that are used in an application.

3.4.5. Dynamic Queries

NEFS allows developers to define report tables, columns, joins, sort orders and other important data through the use of Meta information about data relationships. The entire suite of fields, joins, rules, conditions, select dialogs, and where expressions from Query Definitions are fully documented within Console.

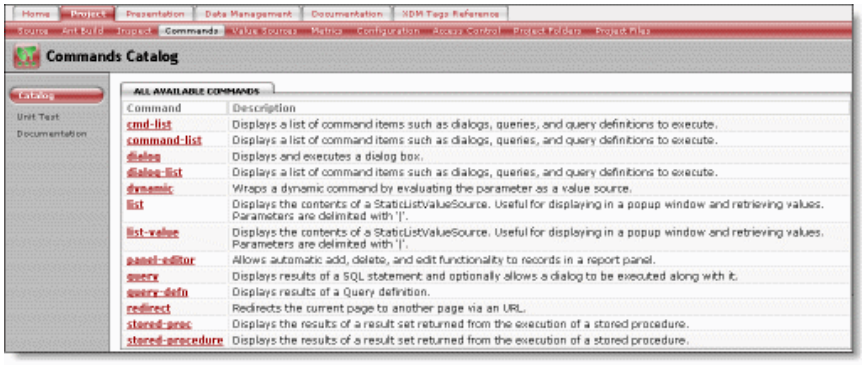
Table 5. Centrally Manage Dynamic Queries



3.4.6. Commands

Commands are Java classes that execute arbitrary tasks defined either by you or the framework. They are used to encapsulate common application logic and reuse that logic across pages and dialogs/forms. All the built-in and custom commands can be viewed through the Commands catalog in the Console.

Figure 5. Commands Catalog for Built-in and Custom Commands



The full documentation for each command is also available automatically through the Console.

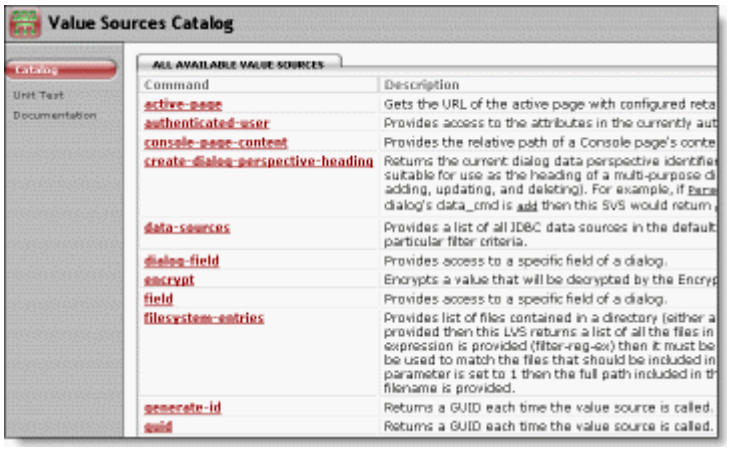
Figure 6. Command Documentation

3.4.7. Value Sources

Value Sources allow dynamic data to be included in XML without creating a programming language inside XML They allow common business logic and business values to be stored in shareable instance and then used either in XML or Java files where necessary.

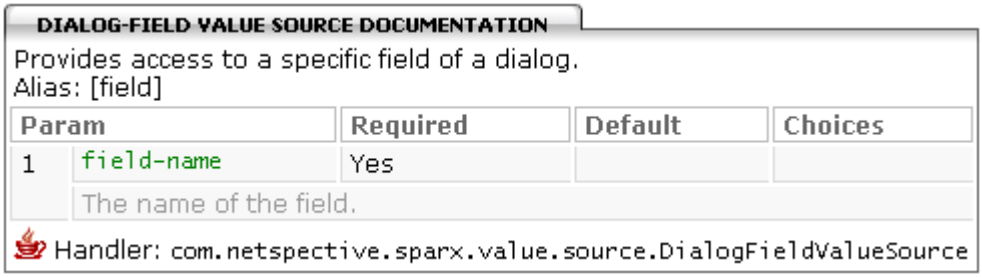
The NEF ships with many built-in value sources and you can create as many value sources as you need on your own. You can view a list of all of the value sources available to your project (including all built-in value sources and your own custom value sources) through the Value Sources Catalog within the Console.

Figure 7. Value Sources Catalog for Built-in and Custom Value Sources



The full documentation for each value source is automatically available through the Console.

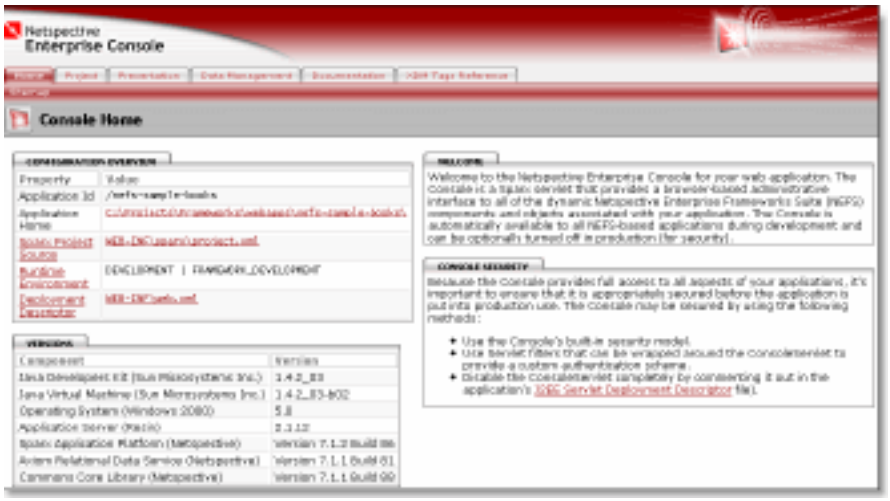
Figure 8. Value Source Documentation



4. Configuration

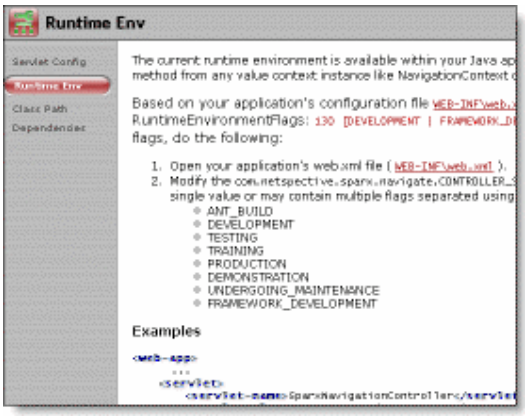
The Console allows you to centrally manage all of your configuration parameters. It allows multiple properties to be defined in a single XML file (web.xml), complete with variable replacements. All of these Configuration variables are displayed on a single page within the Console, displaying the variable names and their values.

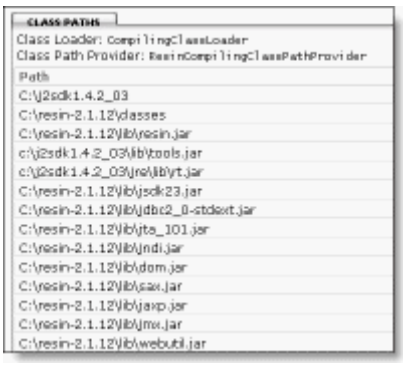
Figure 9. Application Configuration Management



Many servlet-context variables are also visible within Console, including the current execution mode (Production, Testing, or Development). The NEFS supports multiple runtime environments to support different phases of development. Now you can easily access the full configuration details from a central console application. The availability of complete configuration information greatly helps in the debugging process.

Table 6. Easy Management of Configuration Parameters





5. Customization

NEFS provides full support for customization of frameworks to match your needs. This is done using standard delegation and inheritance mechanisms. Since you already have high quality packaged code and database controls (in the form of XML tags), it releases your team from the need to create error-prone low-level

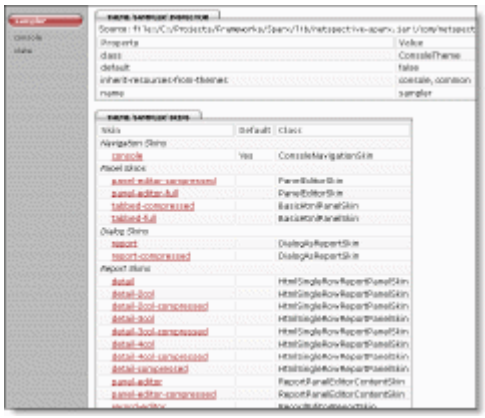
infrastructure code.

5.1. Support for Multiple Skins

NEFS separates form/report presentation from form/report design and logic by automatically creating all HTML and DHTML in user-defined *skin* objects. It provides several built-in skins to be used to build your applications. In addition to that, you can define your own skins and use them to customize your application's look-and-feel.

You can view the detailed design and functional description of these themes/skins through Console.

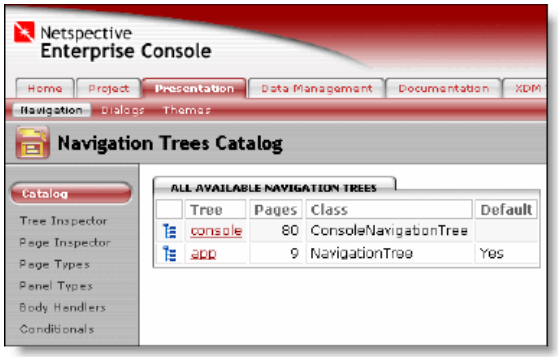
Table 7. View Design and Functional Specs of a Theme

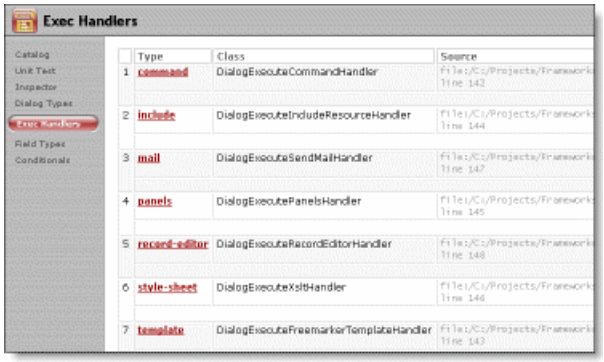


5.2. Custom Pages and Classes

You can define your own pages and handler classes to customize the framework according to your own needs. All the page definitions are available for analysis through the Console. Similarly, all the built-in and custom handlers are also available for viewing through the Console.

Table 8. Viewing Custom Pages and Handler Classes

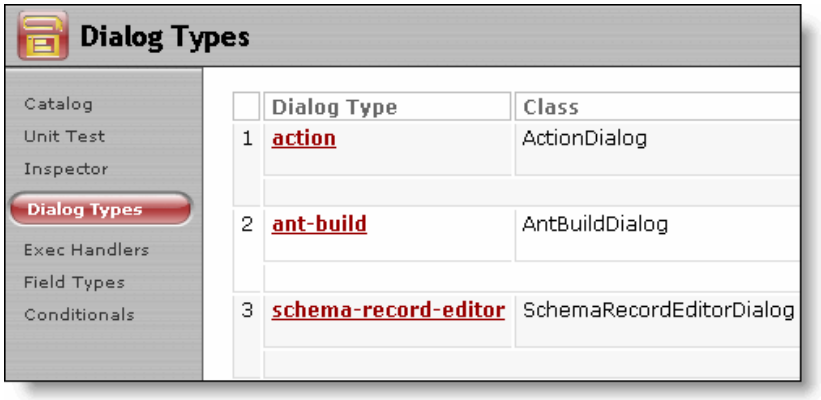




5.3. Custom Dialog Types and Field Types

A Sparx dialog is a container/manager object consisting of data fields that provide the flexibility to create customized forms for data processing. Sparx provides different types of dialogs to be used for different purposes. For example, Schema-Record-Editor dialogs are used for editing the records of a database table. You can also define your own dialog types which are automatically documented through the Console.

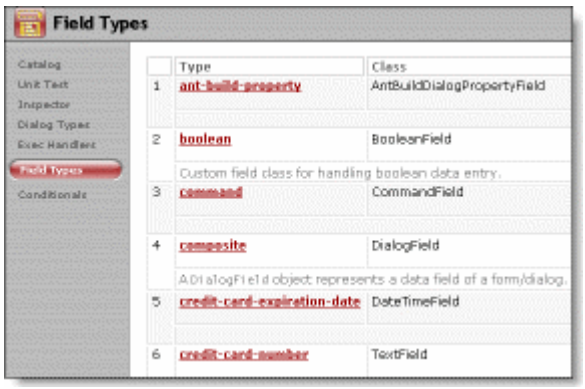
Figure 10. Customization through Dialog Types



They also provide the ability to create new fields or modify existing ones. These fields are similar to HTML fields but are far more powerful because they can format and validate themselves according to rules that you declare.

Sparx comes with lots of pre-defined field types. The Console always displays all of the available field types (including all built-in field types and any custom field types you register).

Figure 11. Customize Dialogs using Custom Field Types



6. Unit Testing

The NEFS allows you to increase the quality of your applications by running automated unit tests. While developers are working on forms and SQL statements, Console automatically provides browser-based testing of the forms and statements. No servlets, or JSPs need to be written for basic testing of forms, validations, and SQL statements. This also means that no compilation or configuration effort is required for testing your app components. This provides the ability to quickly test things "*in container*" without requiring any extra effort from the development team.

Note

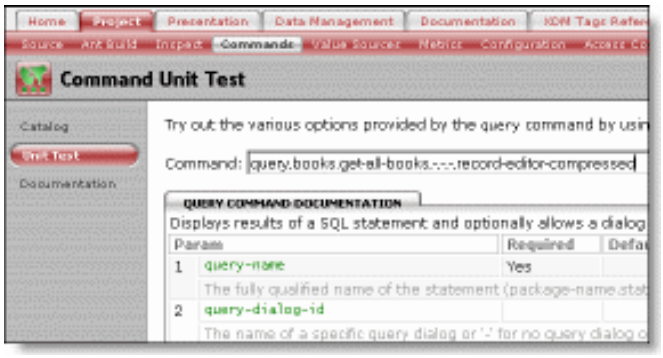
The testing is done using the same classloader, classpaths, and other container (app server) configurations as the ones used by the application. So problems related to the container are easier to find.

Once initial testing is completed and requirements are solidified, the forms and statements can be aggregated to create interactive applications. End users can use the interactive testing tools to see code as it is being developed (supporting eXtreme Programming concepts).

6.1. Commands

Each command may be unit tested inside the Console before using in the integrated application.

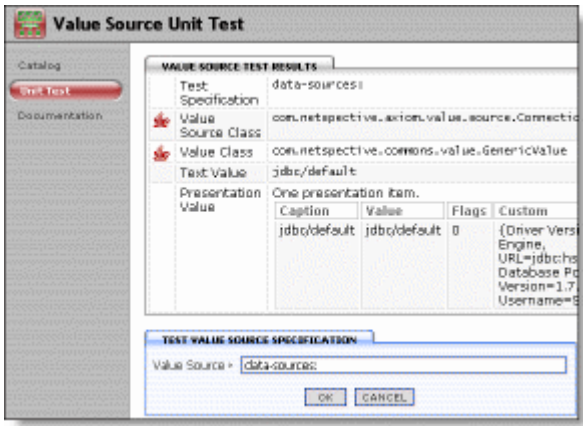
Figure 12. Automatically Generated Unit Test for a Built-in or Custom Command



6.2. Value Sources

Each value source may be unit tested inside the Console before using in the integrated application.

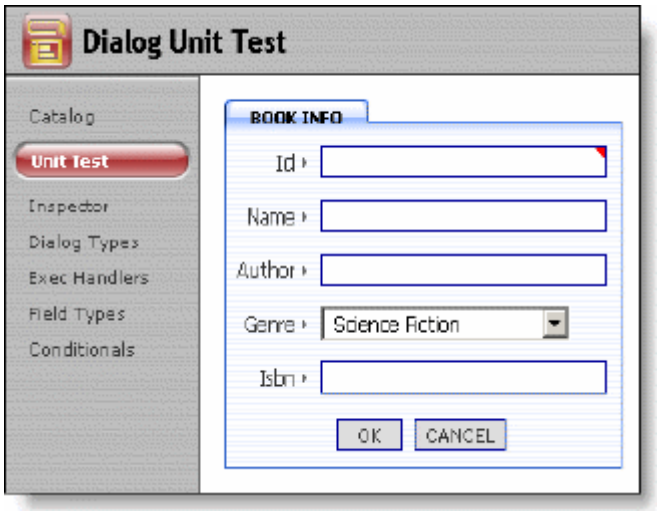
Figure 13. Automatically Generated Unit Test for a Built-in or Custom Value Source



6.3. Dialogs

Each dialog may be unit tested inside Console to ensure that user requirements are being met and allowing analysts or end users to start reviewing a running application quickly and without programmers writing test harnesses. As soon as the XML is completed, the test harness is automatically created and may be executed by anyone with access to Console.

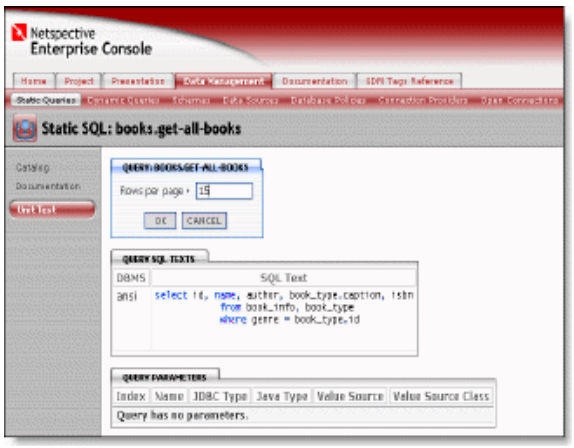
Figure 14. Automatically Generated Unit Test for a Dialog

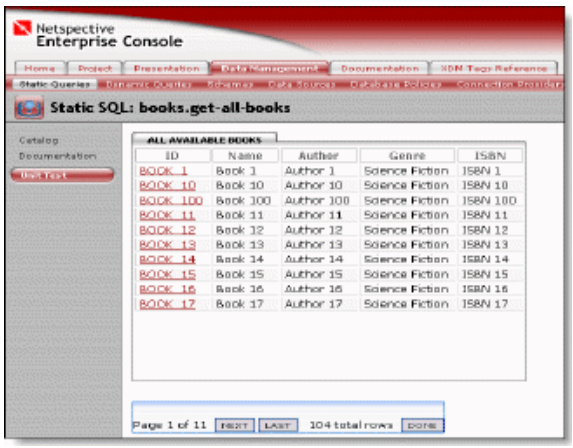


6.4. SQL Queries

Each SQL statement may be unit tested inside Console to ensure that user requirements are being met. As soon as the XML is completed, the test harness is automatically created and may be executed by anyone with access to Console. The statement, along with any reports/views can be tested without creating any additional code.

Table 9. Automatically Generate Unit Test Harness for SQL Queries

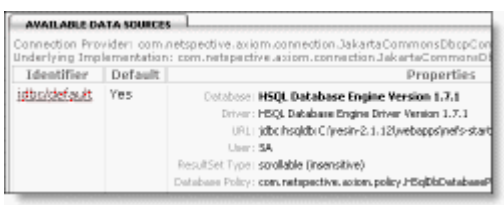




6.5. Data Sources

NEFS provides XML tags for definition of your data sources. You can either use the default data source or define your own data source. All the data sources, defined by the Axiom tag, are automatically documented and managed by the Console. You can view your data source(s) through this Data Sources Catalog.

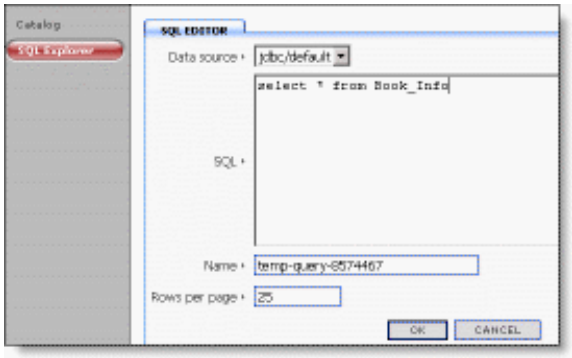
Figure 15. Data Sources Catalog

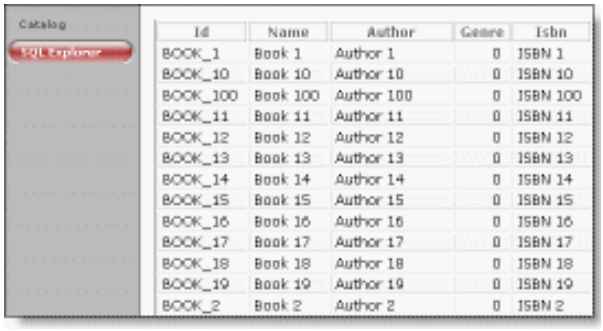


6.5.1. SQL Explorer

You can use the SQL Explorer within the Console to unit test tables within your schema. Now you do not require any other database management tool to manage your database tables. Your database programmers spend time working on the tables and schema elements significant to your application instead of writing test cases for them. Also, you can test your DB tables without using any specific database tool.

Table 10. Unit Test the Database Tables using SQL Explorer





Id	Name	Author	Genre	Isbn
BOOK_1	Book 1	Author 1	0	ISBN 1
BOOK_10	Book 10	Author 10	0	ISBN 10
BOOK_100	Book 100	Author 100	0	ISBN 100
BOOK_11	Book 11	Author 11	0	ISBN 11
BOOK_12	Book 12	Author 12	0	ISBN 12
BOOK_13	Book 13	Author 13	0	ISBN 13
BOOK_14	Book 14	Author 14	0	ISBN 14
BOOK_15	Book 15	Author 15	0	ISBN 15
BOOK_16	Book 16	Author 16	0	ISBN 16
BOOK_17	Book 17	Author 17	0	ISBN 17
BOOK_18	Book 18	Author 18	0	ISBN 18
BOOK_19	Book 19	Author 19	0	ISBN 19
BOOK_2	Book 2	Author 2	0	ISBN 2

7. Reusable Code

The centrally managed business rules library is accessible across projects. This reduces the time and effort required for application development. Dialogs (UI), field validation rules, conditional processing, all SQL statements, dynamic queries, configuration files, and many other resources are stored in XML files that are reusable across applications.

8. Managing Open Connections

When working with databases, one of the most common problems is that developers open connections but may forget to close them. The Open Connections page within the Console displays open database connections and their location in the code. Using this facility you can easily debug and remove the possible orphan database connections.

9. Security and Personalization

The NEFS provides a security and personalization layer which allows business users and developers to permeate restrictions on forms, reports, pages and other resources based on user names, types, location, roles, and capabilities. All of the application primary permissions, child permissions, aliased permissions, and roles are automatically documented in Console.

9.1. Access Control Lists

NEF has built-in support for building user authorization rules and declaratively controlling access to application components based on authorization rules such as roles and permissions.

10. Remote Development and Debugging

Because it uses thin-client for Console, it supports remote development and debugging. This is beneficial regardless of where the development team is physically located. This means that multiple developers in different cities can now review code across hundreds of miles using this thin-client web application.

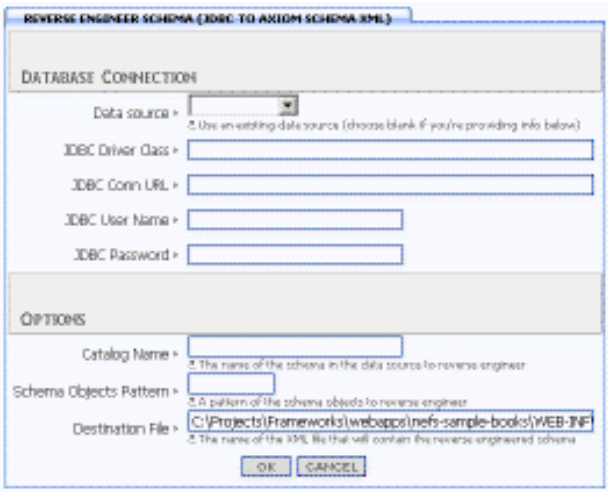
A developer in one city could easily demonstrate prototypes to a customer in another city with no extra work. All the team members can now collectively administer the framework and all the processes. At the same time,

managers can use the Console for tracking programmer work and productivity.

11. Reverse Engineering the Existing Database

The Console provides a facility to reverse engineer a schema from an existing JDBC data source into Axiom's schema XML. Using this facility, you can reverse engineer your existing databases and use them in your applications, without requiring any extra effort.

Figure 16. Reverse Engineer an Existing Database



12. Automated Code Generation

The XML declarations for schema definition can be used to generate database-specific SQL DDL allowing a single XML source schema to work in a variety of SQL relational databases (like Oracle, SQL Server, MySQL, etc.). Once declared, the schema descriptor supports completely automatic generation of SQL DML (insert/updates/removes), SQL DDL (create tables, objects, etc), and XML import/export.

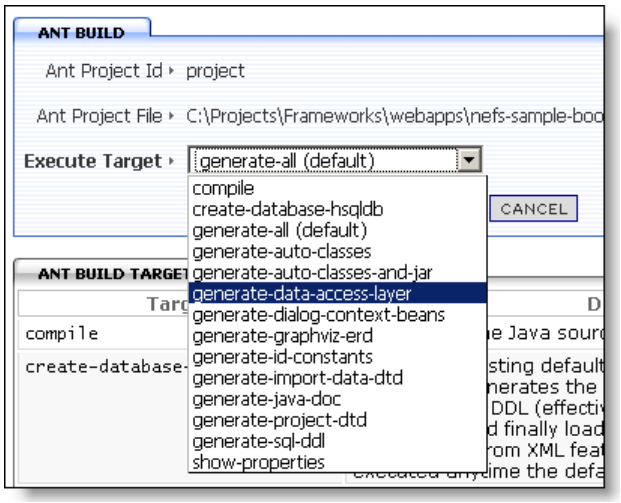
12.1. Data Access Layer (DAL) Generation

Sparx generates database-independent Java Object-relational classes for an entire schema, automating the majority of SQL calls by providing strongly-typed Java wrappers for all tables and columns. This is called the *Application DAL (Data Access Layer)*. The result is real database integration, with your existing and future applications, that costs less to maintain.

The entire schema becomes fully documented through the generation of JavaDoc documentation (the DAL generators generate JavaDoc comments automatically for all classes, members, and methods). This allows your database programmers to focus on customizing high quality packaged database controls.

Once you have a valid XML SchemaDoc, you can generate the DAL through the Console.

Figure 17. Generate Data Access Layer (DAL) Using Built-in Scripts



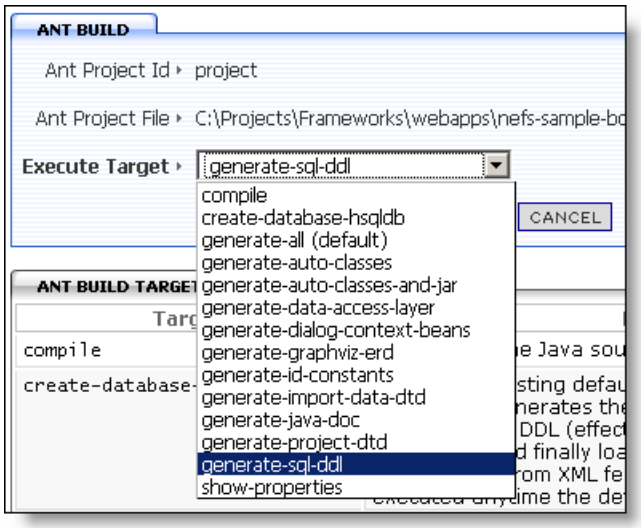
12.2. SQL DDL and DML Generation

Once declared, the schema descriptor supports completely automatic generation of SQL DDL (create tables, objects, etc) and SQL DML (insert, update, delete, etc.) for almost any relational SQL database. Database-specific SQL DDL is created by applying Database Policies to the <schema> tags.

Leverage your existing DBAs to work on advanced SQL functionality and performance while NEFS generates high quality DML automatically for non-DBAs. These packaged components also result in significantly reduced time to deployment of data-driven applications and web services.

Built-in Ant scripts are provided, through Console, to automatically create/update/remove DDL and DML commands.

Figure 18. Generate DDL Using Built-in Scripts



13. Version Control

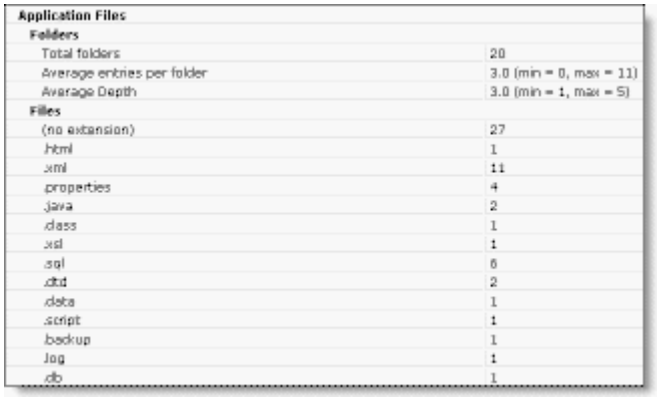
NEFS fully supports version control using your existing version control techniques (CVS, ClearCase, PVCS, etc.). Other frameworks with consoles prevent true revision control because they put components in a database *inside* their admin consoles and not in simple revisionable text files. In these frameworks, the configuration options are stored in a proprietary database where multiple versions are not visible. This does not allow the

developers to *roll back* their configuration changes. Since NEFS is specifically designed for large-scale applications, it stores everything in user-visible, user-editable text files which are easily managed by revision control.

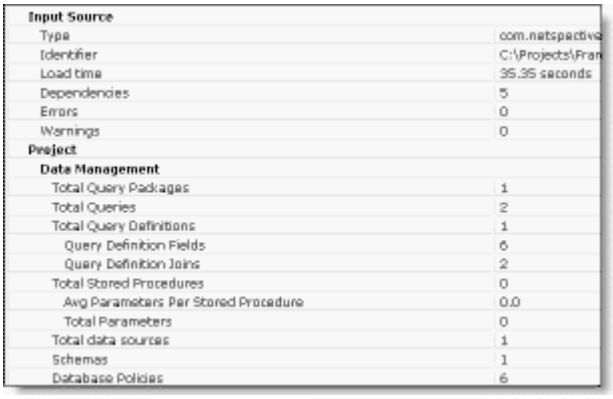
14. Project Metrics

As developers create forms, SQL statements, query definitions, JSP, servlets, and other code, the Console automatically maintains basic application metrics. Metrics are an important part of every sophisticated software development process and Sparx can not only capture the metrics but store them in XML files so that they can be analyzed over time. Your team concentrates on implementation of business functionality while the Console automatically collects all relevant project metrics, documentation and details.

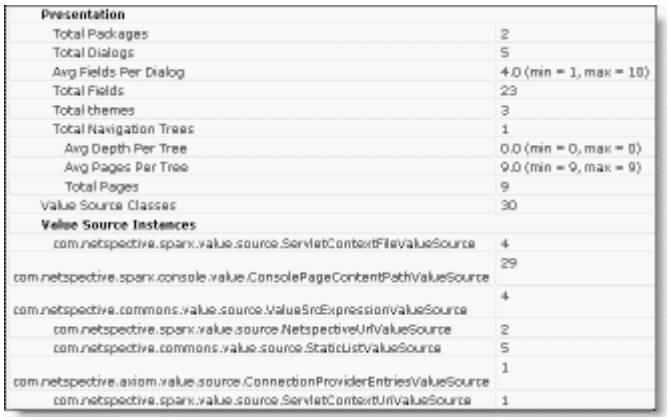
Table 11. View and Analyze Project Metrics



Application Files	
Folders	
Total folders	20
Average entries per folder	3.0 (min = 0, max = 11)
Average Depth	3.0 (min = 1, max = 5)
Files	
(no extension)	27
.html	1
.xml	11
.properties	4
.java	2
.class	1
.xsl	1
.sql	6
.dtd	2
.data	1
.script	1
.backup	1
.log	1
.db	1



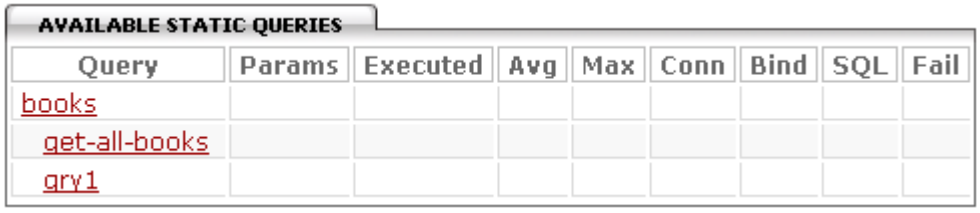
Input Source	
Type	com.netspective
Identifier	C:\Projects\Fran
Load time	35.35 seconds
Dependencies	5
Errors	0
Warnings	0
Project	
Data Management	
Total Query Packages	1
Total Queries	2
Total Query Definitions	1
Query Definition Fields	6
Query Definition Joins	2
Total Stored Procedures	0
Avg Parameters Per Stored Procedure	0.0
Total Parameters	0
Total data sources	1
Schemas	1
Database Policies	6



Presentation	
Total Packages	2
Total Dialogs	5
Avg Fields Per Dialog	4.0 (min = 1, max = 10)
Total Fields	23
Total themes	3
Total Navigation Trees	1
Avg Depth Per Tree	0.0 (min = 0, max = 0)
Avg Pages Per Tree	9.0 (min = 9, max = 9)
Total Pages	9
Value Source Classes	30
Value Source Instances	
com.netspective.sparx.value.source.ServletContextFileValueSource	4
com.netspective.sparx.console.value.ConsolePageContentPathValueSource	29
com.netspective.commons.value.source.ValueSrcExpressionValueSource	4
com.netspective.sparx.value.source.NetspectiveUrlValueSource	2
com.netspective.commons.value.source.StaticListValueSource	5
com.netspective.axiom.value.source.ConnectionProviderEntriesValueSource	1
com.netspective.sparx.value.source.ServletContextUrlValueSource	1

15. Application Performance and Logging

All mission-critical and sophisticated web applications need to be tuned for both database and application performance. Console tracks the log outputs for maintaining data about execution statistics for SQL statements, servlet and JSP pages, dialogs/forms, and security. This is used to automatically collect the project metrics.



AVAILABLE STATIC QUERIES

Query	Params	Executed	Avg	Max	Conn	Bind	SQL	Fail
books								
get-all-books								
qry1								

In addition to the actual SQL statements, performance metrics accompany each SQL statement indicating the

number of times each statement is execute and the average/maximum time (in milliseconds) it took to execute the statement.

16. Support Documentation

NEFS contains hundreds of pages of API and developer document which helps lower the learning curve for the development team. Console provides a centralized location for all project documentation for any application.

Document	Purpose
Getting Started with NEFS (HTML) (PDF)	Provides instructions for how to evaluate the Netspective Enterprise Frameworks Suite (*opens in a new window*).
NEFS User's Manual	Provides instructions for how to use the Netspective Enterprise Frameworks Suite (*opens in a new window*).
NEFS Upgrade Guide	Provides instructions for how to upgrade one or more of our Java Frameworks.
NEFS Changes Log	Provides a history of the changes to NEFS since 7.0 was released.
NEFS Introduction Presentation	A PowerPoint presentation that introduces the main features and functions of the NEFS.
NEFS Sampler App Tutorial (HTML) (PDF)	Provides a tour of NEFS Sampler Application.
Books Sample App Tutorial (HTML) (PDF)	Provides step by step instructions for how to build the Books sample application using NEFS.
Old developer.netspective.com site	In case you need access to the old site, the contents remain available.

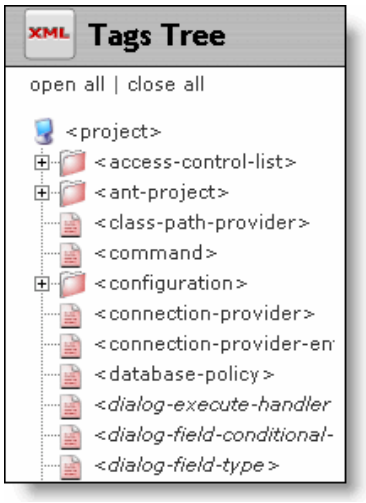
16.1. XDM Tags Reference

Instead of storing application code and programmer documentation separately, Console brings tag documentation, javadocs, and other project documents into a single easily accessible place. Managers no longer need to hunt for documents. It also lowers the learning curve for the developers.

Note

Note that any classes that are extended by customers are also documented through the console automatically.

Table 12. NEFS Comprehensive XDM Tags Reference



17. Accessing the Console in Applications

Each application has a private instance of the Console using the http://*server*/*appName*/console pattern. When you log into the Console for Application X (appX/console) versus Y (appY/console) you will only see components for the appropriate application.

The Console is an optional component for every application built with NEF and it is turned on by default. You may decide to turn it off completely for your applications or secure it differently.

Here are some examples of how to access the Console for some of the sample applications.

Sampler Application	The Sampler App demonstrates various elements of NEFS. Its application identifier is `nefs-sampler`. The URL for Sampler App is http://www.netspective.com/nefs-sampler and its Console URL is http://www.netspective.com/nefs-sampler/console. Note that the Console is accessed simply by adding the /console path at the end of the application's URL
Sample Books Application	The Books sample application is the sample application that demonstrates how to create a database application and its application identifier is `nefs-sample-books`. The URL for Books App is http://www.netspective.com/nefs-sample-books and its Console URL is http://www.netspective.com/nefs-sample-books/console. Note that the Console is accessed simply by adding the `/console` path at the end of the application's URL.
Your Application	When writing your own application you will simply append `/console` to the end of your own application's context identifier. If your app is available at http://*your-server*/*your-app-id* then the Console for your application would be available at http://*your-server*/*your-app-id*/console.

18. Login to Enterprise Console

The Console's default user name is '`console`' and the default password is '`console`' (each without quotes) . Unless otherwise specified, that is the user name and password combination you should use if the Console prompts you to login.

Figure 19. Console Login

To save your Console User ID on the computer, select the checkbox provided with Remember my ID on this computer option. The "Remember my login" ability allows users to store encrypted cookies and only login once.