zen Platform technical white paper

The *zen Platform* as Strategic Business Platform

The increasing use of application servers as standard paradigm for the development of business critical applications meant a great step towards the effective use of modern server-based software solutions. This more or less establishes a standard for the complexity of the planning and development of such software, but it does not reduce the complexity sufficiently enough in order to work effectively on the mostly technical interfaces.

The indeed difficult decision in favour of a specific application server therefore only defines the technical basis for a solution-oriented application development. It is not the first milestone in the solution of the actual business objective, but only defines the methodological background. The decision regarding the architecture of an application, its flexibility with respect to extensions, its scalability and thus its future usability, has to be made by experienced architects of Java 2 Platform, Enterprise Edition (J2EE). The basic framework and the necessary means for a structured solution of the business objective therefore have to be provided to the application developers in the form of time-consuming and costly new developments.

Since valuable expert resources are drawn on even before the actual objective can be addressed, the factors of time and cost are adversely affected as well. The period from the beginning of the project to the testing of the first functional prototypes is unnecessarily prolonged. And not only in the more dynamic fields of business has the original nature of the underlying objective changed in the meantime. Often, this results in the failure to meet the initial objective.

The *zen Platform* of zeos informatics GmbH closes this extensive gap in many respects. As middleware in the application server it encapsulates the complex technical interfaces and provides a simplified and flexible abstraction layer on which developers can quickly get to work. The development of the business logic on the *zen Platform* takes place on a very high level of abstraction, releasing the developer from all tasks which are not primarily part of the objective, and which on the other hand helps him to meet the objective by providing him with a large number of effective methods. So from the very beginning, the developer is able to concentrate on the business objective without having any expert knowledge himself. And the high level of abstraction does not in any way limit the expert in the use of the J2EE tools.

Many basic concepts which are either missing or only partially realized on the application server, such as dynamic application flexibility, rapid prototyping or localization (national language support), are additionally provided by *zen*.

The *zen Platform* provides technologies to quickly and efficiently develop or extend prototypes and new web front ends, web services and applications with proprietary interfaces, offering you many benefits by:

- **Visual Programming**
  Business logic for the *zen Platform* can be developed on the *zen Developer* or any other favoured environment. The application is then modelled on the graphic user interface of the *zen Developer* and linked with the business logic.

- **Rapid Prototyping**
  Any application can be visualized and executed already during its design phase, offering you the possibility to talk to partners, customers or competent departments in order to discover deficiencies at an early stage.

- **Increasing productivity in all development phases**
  Visual programming, the abstraction from the technical details of the system and the extensive support in achieving the business

objective tremendously accelerate the development process. Even beginners can become productive very quickly because their training is simplified. Comprehensive experience in architecture and design is no longer necessary. In addition to that, the visual debugger supports all development phases.

- **Flexible and Dynamic Architecture**
  At various technical and logical, clearly defined open interfaces a *zen* application can be extended at will. The business and process logic can be modified during runtime.

- **Scalability**
  Every *zen* application can be scaled without any modification from a simple servlet engine to a cluster of application servers.

- **Easy Administration**
  The management console allows you to centrally control, monitor and reconfigure even distributed applications.

- **Integration**
  Through its flexibility, the *zen Platform* can be used as centralized integration platform supporting any kind of application and front end. The system's inherent support of any type of XML processing considerably facilitates the connection of *zen* applications to third-party or legacy systems.

The high degree of abstraction of the *zen Platform* and the clear structure of the application code this entails, as well as the graphic display and the visual debugging turn any extension into simplicity itself. Checking for errors and training employees is also made extremely easy by the structure of the system.

Architecture and Performance Features of the *zen Platform*

The *zen Platform* provides the developer with a variety of tools on different levels which support him in the efficient development of applications. Irrespective of the type of application, be it web front ends, web services or applications with any other type of interface, the same automated data validation methods and the same methods of executing application-specific business logic will be employed. For the implementation the developer is free to use any of a number of services the *zen Platform* has to offer. All applications can be configured, monitored and even extended or adjusted while in operation using the centralized management console.

Every *zen* application is modelled in a repository on the graphic user interface of the *zen Developer*. The application's business logic is as usually programmed in Java. In the *zen Developer*, following the creation of the application workflow and the modelling of the data structure, developers define at which workflow interfaces the business logic is to be executed. Then each part of the business logic is linked with the corresponding modelled data. The thus modelled *zen* application is executed by the *zen Engine* during runtime. The business logic is automatically integrated, receiving the appropriate data from the data model during the execution.

Different from the customary approach today, with *zen*, the model of the application is always consistent with the actual implementation. The application is exactly documented by the repository at any point in time.

Visual Programming

Every *zen* application is developed with the *zen Developer*. The *zen Developer* is integrated in the widely used free Java development environment (IDE) Eclipse. In the *zen Developer*, new applications are modelled with the help of workflows; the data model is developed and the application-specific business logic integrated. In contrast to that, the development of the Java code itself can take place in any other IDE. Once a workflow has been created, any part of its structure can be reused without limitation in other applications.

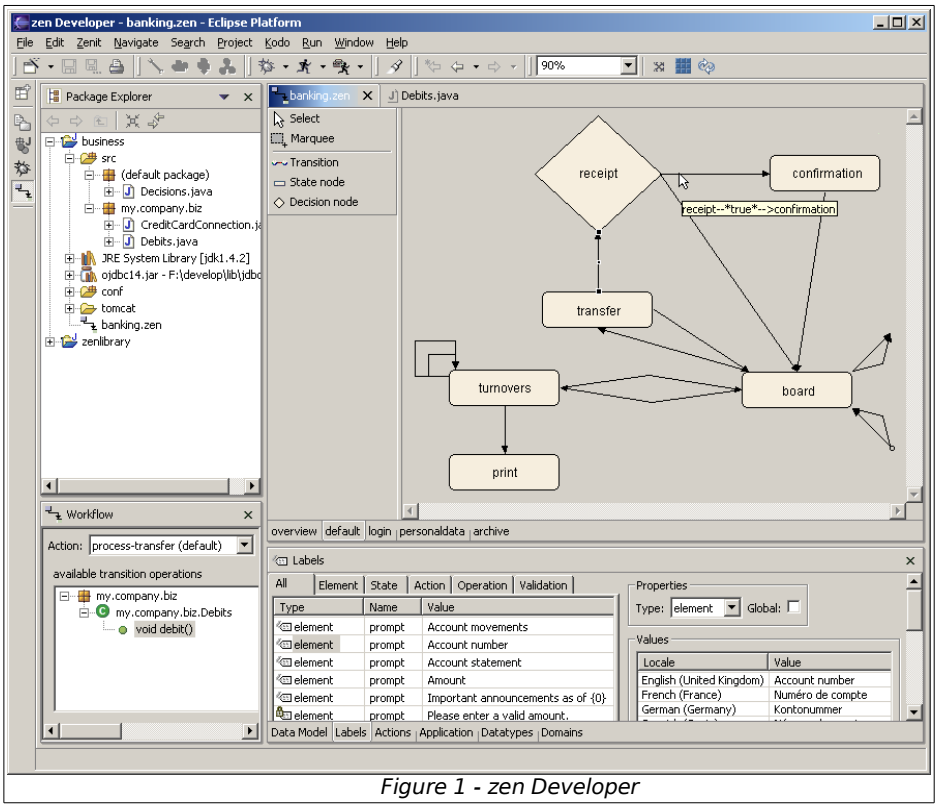

Figure 1 - zen Developer

With a simple click in the *zen Developer* you can start a prototyping session to thoroughly test your application on the basis of the current model in any web browser of your choice, without having to create specific output front ends. In the *zen Developer*, applications can also be viewed and tested in a graphical debugging session, during which the *zen Developer* provides you with insights into the state of the running system at any point in time.

Runtime System

An application modelled with the *zen Developer* is executed by the *zen Engine*. The *zen Engine* is entirely Java-based and will therefore run on any Java Virtual Machine (VM) as of v1.3. In order to execute an application, the *zen Engine* looks into the repository for the corresponding definition of the application which was previously created using the *zen Developer*. The combination of *zen Engine* + business logic + repository behaves like a standard Java application at runtime. A *zen Engine* can execute any number of different *zen* application at the same time.

The *zen Engine* is an abstraction layer on a J2EE application server with which the complex interfaces of the application server resp. the J2EE specification are enormously simplified for the user, as well as optimized and extended to include additional, often sought-after services. This enables the *zen Engine* and any application developed with it to run on any VM of Java 2 Platform, Standard Edition (J2SE), even without application server.

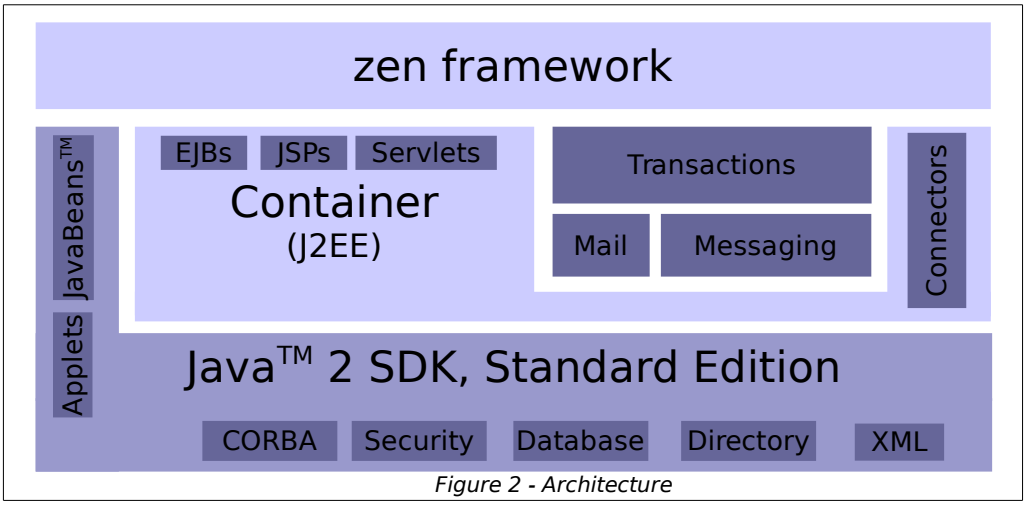

Figure 2 - Architecture

For applications with web front ends all you need is a servlet engine

Scaling

The runtime flexibility allows every *zen* application to be scaled to meet the respective requirements without having to rewrite a single line of program code. This approach is to be distinguished from the traditional development method in many respects. Usually, the increasing need for scaling makes it necessary to completely redesign the application, especially if the need for scaling was initially not assessable, not considered due to budget restrictions or not fulfilled because of architecture-related limitations. With *zen*, however, the business logic of every application automatically grows with your needs because it utilizes the abstracted services of the *zen Platform* and is therefore independent of the environment in which it is executed during runtime.

You can also benefit from this flexibility during the development phase of an application, because you do not need to install an application server. The combination of Eclipse and *zen Developer* is sufficient to develop and test even business critical applications.

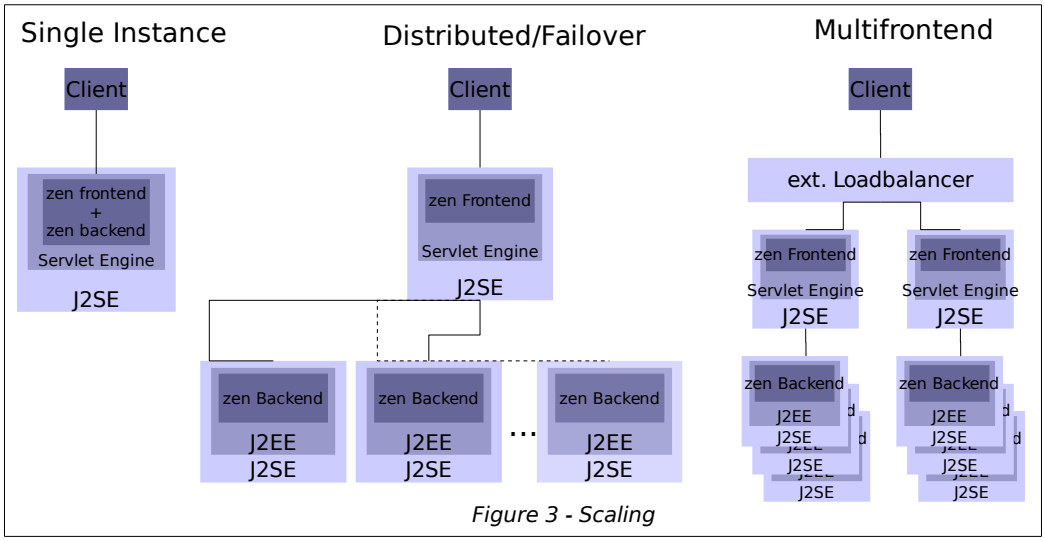

Figure 3 - Scaling

In a cluster deployment the session data are automatically replicated in the cluster in accordance with your configuration, thus guaranteeing transparent failover and maximum availability.

Process Control

5-Level Validation

The *zen Platform* operates with an automated 5-level data validation which does not accept new input data into the application until they match the developer's definition. In each validation phase, the request data undergo a more exact validation.

(1) **Workflow validation**: The system ensures that a workflow can only be entered at defined entry points and is then passed through in a consistent manner.

(2) **Structural validation**: Every new request that is put to the system must exactly match the defined expectation with respect to its structure, regarding the workflow state.

(3) **Syntax validation**: The content of the input data is checked and its type safety is examined.

(4) **Domain validation**: If data has been assigned to a domain, the belonging of the input data to the domain is looked up.

(5) **Inter-object validation**: After a dynamic dependency analysis the application data are validated by business rules and business logic.

The request will only be further processed if the validation has been passed. If errors have occurred, the request is automatically returned with error messages to the front end it originated from.

Workflow-Driven Execution

An application may enter the process control of the *zen Engine* at different points by linking the business logic with defined entry points in the workflow of the application. If an application is executed and one of the respective points in the workflow has been reached, the connected business code will be executed using the required data, with the code having access to all application data.

Data-Driven Execution

Business logic cannot only be linked with the workflow, but also with the data model. Unlike the workflow-driven application the business logic is only activated if data linked with it have changed.

Layered Architecture

Technical View

On a technical level the *zen Platform* is a multi-tier architecture. In most cases, the web container is the front end of the system, and all of the validation and business logic are carried out in the back end, i.e. the application server.

The clear separation of front end and back end makes it possible to couple applications to any number of different and independent output channels without affecting the business logic of the application, simply by exchanging the front end. Existing output formats include XML, HTML, TEXT, WML(WAP), and PDF, and there is no limit to extensions.

Logical View

The workflow of a *zen* application is modelled like a state diagram. A workflow can be broken down into any number of processes. First, the data structure of an application is defined independently of the workflow. Then the input and output data for every workflow step are specified, and the business logic is linked with the workflow and data.

The clearly structured logical view is entirely independent of the technical view on the *zen Platform*. This is one of the reasons why the application developer can focus on achieving the business objectives, because extensive J2EE expert knowledge is not necessary for developing the application.

Services

zen offers the application developer a transparent layer of services which can be used without placing any restrictions on the business logic. The developer can access and use each of the individual services regardless of the deployment of the overall system. zen abstracts the service layer from the runtime environment. Apart from the standard services defined by the Java Beans specification a number of additional useful services are provided.

Data Sources

Aside from the access to data bases via JDBC, *zen* also enables the developer to access data via JDO (Java Data Objects).

Messaging

Developers are able to directly access messaging subsystems using the Java Message Service (JMS), and in addition to the standard, *zen* offers the possibility of querying the processing state of dispatched messages.

Transaction Support

Data sources can be marked as transactional in order to administer them automatically via a transaction manager (Java Transaction API). This gives the developer full control of the entire transaction process. An integration with existing transaction managers is possible.

Logging

For every application individual logging channels can be defined, into which the developer can write out of the business logic. Every log entry can moreover be classified as well. The output of the individual log channels can be made to several different output targets by simple configuration, depending on the classification. Aside from the output as a file, this allows you to have any number of individual log entries be e-mailed or made persistent in a database.

Access to Resources

Even if the Enterprise JavaBeans (EJB) specification does not support direct file access, you sometimes don’t want to do without it. Therefore *zen* offers a service which allows secure and transparent access to any resources you need.

Administration

A *zen deployment* during runtime is controlled with a centralized administration console in the web browser. All running applications can be monitored on the administration console. It is also possible to add new servers for a cluster deployment or to switch off existing servers.

The integrated monitoring allows you to control the runtime behaviour of every server right down to the level of individual requests. If load problems occur, the monitoring function is able to send automatic alarm messages. Moreover, the runtime system can be configured by using the console, e.g. for a reconfiguration of log channels.

Localization

Applications designed with the *zen Platform* basically have the capacity of being localized. The resources for localizing such as labels or URLs are defined in the repository and marked with a combination of country and language. During runtime the *zen Engine* selects the proper resources matching the selected language. Therefore code adjustments are no longer necessary for new localizations. A translation stored in the repository is all you need to execute an application in a new language.

The resources themselves are either of a static or dynamic nature, meaning that for the latter the *zen Platform* actually generates session-specific contents during runtime. The problem of localizing images and labelled buttons is solved by an additional module which newly generates these buttons during runtime, including dynamic rendering of colours.

Summary

In order to be a short term and long term value driver, an IT platform must have the ability to run in any environment, must integrate seamlessly into existing systems and offer efficient support of any business problem. The *zen Platform* was designed on the basis of these fundamental requirements as a flexible platform for business critical applications. The visual development, the clear and well defined structure of the system, the abstraction of complex technologies and last but not least the comprehensive integrated services give every development process a productivity boost. With rapid prototyping, problems can be isolated at an early stage and costly abortive developments can be avoided. Enabling significant cost and time savings, the *zen Platform* already pays for itself during the development phase.

Flexibility and productivity at all development and maintenance stages thus rise to a new dimension.

For maximum success of complex IT projects we additionally provide consulting and training services by experienced experts.

For further information please contact info@zeos.de

zeos informatics GmbH
Gerhardstraße 1
D-81543 München
Phone: +48 89 65114244
Fax: +49 89 65114238
Web: www.zeos.de