

*MainTrend Ltd.*

E-mail: [info@maintrend.net](mailto:info@maintrend.net)  
Site: [www.maintrend.net](http://www.maintrend.net)

---



# PowerBuilder to Java (PB2J) Thin Client J2EE Migration

*White Paper*

**December 2008**

---

Copyright © 2001–2008 MainTrend Ltd. All rights reserved.

The information contained herein may not be redistributed, sold or modified or used to create in any derivative work without the prior written consent of MainTrend Ltd.

# *MainTrend Ltd.*

E-mail: [info@maintrend.net](mailto:info@maintrend.net)  
Site: [www.maintrend.net](http://www.maintrend.net)

---



## **Contents**

Executive Summary	3
About MainTrend	4
Introduction to PowerBuilder Migration	5
Post Conversion Environment	6
Migration Methodology	7
Migration Process and Solution Architecture	9
Conversion Services and Pricing Model	13
Appendix A: JDataPanel Framework	14

---

Copyright © 2001–2008 MainTrend Ltd. All rights reserved.

The information contained herein may not be redistributed, sold or modified or used to create in any derivative work without the prior written consent of MainTrend Ltd.



## Executive Summary

This document describes MainTrend's solution for successful PowerBuilder to Java Migration.

MainTrend offers an automated conversion service that lifts PowerBuilder applications to Java/XML technologies while preserving business logic and user interface investments.

The new architecture is a pure thin client ("browser") solution with a J2EE/XML application and web server on the server side.

The solution is provided for the automated conversion of PowerBuilder applications (with some minimal manual intervention) in a way that:

- Maintains and saves all the existing investment in business knowledge and logic.
- Removes the high costs and risks of a complete manual application rewrite.
- Provides the quick ROI, in that both migration and post migration costs are minimized.
- Opens the way to further development and maintenance in a new modern J2EE thin-client environment.
- Imposes no limitations for the target server-side platform: any Java-enabled platform is supported.

Organizations that have already moved applications, or who are investigating such a move, have had two options:

- To manually rewrite their PowerBuilder applications.
- To automate the conversion using tools currently available on the market.

Each replacement methodology approach has its advantages and disadvantages.

PowerBuilder as a migration source is the ideal case for **a mixed approach**. To this end, MainTrend's solution is to automate the conversion of PowerBuilder applications, while at the same time enabling and facilitating easy manual intervention (where required) to the business logic code.

Most of the work can be done remotely. The manual part can be divided between MainTrend and the customer or a third party conversion partner.

Working with MainTrend as the conversion partner frees organizations up from the translation process so they can continue to do what they do best — remain dedicated to running their core business functions instead of dedicating themselves to one usually lengthy migration project.

# *MainTrend Ltd.*

E-mail: [info@maintrend.net](mailto:info@maintrend.net)  
Site: [www.maintrend.net](http://www.maintrend.net)

---



## **About MainTrend**

Founded in 2002, MainTrend is a company that unites high-level information technology and organization architecture professionals. MainTrend is a software and solution provider, focused on integrative decisions for its customers.

MainTrend's professionals work in government structures, banks, private corporations and software development organizations; MainTrend know the needs of these types of organizations and is able to both supply integrative solutions and assist in their decision making processes.

MainTrend's professionals have extensive expertise in system architecture design, databases, application programming and decision making systems.

MainTrend is proud of its ability to perform complex projects, meeting planed budget, technology and time goals.

By working with the MainTrend's team of specialists, organizations receive the added benefit of applying MainTrend's experience and advanced expertise to their unique and often complex requirements.

---

Copyright © 2001–2008 MainTrend Ltd. All rights reserved.

The information contained herein may not be redistributed, sold or modified or used to create in any derivative work without the prior written consent of MainTrend Ltd.



## Introduction to PowerBuilder Migration

For many years PowerBuilder was one of the most serious and robust Rapid Application Development tools for Client/Server and n-tier development. Reliable high-performance PowerBuilder applications are deployed worldwide. Software firms invested in PowerBuilder frameworks, resulting in billions of PowerBuilder code lines written by thousands of trained developers. Unfortunately, PowerBuilder is now a niche player in the application development tool market. Even PowerBuilder's attempts to make inroads in the Web development market have not been very successful. The number of new installations is diminishing. PowerBuilder professional personnel are becoming scarce.

Enterprises with deployed active PowerBuilder applications need to both maintain their current systems, and provide solutions for dynamic changes and enhancements. Enterprises are at a point where they have to make a strategic choice: to continue building and updating applications in PowerBuilder where there is no future, or to move to another platform.

Remaining with PowerBuilder causes no upheavals but insures the widening of the legacy gap.

Moving to another platform is probably strategically correct but has serious implications.

- How to prevent the loss of the investment already made in the legacy applications
- How to reduce the significant cost of new application development.
- How to ensure a rapid and smooth migration from one infrastructure to another without disturbing critical systems; specifically how to integrate the new with the old.

More and more, organizations are being urged to move their PowerBuilder applications to a modern thin-client environment.

There are 3 main options:

- Face lifting (a strategy provided with the new PowerBuilder versions)
- Manual rewriting from scratch
- Automated conversion.

Face lifting uses the existing PowerBuilder application as a back-end, and therefore does not really move it to a modern environment.

Rewriting PowerBuilder applications from scratch is a very costly, time consuming process and may lead to a long learning curve for the end-users. Another significant point is that legacy applications and maintenance changes are not always well documented. Even when original system analysis documents exist, the risk of business knowledge loss is very high.

Automated conversion (with some minimal manual intervention) as a replacement methodology saves all the investment in business knowledge and enables further development and maintenance in the new modern environment.

This document presents the migration process suggested by MainTrend in order to meet all aspects and challenges of migrating PowerBuilder applications to full J2EE thin-client Java applications.



## Post Migration Environment

The migration target is a full thin client environment based on J2EE and AJAX methodologies.

The new architecture is a pure thin client ("browser") solution with a J2EE container and web server on the server side; the application is packed as a J2EE servlet running within a J2EE container; database access and some business logic are provided via J2EE components. All the data access layer (DAL) and presentation layer (GUI) object definition XML files may also be stored on the central web server.

The server side can be any J2EE container / Java Application Server, running on any Java supporting operating system (IBM WebSphere, BEA WebLogic, Oracle Application Server, Tomcat, JBoss, Sybase EAServer, etc.).

The resulting application requires no client installation, except for a browser, and no operating system limitations. Currently supported browsers are Internet Explorer 6.0 and higher, and Firefox 3.0 and higher. The same browser software version has to be installed on all the client workstations. All the browser instances on the client workstations should have the same settings (security, JavaScript etc.).

This architecture provides centralized application management and does not require any additional software to be installed on the end-user workstations:

- Application changes that are made to the XML definition files (database access definitions, GUI definitions, etc.) are available immediately.
- Business logic changes, encapsulated within application server components, are available immediately after installation.
- Business logic and GUI framework changes, encapsulated within the J2EE Servlet, are available immediately after the Servlet container restart.
- The database is accessed from the application server components.



## Migration Methodology

Migration to another platform requires a number of factors to be considered:

- Existing investment in legacy applications and the significant cost of new application development.
- Ensuring a rapid and smooth migration path from one infrastructure to another without disturbing critical production systems.
- Smooth and reliable transformation preserving all of the needed business logic. *Maintaining legacy software business logic is the important part of an organization's knowledge storage. The more valuable the encapsulation of the business logic software is, the more complex the process of its evolution.*

More and more organizations are looking to move their PowerBuilder applications away from the thick client, proprietary environment provided via PowerBuilder, into the n-tier thin client environment provided by Java. In order to accomplish this, some organizations have attempted to manually rewrite their PowerBuilder applications in Java. Others have used conversion tools currently available on the market.

Organizations that have already moved applications, or who are investigating such a move, have had two options:

- To manually rewrite their PowerBuilder applications.
- To use conversion tools.

Each replacement methodology approach has its advantages and disadvantages.

- Manually rewriting can produce very effective resulting code, which may exactly match the target environment. On the other hand, it is a very costly and time-consuming process and may lead to a long learning curve for the end users and for the maintenance team. In addition, the risk of business knowledge loss is very high, in that legacy applications and the changes made to them are not always well documented.
- Automatic conversion is much faster, much cheaper, and also ensures that all the business knowledge is preserved. The resulting application is similar to the original one, so the learning curve for the end users and maintenance team is not as long as with a "rewritten" application. On the other hand, the resulting code is generated automatically. It may not necessarily be optimal for the target environment and for the specific application.

PowerBuilder as a migration source is the ideal case for a **mixed approach**.

A central element in the PowerBuilder environment is DataWindow, a very powerful object, concentrating almost all the data processing of a PowerBuilder application and almost all the visual representation of the application's data. The PowerBuilder DataWindow object is one of the main reasons for the success that PowerBuilder has achieved as a software development tool.

# MainTrend Ltd.

E-mail: [info@maintrend.net](mailto:info@maintrend.net)  
Site: [www.maintrend.net](http://www.maintrend.net)

---



Usually in a PowerBuilder application DataWindow objects consume about 60% of the application building efforts.

MainTrend approach to the conversion is to include DataWindow objects in the automatic part of the conversion, while making all the other PowerBuilder objects' conversion as flexible as possible to allow manual rewriting of the objects' methods.

Such a migration concept unites advantages of both approaches:

- The original application is automatically reversed engineered.
- All the DataWindow definitions are automatically converted to JDataPanel XML definitions and corresponding Java classes (JDataPanel framework is MainTrend's product for the J2EE environment; it is described in details in Appendix A).
- All the other PowerBuilder objects are parsed and converted to corresponding Java classes.
- All the Java classes are generated in an exact match with the original inheritance tree.
- All the attributes and method signatures of the original classes are automatically converted.
- The bodies of all the methods in the generated Java classes are empty, with the original PowerScript code included as a comment. This allows for easier manual rewriting of these scripts and ensures effective resulting code, which exactly matches the target environment. This approach also prevents business knowledge from being accidentally dropped or omitted.

PowerBuilder to Java (PB2J) is MainTrend's software engine toolset and set of methodologies that take an application coded in PowerBuilder and generates Java / Xml application source code with a minimal need for a programmer intervention. Java as a target environment allows maximum platform flexibility. Java has proven itself to be a superior technology as a modern business application tool for highly reliable applications.

This migration methodology provides:

- Rapid, accurate and valid way to migrate PowerBuilder applications to Java.
- Low assimilation costs for the new application
- The same productivity and more, in a new modern environment.
- No limitations for the target server-side platform: any Java-supporting platform will fit
- Much lower cost and faster solution for the legacy applications' migration path.
- Flexible open-platform development environment. The applications can be further developed by our GUI designing tools and framework alongside and integrated with the most cutting-edge development tools
- Code reuse enabled environment and reduced maintenance costs.
- Seamless integration with modern technologies and new approaches.

---

Copyright © 2001–2008 MainTrend Ltd. All rights reserved.

The information contained herein may not be redistributed, sold or modified or used to create in any derivative work without the prior written consent of MainTrend Ltd.



## Migration Process and Solution Architecture

Generally the migration process has the following steps:

1. Detailed analysis. Selecting the target server-side platform
2. Verification and fixing of the source application
3. Export of all the PowerBuilder objects of the application to be migrated
4. Reverse engineering of the original application
5. Code generation
6. Manual rewriting of the application classes' method's code
7. Preparing test environment
8. Database migration (if a new database platform is chosen)
9. Unit testing, including required database connection for the data access layer objects
10. External links migration (if any)
11. Code integration and thin platform adaptation. Application restructuring required for the thin client conversion model.
12. Database integration and changes required for the chosen database platform
13. General graphical design; fine tuning of the resulting GUI in line with customer standards
14. Application integration testing and corrections
15. User acceptance
16. Building the production environment and implementing the new application in production
17. Trainings for the customer's developers

The [conversion](#) itself is covered by steps 3 through 6 and uses PowerBuilder Conversion Suite.

PowerBuilder Conversion Suite consists of these parts:

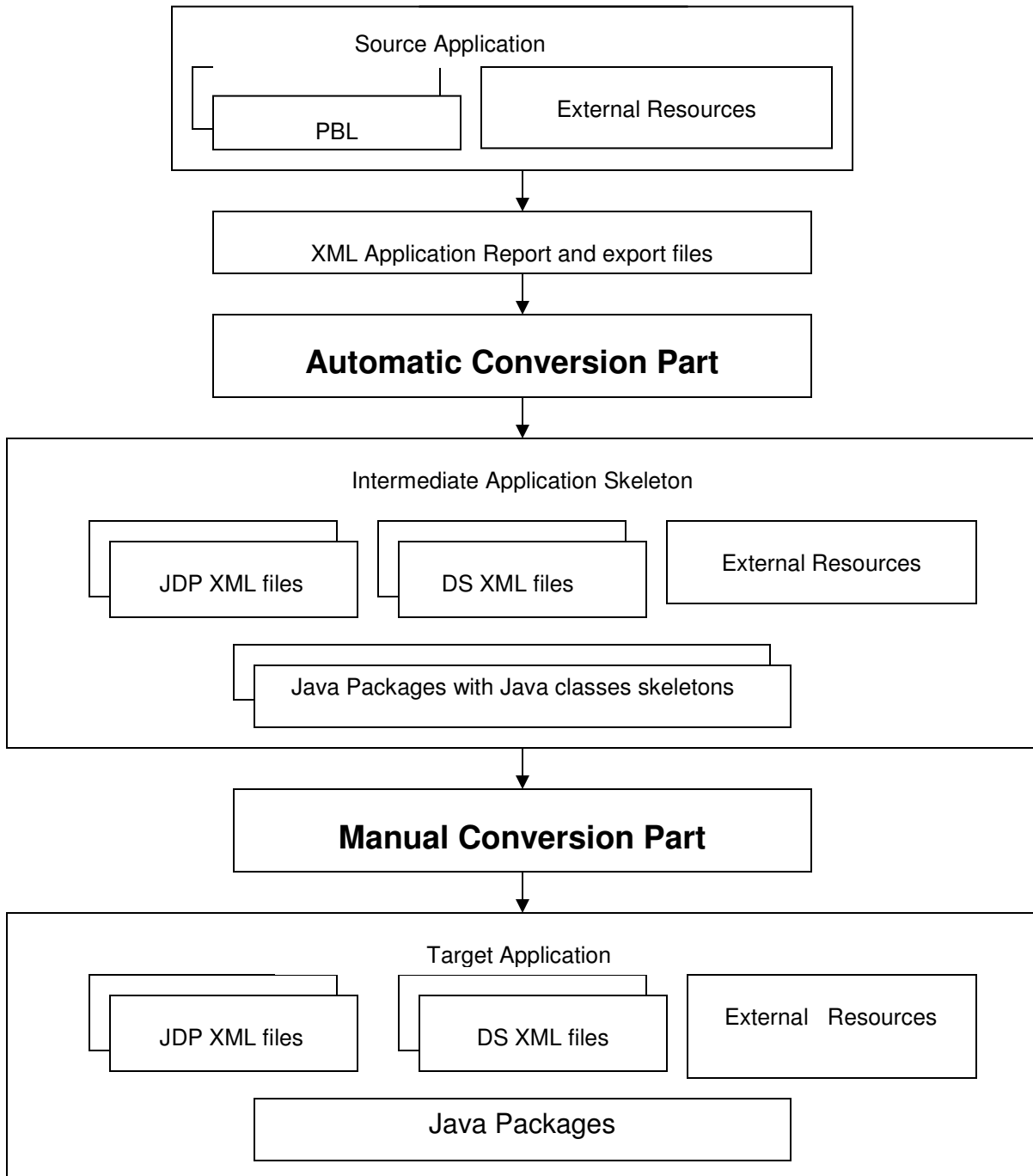
- J2EE-based JDataPanel framework that allows future flexible and reliable development in the new Java-based environment, and which also includes an independent product for rich Form/Report design – MainTrend's JDataPanel Designer. The framework supplies all the needed functionality that PowerBuilder DataWindow objects have, including database access, forms and reports. JDataPanel can be also used to create new Java applications, not just to convert existing PowerBuilder applications.
- The PowerBuilder Converter itself, which translates each PowerBuilder application into a set of Java and XML modules.
- Conversion and development methodologies, including best practices for the most effective manual completion and enhancement of the resulting applications.

We use the PowerBuilder “library export” functionality to obtain all the sources of PowerBuilder objects for the specific PowerBuilder library set. After the reverse engineering stage is the code generation stage. All the DataWindow definitions are automatically converted to JDataPanel XML definitions and corresponding Java classes. All the other PowerBuilder objects are parsed and converted to corresponding Java classes. The bodies of all the methods in the generated Java classes are empty with the original PowerScript code included as a comment. These methods are then rewritten in the manual completion stage, which also includes GUI tuning (based on the customer's requirements) and database access tuning.



The PB2J toolset uses a sophisticated concept of multi-level embedded parsers and template-based code generators. This concept allows use of different parsers for different parts of the scripts and objects' types, and improves the conversion performance.

Visually the conversion process can be depicted in this manner:



# MainTrend Ltd.

E-mail: [info@maintrend.net](mailto:info@maintrend.net)  
Site: [www.maintrend.net](http://www.maintrend.net)



The generated objects are grouped into the following packages:

- “dal” – XML-definitions of the data access layer objects (DataStores, the database-related part of DataWindow objects)
- “app” – all the Java classes
- “presentation” – all the XML-definitions for the GUI objects (GUI-related part of DataWindow objects)

PowerBuilder's DataWindow object unites in one object two layers – presentation layer and data access layer. It was tolerable for rapid application development concept, but does not meet modern ideology of layers separation. Therefore during the conversion each DataWindow object is converted into three JDP framework objects: a DataStore XML definition (data access layer object), a JDataPanel XML definition (presentation layer object), and a Java class for embedded logic. Thus the more close correspondence to n-tier requirements is achieved.

The presentation layer object definitions and data access layer object definitions (XML files) can be stored with the resulting Java application, or can be deployed on a web server.

The generated application is deployed with the “thin” framework as follows:

```
[webapps]
  [... the application's directory
    [css]
    [html]
    [images]
    [js]
    [jsp]
    [META-INF
      MANIFEST.MF
    ]
    [WEB-INF
      [classes
        [The application's Java classes / XML files by package name:
          [dal]
          [app]
          [presentation]
        ]
        [net]
        [maintrend - MainTrend's engine Java classes]
      ]
      [lib
        [all the needed libraries, including Oracle JDBC driver]
      ]
      web.xml
    ]
    index.jsp
    login.jsp
    Configuration.xml
  ]
]
```

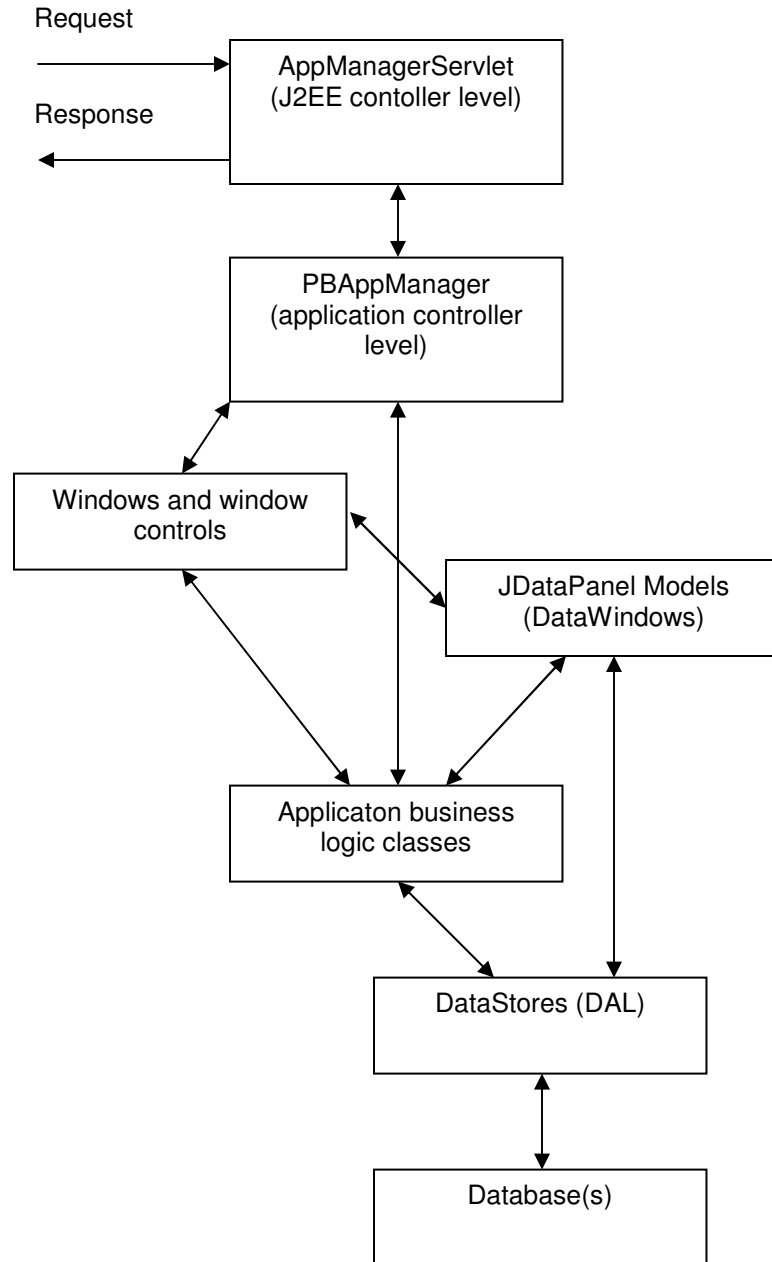
---

Copyright © 2001–2008 MainTrend Ltd. All rights reserved.

The information contained herein may not be redistributed, sold or modified or used to create in any derivative work without the prior written consent of MainTrend Ltd.



Logically the converted application works within a J2EE container as a servlet, which provides all the needed for the thin client environment functionality:



# *MainTrend Ltd.*

E-mail: [info@maintrend.net](mailto:info@maintrend.net)  
Site: [www.maintrend.net](http://www.maintrend.net)

---



## **Conversion Services and Pricing Model**

MainTrend offers an automated conversion service that lifts PowerBuilder applications to Java/XML technologies while preserving the business logic base and user interface investments.

Most of the work can be done remotely. This reduces or eliminates most of the costs associated with travel to and accommodation at the customer location.

The manual part can be divided between MainTrend and the customer or a third party conversion partner.

Varying options can be applied to the conversion projects, from full conversion to different "reverse engineering" stages and automatic generation of scripts to be used by customer's developers as a tool for business logic capture, etc.

The business model can also vary, from fixed price projects to hourly based consulting. A mix of type models for different stages of the conversion is also possible.



## Appendix A: JDataPanel (JDP) Framework

JDataPanel Framework is a central element of MainTrend's programming environment. It is both a base for all the conversions (Rally-to-Java, PowerBuilder-to-Java etc.), and also the independent framework for building J2EE applications. When developed, the purpose of the framework was not to write a PowerBuilder engine using Java as a programming language, rather to build a general framework that included PowerBuilder functionality and PowerBuilder's DataWindow as a special case. The purpose was to give Java developers a powerful and flexible Java and XML based environment for building Data Access Layer and Presentation Layer components.

JDataPanel Framework consists of the following parts:

- JDataPanel components ( Presentation layer )
- DataManager components ( Data access layer )
- Application manager components ( Control layer )
- Graphical designers ( JDataPanel Builder and DataStore Builder )

JDataPanel is a universal XML-based user interface description:

- It has both "wide" and "deep" architectures: correlates with intuitive concept of object with attributes ("wide") and sub-objects ("deep").
- It has built-in integration with the Business Logic and Data Access Layers together with readiness for layers separation.
- It is a powerful form / report framework.

Here is a User Interface Description Structure:

```
<?xml version="1.0" encoding="utf-8"?>
<group name="Main" ... (other attributes) >
  <textgroup name="t1" fontfamily="Courier New" x="0" ...>
    <text name="t1_1" container="t1" ...>Hello!</text>
  </textgroup>
  <group name="Header" datasource="DS1" ...>
    <text .../>
    <field name="date" datafield="indate" datatype="Date" ...>
      <editcontrol name="JDateField".../>
    </field>
    <group ...>
      <field .../>
      <text .../>
    </group>
  </group>
</group>
```



JDP framework supports an MVC (Model – View – Controller) design pattern even in thin client environment (HtmlRenderer + browser as “View”, JDP Model as “Model” and JDP commands as “Controller”).

DataStore part of the framework is used as a “Data Model” part of the “Model” in MVC pattern. DataStore framework is also a universal data access layer solution, and can be used independently from JDataPanel part of the JDP framework.

DataStore objects have their xml representation, which looks as follows:

```
<?xml version="1.0" encoding="utf-8"?>

<datastore name="EXAMPLE_DSD" clearOnRetrieve="true" comment=""
deleteAllowed="true" insertAllowed="true" retrieveLimit="0"
trimEndOnRetrieve="false" updateAllowed="true" updateFields="CODE,
DESCRIPTION" updateKey="CODE" updateSourceName="EXAMPLE_TBL"
definition="select EXAMPLE_TBL.CODE, EXAMPLE_TBL.DESCRPTION from
EXAMPLE_TBL EXAMPLE _TBL order by EXAMPLE_TBL.CODE asc" adapter="1">

  <field name="CODE" dataType="String" sourceAlias="EXAMPLE_TBL"
sourceFieldName="CODE" sourceName="EXAMPLE_TBL" updateable="true"/>

  <field name="DESCRIPTION" dataType="String"
sourceAlias="EXAMPLE_TBL" sourceFieldName="DESCRIPTION"
sourceName="EXAMPLE_TBL" updateable="true"/>

</datastore>
```

Flexibility of the DataStore framework is based on the encapsulation of the data access functionality with the “adapters” concept. Each type of the data storage can be accessed with the appropriate adapter, for example:

- SQLAdapter -> JDBC -> Database
- XMLAdapter -> XML files
- InMemoryAdapter

The adapters’ concept is open and any needed additional adapter can be easily developed.

Application manager components are the base for the building thin client solutions for converted from different sources applications, or for the applications which are built from scratch with JDP framework.

These components can be easily inherited and extended to suite all the needs of the chosen target platform.

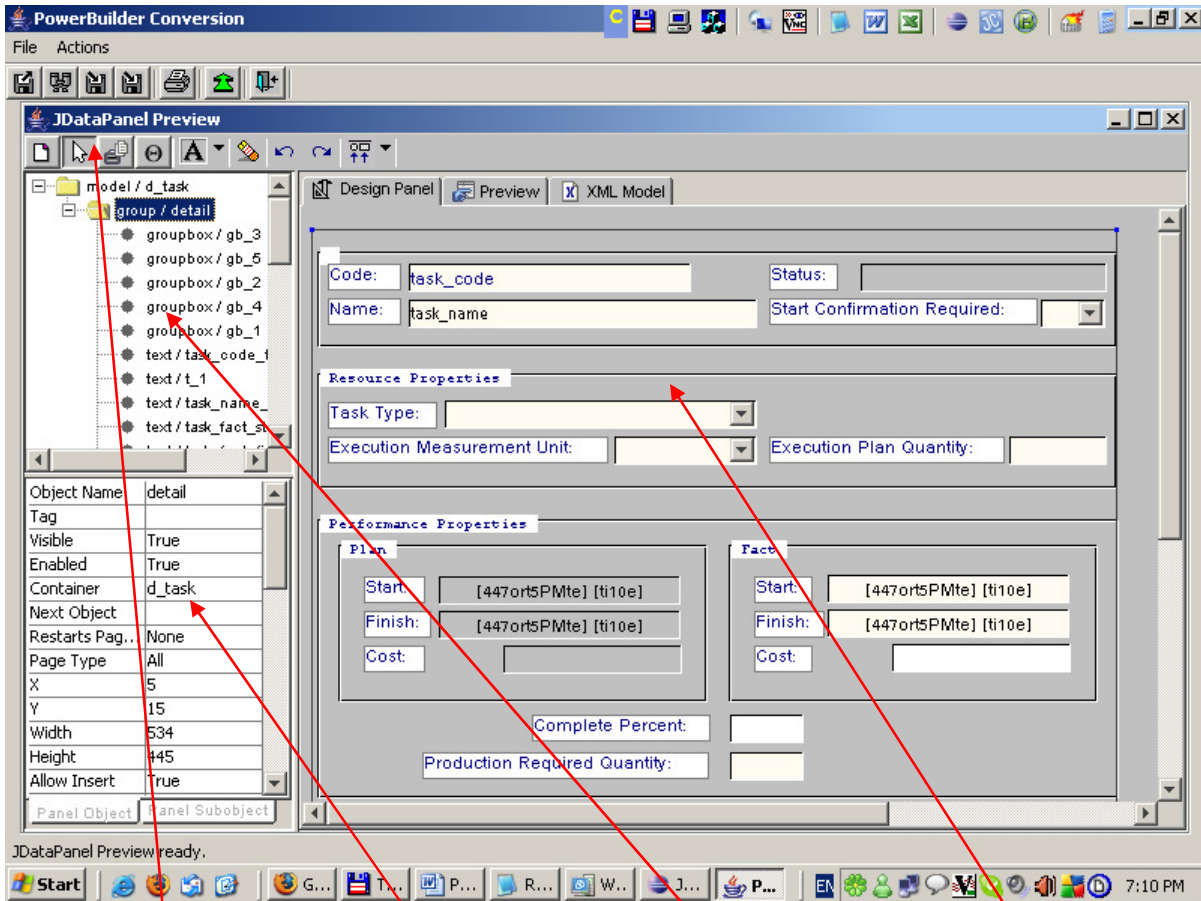
# MainTrend Ltd.

E-mail: [info@maintrend.net](mailto:info@maintrend.net)  
Site: [www.maintrend.net](http://www.maintrend.net)



To increase the development productivity and to help the developers in working with JDP framework, the Graphical designers (JDataPanel Builder and DataStore Builder) can be used.

Here is the example of the JDataPanel (form), opened within JDataPanel Builder:



Toolbar Area

Properties Panel

Object Hierarchy

Preview Area

Copyright © 2001–2008 MainTrend Ltd. All rights reserved.

The information contained herein may not be redistributed, sold or modified or used to create in any derivative work without the prior written consent of MainTrend Ltd.