Journal of Computer Science 4 (4): 290-297, 2008
ISSN 1549-3636

Towards Persistence Framework-Based Rapid Application Development Toolkit for Web Application Development

Choon How Choo and Sai Peck Lee
Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia

Abstract: Software systems must be delivered quickly in order to respond to today's rapid changing business environments. Persistence framework was introduced to overcome the problem of object-relational impedance mismatch, occurred in most enterprise applications that need access to a relational database. However, most of these persistence frameworks are difficult to configure and use, thus do not really contribute much to the improvement of software developers' overall productivity. This study proposes the concept, architecture, design and development of a rapid application development toolkit that will leverage on a persistence framework to subsequently provide an easy-to-use and customizable front-end web application development environment for developers to perform rapid web application development. Unlike prior efforts, the combination of the features of rapid prototyping, code generation and configuration wizard on top of the persistence framework provided by the proposed rapid application development toolkit enables developers not only to deliver their target web applications within a shorter timeframe through an easy-to-use front-end environment, but also to achieve encapsulation of database access from the business objects of a web application.

Key words: Rapid-prototyping, code generation, framework reuse

INTRODUCTION

In a rapid changing environment, software systems must be delivered quickly in order to meet business delivery schedules. Spending months and years developing systems to high standards is fruitless if over time requirements change beyond recognition. Software development must serve its customers. Simple value-for-money systems that work are better than expensive and complex ones delivered late, over-budget and difficult to maintain[1].

Rapid Application Development (RAD) has long been promised to be a boon to the computing community. The idea is to develop a method of designing software so that the whole process is quick, painless and nearly effortless. The tools used should be easy to learn, powerful and allow the designers to interface their freshly minted application with other applications, databases and file types[2]. While no universal definition of RAD exists, it can be characterized in two ways: as a methodology prescribing certain phases in software development (similar in principle to the spiral, iterative models of software construction) and as a class of tools that allow for speedy development of objects, graphical user interfaces and reusable code for client/server applications[3].

Vidyadharan[4] described the approach of rapid application development through automatic code generation of model objects and persisting them using object persistence frameworks. Most of the application source code can be generated based on rapid application development framework which helps reduce development timeframe. Some companies offer products that provide some or all of the tools for RAD. These products include requirements gathering tools, prototyping tools, computer-aided software engineering tools, language development environments such as those for the Java platform, groupware for communication among development members and testing tools. RAD usually embraces object-oriented programming methodology, which inherently fosters software reuse[5]. The most popular object-oriented programming languages, C++ and Java, are offered in visual programming packages, often described as providing rapid application development.

What is a persistence framework? In computer science, Persistence refers to the characteristic of data that outlives the execution of the program that created it. Without this capability, data only exists in memory

Corresponding Author: Choon How Choo, Department of Software Engineering, Faculty of Computer Science and Information Technology University of Malaya, 50603 Kuala Lumpur, Malaysia PH: 6012-9107928

and will be lost when the memory loses power, such as on computer shutdown[6]. A framework is a basic conceptual structure used to solve a complex issue[7]. In the software context, a framework is a reusable design for a software system (or subsystem). A software framework may include support programs, code libraries, a scripting language, or other software to help develop and glue together the different components of a software project[8]. Frameworks are also defined as a set of classes that embodies an abstract design for solutions to a family of related problems or a set of objects that collaborate to carry out a set of responsibilities for an application subsystem domain[9]. A persistence framework moves the program data in its most natural form (in-memory objects) to and from a permanent data store (database)[10]. Most such frameworks require developers to maintain lots of meta-data describing how to map the object data into the relational database. However, with the advent of advanced language features (reflection), most of this meta-data can be obtained at runtime[11].

This study is organised as follows. First, the motivation of our research work is described. The following section gives an architectural view of our RAD toolkit as well as a detailed description of each of its components and rapid application development process. Subsequent design and implementation of the toolkit is then given. The next section provides a comparative evaluation of our toolkit in relation to existing research prototypes. Finally, a concluding remark is given in the last section.

MOTIVATION

Nowadays, most enterprise applications need access to a relational database[12]. The Java standard for accessing relational databases is the JDBC APIs that utilize SQL as a data manipulation language. This approach of directly accessing a relational database from an object-oriented Java application was shown to be inefficient and introduces a problem called object-relational impedance mismatch, or simply impedance mismatch for short, which refers to the differences between object-oriented technology and relational technology. An approach for solving the impedance mismatch problem is to introduce an abstract layer, called a persistence layer/framework, between the relational database and the object model of the application. This layer fully encapsulates the database access from the business objects[13].

Some known existing persistence frameworks (both Open Source/Academia and Commercial) are such as Hibernate, JDO, Castor[10] and PersistF[14]. Most of the persistence frameworks require users (developers) to configure the framework's behaviour by providing enough information about the target application and the way in which its persistence aspects are to be handled in the context of a given framework. This affects the overall productivity of developers and complicates the development process by forcing the developer to learn the lingo of the target framework.

Tedious, error-prone configuration tasks and coding works may lead to costly, time consuming process, that increase time-to-market for the target application. PersistF, a persistence framework prototype which was developed within our research programme, has been chosen as a basis for exploration, design and development of our proposed RAD toolkit due to easy configuration of its framework behaviour. This research intends to develop a RAD toolkit that will leverage on a persistence framework to subsequently provide an easy-to-use and customizable front-end web application development environment for developers to perform rapid web application development.

ARCHITECTURE OF PROPOSED RAD TOOLKIT

In the context of this research work, software developers are programmers who want to develop web applications. They will be involved in two development phases using the proposed RAD toolkit – pre-development through RADEWeb and post-development through Java IDE. Pre-development involves using RADEWeb which is a front-end web application specification environment of the toolkit installed in a server to enable web application specification, rapid prototyping through the web and generation of PersistF-based web application skeleton source code. Post-development involves using Java integrated development environment (Java IDE) that provides comprehensive facilities such as Eclipse and NetBeans to perform further application development on the generated PersistF-based source code such as code customisation, specific methods addition and code fine-tuning. Fig. 1 together with Fig. 2 shows the overview of the proposed RAD toolkit's concept and architecture which aim to perform rapid web application development in synergy with PersistF.

Pre-development through RADEWeb: In the pre-development phase (Fig. 1), software developers can access to RADEWeb through their web browsers. They can either create a new web application workspace or access to the existing workspaces in order to proceed with the web application development. A Web Application Workspace and a RP Runtime Database (

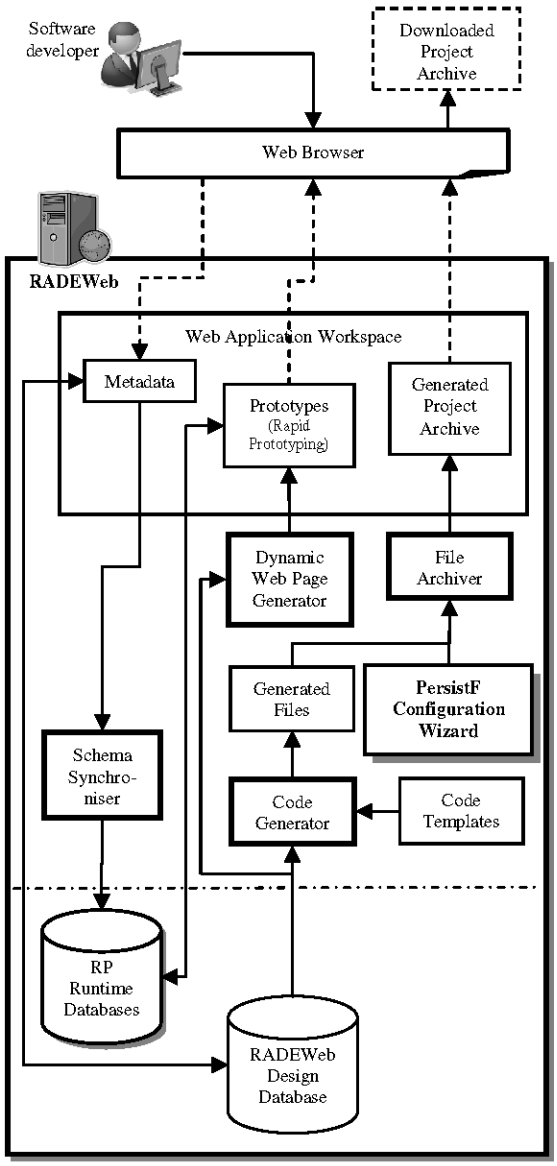

Fig. 1: Overview of the proposed RAD toolkit for web application development through RADEWeb

temporary database for the purpose of rapid prototyping) will be allocated for each web application to be developed.

In this context, the web application design information which will be entered by the software developer is referred to as Metadata. It may include, but not limited to, class information (names of classes and their parent classes related through inheritance) for the web application, field names for each class, data type and input type for each field (for web interface customisation). These metadata will be stored in RADEWeb Design Database and will also be used to synchronise with RP Runtime Database. Software developers can also access to the targeted web application prototypes through the workspace. They can test and feel the freshly minted prototypes before generating and downloading the web application project archive which contains the skeleton source code bundled with PersistF.

Basically, the functions of metadata manipulation (add, edit, view and delete), rapid prototyping, web application code generation and project archive downloading which are available in the workspace supported by four main components in RADEWeb. They are Schema Synchroniser, Dynamic Web Page Generator, Code Generator and File Archiver.

Schema synchroniser: Schema synchroniser plays an important role in synchronising the schema of a RP Runtime Databases based on the design information that has been captured in the metadata. This component enables the latest database to be ready for rapid prototyping immediately after any metadata manipulation performed by the software developer.

Once a new web application workspace has been added by the software developer, this synchroniser will create a new RP Runtime Database schema. Likewise, the database schema will be dropped by the synchroniser if the workspace is deleted by the software developer. The schema synchroniser will also take care of database tables and their columns. Once a class has been added, it will create a new table in the corresponding RP Runtime Database based on the class name. The table name will be synchronised if the class name is renamed and also the table will be dropped once the corresponding class is deleted. This is the same for the class fields, which will be synchronised to the table columns. The following pseudo code shows some of the rules of schema synchronisation.

- Create a new database schema:

```
IF new web application workspace was created
  successfully THEN

      CREATE new database schema with the
        prefix name "radeweb_db" and ending
        with web application increment id

ELSE
      DISPLAY error message
END IF
```

- Create a new table:

```
IF new class was created successfully THEN
    CREATE new table with the same name of
      the created class into the corresponding
      RP Runtime Database
ELSE
    DISPLAY error message
END IF
```

- Add a new column:

```
IF new field has been added successfully THEN
    ALTER the corresponding table with
      command of Add Column to add a new
      column with the same name of the created
      field
ELSE
    DISPLAY error message
END IF
```

Since the schema synchroniser will take care of all the synchronisations automatically, software developers can browse the prototypes from time to time without the need to worry about database schema compatibility.

Dynamic web page generator: For the purpose of rapid prototyping, Dynamic Web Page Generator generates the requested web page prototypes based on the design information of the web application stored in RADEWeb Design Database. The dynamically generated web page will be provided with sufficient parameters to be passed to the generic data manipulation Servlets (web pages backend code) to implement data storing and data retrieving. The following pseudo code shows some of the rules on how the dynamically generated web page can be ready with the basic input validation in javascript function and parameters for each field assigned to the web form:

- Input Validation:

```
FOR each field of the class which is to be filled in
    Prepare checking for the corresponding field
END FOR
```

- Web form preparation:

```
FOR each field of the class which is to be filled in
    Prepare the label name which is derived from
      corresponding field
    Prepare the input field which has been
      assigned with the corresponding attribute
      name
END FOR
```

With this approach, the web forms to add and edit the record and the web pages to list down the records can be generated dynamically and this allows software developers to implement manipulations of data such as add, edit, view and delete to and from the corresponding RP Runtime Database just after any metadata manipulation.

Since there is no time consuming deployment process involved during the rapid prototyping, software developers can change the design information immediately if they are not satisfied and check again with the prototypes within a very short timeframe.

Code generator: Code Generator generates the Tomcat-based web application's skeleton source code files, which are built upon PersistF, by referring to the design information described in the metadata stored in the RADEWeb Design Database. These generated source code files include domain classes, Servlets, dynamic web pages, javascript, cascading style sheets, web configuration and deployment descriptor and PersistF framework related files.

For the generation of each domain class source code, the code generator will first retrieve the package containing any available class, its parent classes linked through inheritance (if available), fields and data type for each field from the RADEWeb Design Database. It will then start to generate each domain class into a package name (if available). Next, it will look into the body of the class which starts with the class name and its inheritance information (if available) will be filled up with fields and basic methods (constructor, getters and setters). These fields will be generated based on the defined field names and data types, with private as their default access modifier. For the generation of the basic methods, constructors will be generated based on the given class while getter and setter methods will be generated based on the names derived from the defined field names. In order to follow the Java naming conventions, these getter and setter methods are generated by combining get or set with the field name in which the first letter is converted to upper case.

For code generation of Servlets, dynamic web pages, javascript and cascading style sheets which may refer to domain classes involving class inheritance, information of parent/super class such as package name, class names, field names and data types are necessary for the proper code to be generated. If the class extends from another class (parent class), then the code generator will analyse its parent class recursively until the base class is found. The following pseudo code

shows the algorithm on how the base class can be traversed from the child class recursively:

```
traverseToBaseClass(class)
    IF the class exists THEN
        CALL traverseToBaseClass(class's parent)
        Statement to be performed for every class
          which will be implemented with the
          sequence from base class to the lowest
          child class
    END IF
```

Based on the generic design of the web application to be generated from this RAD toolkit, the web configuration and deployment descriptor (web.xml) can be generated by providing the code generator the package name and name of class of the web application main controller. All these automatic coding functions provided by the code generator avoid a lot of manual coding errors and at the same time reduce the development timeframe.

File archiver: File Archiver combines the generated files with PersistF configuration wizard of the web application into a project archive file (in .zip format) for easier transportation from RADEWeb to the developer's workstation. This approach saves software developers from wasting a lot of time on downloading source code files one by one from the web pages.

Post-development through Java IDE: In the post-development phase through Java IDE (Fig. 2), further web application development will be carried out in the developer's workstation. By using facilities in Java IDE such as Eclipse or NetBeans, he/she can import the project archive into a new web application project – Tomcat project, to proceed with further development as needed.

During the development of a web application, the developer can configure PersistF's behaviour by using the PersistF configuration wizard (which has previously been archived as a jar file) along with the web application's source code in the project folder. Configuration through the wizard avoids tedious and error-prone manual coding task which may lead to costly and time consuming process.

PersistF plays an important role in taking care of all the persistence-related work on behalf of the developer during the web application development[14]. This means that the management of unique identifiers, relationship between the classes and materialisation /de-

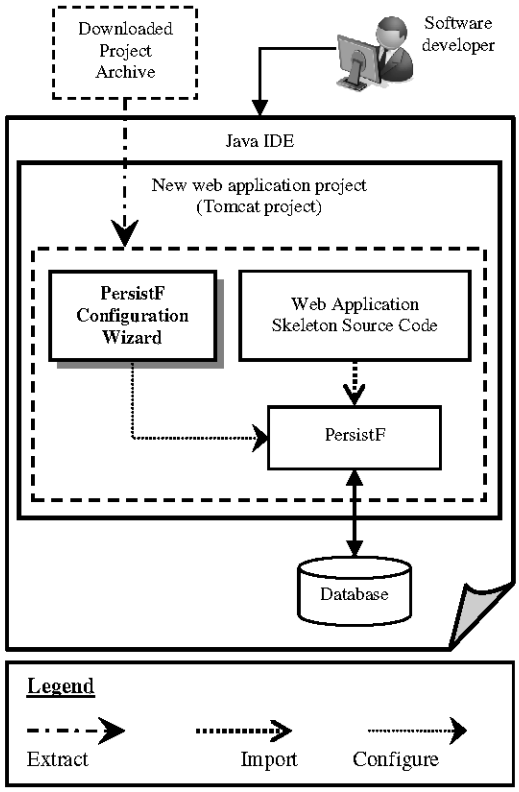

Fig. 2: Overview of the proposed RAD toolkit for development through Java IDE in synergy with PersistF

materialisation of objects from/to the specific data store are done automatically. This saves a lot of time from writing hand-coding SQL and the supporting code to turn it into Java objects.

The resulting web application's skeleton source code and the encapsulated database access layer of PersistF contribute to providing RAD features in subsequent web application development.

DESIGN AND IMPLEMENTATION

The aim of this section is to present the design and implementation of a rapid application development toolkit. One of the ways to increase productivity is to reduce complexity[19]. With the simple development process as shown in Fig. 3, together with the code generation facility and configuration wizard, it will help to reduce coding errors during development and deployment of web applications. In addition, it will also aid developers in building web applications with a set of services without writing basic code.

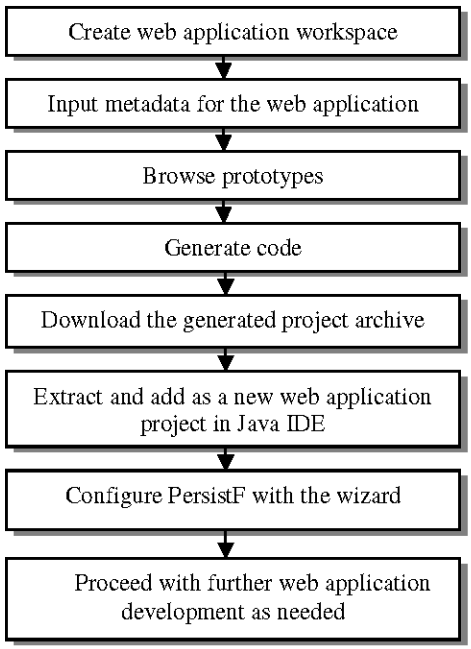

Fig. 3: Process of the web application development with the proposed RAD toolkit

RADEWeb: RADEWeb, a front-end, web application specification, rapid prototyping and skeleton source code generation development environment, allows the software developer to create a workspace for the web application to be developed, input metadata for the web application, browse web application prototypes, generate the web application's source code based on the captured metadata and reusable design infrastructure of PersistF, as well as download the generated web application .zip archive. RADEWeb enables multiple users to access and develop web applications from anywhere and at anytime without having to spend much time with tedious development environment configurations.

Web application workspace: The software developer is allowed to create more than one web application workspace. When a new web application workspace is to be created, the name of the web application needs to be entered. Every successfully created web application workspace will have its own runtime database schema for the purpose of rapid prototyping.

Metadata: The metadata for the web application may include, but not limited to, class names for the Web application, field names for each class, data type and

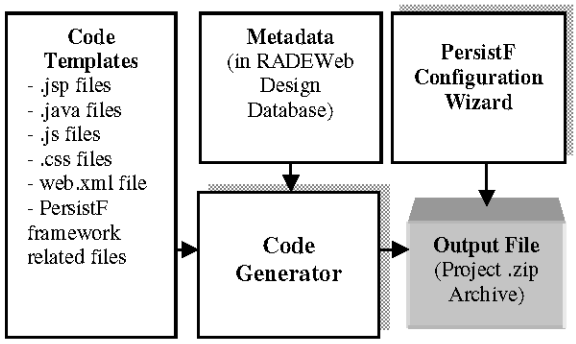

Fig. 4: Overview of Code Generation

input type for each field (for web interface customisation). All of these metadata will be stored in the RADEWeb design database.

Rapid prototyping: From the metadata of the web application, the software developer can browse the web application prototypes which were generated dynamically based on the captured metadata. He/She can test and feel the freshly minted web application instantly and edit the metadata as needed.

Code generation: Once the software developer is satisfied with the prototypes, he/she can generate skeleton code of the web application based on the captured metadata in synergy with PersistF. This saves developers from spending time on repetitive and error-prone manual coding work which may lead to costly, time consuming process and increase time-to-market of the target web application. The generated skeleton code has an architecture that indirectly reuses PersistF defined design patterns and thus, reducing the risk of coding manually. Fig. 4 showed an overview of code generation. There are two inputs for the code generator - metadata which is stored in the RADEWeb design database; and code templates. Only metadata and code templates which are related to the specific web application will be retrieved and analysed for the purpose of code generation.

Downloadable project .zip archive: Generated source code of a web application will be compressed as a .zip archive so that the developer can download it easily from RADEWeb. This compressed archive contains the web application's skeleton source code which can be extracted and added as a new project in Java IDE such as Eclipse or NetBean for further development as needed.

PersistF configuration wizard: PersistF configuration wizard forms part of the proposed RAD toolkit. It is used to configure the settings of PersistF for the web application to be generated. By using this wizard, developers only have to follow the instructions; choose an appropriate setting and edit it as needed from the GUI interface instead of having to edit code manually such as using a traditional text editor to configure the framework's runtime engine with sufficient information about the domain application. This makes error-prone configuration tasks much easier with the wizard. It also prevents developers from spending time on fixing setting errors that frequently occurs during configuration.

COMPARATIVE EVALUATION

Compared to previous work (INTRAD) done by Hyo *et al.*,[9] in terms of features, this proposed RAD Toolkit not only can better improve productivity due to framework reuse but also generate web pages dynamically based on the given analysis and design information for the purpose of rapid prototyping. In relation to Vidyadharan[4] in terms of generation of classes, both code generators allow Model classes to be generated automatically. However, the View classes have to manually code in Vidyadharan's work while this proposed RAD Toolkit allows software developers to enter the needed View class information through the easy-to-use front end interface so that View classes can be generated automatically by the code generator.

The problem faced by Bolwidt and Partington[15] was the speed at which they could build the user interface. The proposed RAD Toolkit overcomes this problem by providing a code generator that enables basic user interface to be generated automatically based on the corresponding design information stored in RADEWeb Design Database.

Basically, previous works by Hyo *et al.*,[9], Vidyadharan[4] and Bolwidt and Partington[15] did not provide any rapid prototyping tool for web application development. This proposed RAD toolkit allows software developers to implement rapid prototyping without much effort once design information exists in the RADEWeb Design Database.

The rapid prototyping tools of Autoweb, JWeb[16] and WARP[17,18] for web application development, as well as the RAD tool of INTRAD by Hyo *et al.*,[9] do not address data storage solution such as a persistence framework which can be used to encapsulate the data access and increase the maintainability of the software. Even though the work by Bolwidt and Partington[15] makes use of a persistence framework, they do not provide any code generator and easy-to-use persistence framework configuration wizard to increase the productivity.

This proposed RAD toolkit can rapidly generate web applications that are built upon PersistF which will take care of all the persistence-related work on behalf of the software developer. The PersistF configuration wizard will be provided together with the generated web application source code files in the generated project archive. Furthermore, the web page prototypes for the purpose of rapid prototyping and all the skeleton source code of the target web application to be developed can be generated dynamically and automatically respectively through this RAD toolkit. The combination of these features to speed up the web application development process is rarely provided by any existing RAD tools.

CONCLUSION

This research proposed a RAD toolkit for rapidly developing web applications, applying a persistence framework that fully encapsulates the database access from the business objects. The RAD toolkit aids software developers in building web applications with an easy-to-use and customizable front-end web application development environment supported by rapid prototyping and code generation facilities on top of a persistence framework without writing any basic code. It avoids having to perform error-prone manual coding tasks during web application development.

In addition, the combination of the features of rapid prototyping, code generation and configuration wizard provided by the proposed RAD toolkit has drastically helped to speed up the web application development process. The simple and organised development process defined has provided a guide to developers in performing rapid web application development with the toolkit.

This research work attempted to provide several features contributing to productivity improvement that so far have not been tackled by existing research works. Future works could be on enhancing this toolkit to support development of complex web applications.

ACKNOWLEDGEMENTS

This research is conducted as part of the framework within the research programme sponsored by Ministry of Science, Technology and Innovation, Malaysia.

REFERENCES

1. Howard, A., 2002. Rapid Application Development: Rough and dirty or value-for-money engineering? Commun. ACM, 45: 27-29. doi:10.1145/570907.570925
2. Brockwood T., 1997. Rapid Application Development. Retrieved July 26, 2007, from http://www.webdevelopersjournal.com/articles/rad.htm
3. Agarwal, R., J. Prasad, M. Tanniru and J. Lynch, 2000. Risks of rapid application development, communications of the ACM. Assoc. Comput. Mach., 43: 177-188. doi:10.1145/352515.352516
4. Vidyadharan, R., 2006. Rapid application development with data binding and object persistence. Retrieved July 20, 2007, from http://www.sptci.com/products/articles/rad.pdf
5. Berry, V. and A. Naumann, 2007. What is rapid application development-a definition from Whatis_com. Retrieved July 27, 2007, from: http://searchsoftwarequality.techtarget.com/sDefinition/ 0, ,sid92_gci214246,00.html
6. Wikipedia, 2007. Persistence (computer science). Retrieved July 25, 2007, from http://en.wikipedia.org/wiki/Persistence_%28computer_science%29
7. Corcho O., A. López-Cima and A. Gómez-Pérez, 2006. The ODESeW 2.0 Semantic Web application framework. Proceedings of the 15th International Conference on World Wide Web, Association for Computing Machinery, Edinburgh, Scotland, pp: 1049-1050. doi:10.1145/1135777.1136009
8. Wikipedia, 2007. Software framework. Retrieved July 26, 2007, from http://en.wikipedia.org/wiki/Software_framework
9. Hyo T.J., K.K. Dong, J.Y. Young and J.Y. Lee, 2000. A Design And Implementation Of Object-Oriented Framework-Based RAD Tool (INTRAD), Systems, Man and Cybernetics, 2000 IEEE Int. Conference. Nashville, TN. 3: 2057-2061. doi:10.1109/ICSMC.2000.886418
10. RoseIndia, 2007. What is Persistence Framework? Retrieved July 25, 2007, from http://www.roseindia.net/enterprise/persistenceframework.shtml
11. Mertner, M., 2005. What is a Persistence Framework, Retrieved July 25, 2007, from http://www.mertner.com/confluence/display/Gentle/1+-+What+is+a+Persistence+Framework
12. EL-Manzalawy, Y., 2007. Accessing Data Through Persistence Frameworks. Retrieved July 27, 2007, from http://www.developer.com/java/data/article.php/3355151
13. Ambler, S.W., 2006. Encapsulating Database Access: An Agile Best Practice. Retrieved July 25, 2007, from: http://www.agiledata.org/essays/implementationStrategies.html
14. Jusic, S. and S.P. Lee, 2007. Persist F: A transparent persistence framework with architecture applying design patterns. Inform. Sci. Inform. Technol., Inform, Sci., 4: 767-779. http://proceedings.informingscience.org/InSITE2007/IISITv4p767-779Jusi281.pdf
15. Bolwidt, E. and V. Partington, 2006. Java with Spring just as productive as a 4GL RAD tool. Retrieved July 20, 2007, from http://www.xebia.com/file_db/File/artikel%20Erwin%20Bolwidt%20en%20Vincent%20Partington.pdf.
16. Bochicchio, M., R. Paiano and P. Paolini, 1999. JWeb: An HDM environment for fast development of web applications, multimedia computing and systems, 1999. IEEE International Conference. Florence, 2: 809-813. doi:10.1109/MMCS.1999.778590
17. Bochicchio M., and N. Fiore, 2004. WARP: Web Application Rapid Prototyping, Proceedings of the 2004 ACM symposium on Applied computing. Association for Computing Machinery, Nicosia, Cyprus, pp: 1670-1676. doi:10.1145/967900.968232
18. Bochicchio M., and N. Fiore, 2005. WARP for Re-Engineering of Web Applications, Proceedings of the sixteenth ACM conference on Hypertext and hypermedia. Association for Computing Machinery, Salzburg, Austria, pp. 295-297. doi:10.1145/1083356.1083428
19. Arthur J. and S. Azadegan, 2005. Spring Framework for rapid open source J2EE Web Application Development-A case study, Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks. IEEE Computer Society, pp. 90-95. doi:10.1109/SNPD-SAWN.2005.74