# J2EE VERSUS ZOPE

Paul L. Juell, Syed M. Rahman and Akram Salah
*Department of Computer Science, North Dakota State University, Fargo, North Dakota 58105, USA*

Keywords: Zope, J2EE, Enterprise Web applications, Software Testing, Content management

Abstract: This paper compares several features between J2EE and Zope technologies. Both technologies have individual strength and will be appropriate in individual contexts. In choosing a development environment or technology for web applications, a criterion is needed to assess the available development technologies. In order to do this comparison, we have designed a web-based prototype for "managing research information" and implemented the prototype in both technologies. We have compared several key features in both technologies including content managements, session handling, safe delegation, security, and testing facilities. The comparison in this paper forms a basis for making choices for web development technology for academia and industry.

## 1 INTRODUCTION

The objective of our research is to compare J2EE (Java 2 Enterprise Edition) and Zope (Z-Object Publishing Environment) (Isaacson 2002) technologies. Both J2EE and Zope are capable of developing enterprise web applications. To do this study we designed a web-based prototype and implemented the prototype in both J2EE and Zope technologies. We compared several key features in both implementations including content managements, application request-handling, safe delegation, security, testing facilities, session handling etc.

The prototype, we developed, is a program to allow teachers supervising their research students' online. We choose this prototype to compare between J2EE and Zope technologies because it has different levels of security users, complexities and functionalities typical of an enterprise web application. In the rest of the paper we focus mainly on the comparison of features that we have experienced from our prototype or individual technology's documentations.

Zope is an open source web application server, primarily written in the Python programming language (Lerner 2002a). It features a transactional object database, which can store not only content and custom data, but also dynamic HTML templates, scripts, search engine, and relational database connections and code. Zope is built around the concept of "safe delegation of control," Zope's security architecture also allows us to turn control over parts of a web site to other organizations or individuals. The transactional model applies not only to Zope's object database, but also to many relational database connectors, allowing for strong data integrity. Zope includes its own Hyper Text Transfer Protocol (HTTP), File Transfer Protocol (FTP), Web-based Distributed Authoring and Versioning (WebDAV), and XML Remote Procedure Call (XML-RPC) serving capabilities. It can also be used with the Apache or other web servers (Lerner 2002a).

On the other hand, J2EE is platform-independent technology. It is a Java-centric environment and developed by Sun. J2EE is used for developing, building, and deploying web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multi-tiered, web-based applications.

We found that under different circumstances, based on a web application's requirements and the nature of the problem domain, either technology can be preferable over the other one. We found, Zope has an advantage on content-management, security, developing environments etc. In contrast, J2EE has many advantages in Internet technology for example in online transaction, session handling, better testing facilities etc.

The rest of the paper is organized as follows. Section 2 explains the experimental setup; section 3 compares different key features in both technologies. Section 4 presents key points of our research's



L. Juell P., M. Rahman S. and Salah A. (2005).
J2EE VERSUS ZOPE.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 64-71
DOI: 10.5220/0002526500640071

findings. Finally, section 5 concludes with a discussion of our findings and a roadmap to future work.

## 2 EXPERIMENTAL SETUP

Our experiment objective was to compare J2EE and Zope technologies. The comparing prototype, "managing research information" has four different types of user with different level of access i.e. Teacher, Administrator, Student, and Guest. Besides guest user, all other types of users must authenticate to the system. Using our prototype, teachers and students can share their documents, upload/download documents on the web, edit their personal information and communicate effectively. Anyone from anywhere can access this online page and explore a teacher's research interests, current projects, and status.

To do our experiment, we went through the following work sequences:

- Step 1: Designed an online "research management system" prototype.
- Step 2: Implemented the prototype using J2EE technology.
- Step 3: Implemented the prototype using Zope technology.
- Step 4: Compared several key issues in both J2EE and Zope technologies.

In our prototype, we had four different types of users with different level of privilege.

Table 2.1: User type and level of access in the prototype

| User levels | Instances | Area of access |
|---|---|---|
| Level 0 | Administrator | Administrator has access all over the system. He/she can access anywhere in the system, edit any part of the system. |
| Level 1 | Teacher | Teacher has admin access only over his/her own research team. He/she can edit any research or user information on his/her research team. |
| Level 2 | Student | Student has access only in his own account. He/she can edit own information and communicate effectively to get the job done. |
| Level 3 | Guest | Guest has limited access into the system. Typically a guest can browse the public area and do not need any authentication. |

## 3 COMPARISON CRITERIONS BETWEEN J2EE AND ZOPE TECHNOLOGIES

In this section, we discussed similarities and differences between J2EE and Zope technologies by developing an online "managing research information" prototype. We showed how the competing platforms handle at each feature and affected a web application. We looked at few features that were not in our prototype, in those cases, we depended on documentations mainly.

We focused on the following features in our study:

- Web Application Platform
- Data Access
- Development Environment
- Server-side Sessions
- Learning Curve and Documentation
- Safe Delegation
- Testing and Debugging

### 3.1 Web Application Platform

Today a successful web application requires the participation of many people across an organization that has different areas of expertise. We looked at how these two platforms handle web applications.

**J2EE:** Web application is a dynamic extension of a web or application server. There are two types of web applications (Armstrong et al. 2004):

- Presentation-oriented: A presentation - oriented web application generates interactive web pages containing various types of markup language (HTML, XML, and so on) and dynamic content in response to requests.
- Service-oriented: A service-oriented web application implements the endpoint of a web service. Presentation-oriented applications are often clients of service-oriented Web applications.

The J2EE platform web components provide dynamic extension capabilities for a web server. A web component includes Java Servlets, JSP pages and web services. The interaction between a web client and a web application is illustrated in Figure 3.1; the client sends an HTTP request to the web server. A web server that implements Java Servlet and JavaServer Pages technology converts the request into an HTTPServletRequest object. This object is sent to a web component, which can interact with JavaBeans components or a database to

generate dynamic content. The web component can then either create an HTTPServletResponse or it can pass the request to another web component. Eventually a web component generates a HTTPServletResponse object (Armstrong et al. 2004).

**Zope:** Zope presents objects on the web. This is called object publishing. One of Zope's unique characteristics is the way it permits us to access to objects and call methods on them with simple URLs. In addition to HTTP, Zope also makes objects available to other network protocols including FTP, WebDAV and XML-RPC (Lerner 2002a, Zope community 2004).

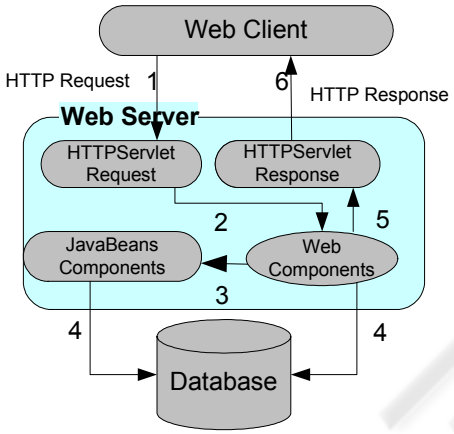


Figure 3.1: J2EE web application request handling

When we access Zope with a web browser, our browser sends an HTTP request to Zope's web server. After the request is completely received, ZPublisher processes it, which is Zope's object publisher. ZPublisher is a lightweight ORB (Object Request Broker, McDonough et al. 2004). It takes the request and locates an object to handle the request. The publisher uses the request URL as a map to locate the published object. Finding an object to handle the request is called traversal, since the publisher moves from object to object as it looks for the right one. Once the published object is found, the publisher calls a method on the published object, passing it parameters as necessary. The published object then returns a response, which is passed back to Zope's web server. The web server then passes the response back to our web browser (McDonough et al. 2004).

Zope object publishing process is summarized in Figure 3.2.

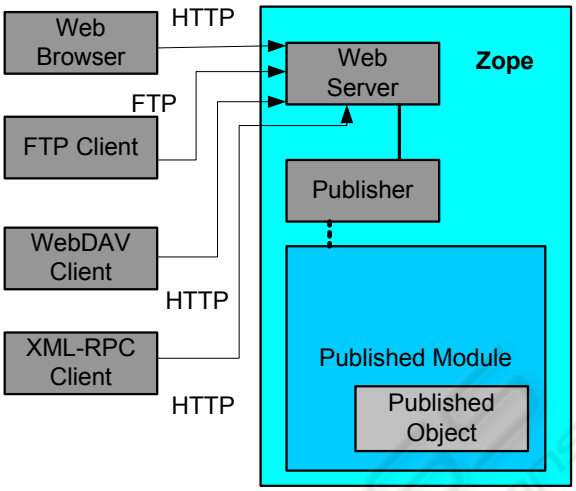


Figure 3.2: Zope object publishing (McDonough et al. 2004)

Typically the published object is a persistent object that the published module loads from the ZODB. We can present the object publishing main steps (McDonough et al. 2004):

- The client sends a request to the publisher
- The publisher finds the published object using the request URL as a map.
- The publisher calls the published object with arguments from the request.
- The publisher interprets and returns the results to the client.

To create and work with Zope objects, we use our web browser to access the Zope management interface. All management and application development can be done completely through the web using only a browser.

The Zope management interface provides a familiar Windows Explorer-like view of the Zope object system. Through the management interface a developer can create and script Zope objects or even define new objects, without requiring access to the file system of the web server.

In our project, we received benefits from Zope. It has built-in application control management system. All we had to do is create different sort of users with different privileges. It is just few mouse clicks away. In compare to J2EE it was complicated, we created different role based user interface by writing Servlets program. This is the most significant facility that we experienced from our experiment.

## 3.2 Data Access

Enterprises revolve around their data, often stored in multiple, heterogeneous data stores.

**J2EE:** In J2EE, persistent data are typically modelled as entity beans. There are two main types: container-managed and bean-managed persistence (CMP and BMP, respectively). Both let developers tap into container-provided services such as transaction management. It's security framework, as well as leveraging the container's innate characteristics such as scalability and fault tolerance. In general, there is a Java package to access almost any corporate data source we care to mention (Sheil & Monteiro 2002, Farley 2002).

**Zope:** Zope provides a consistent object oriented way to access all kinds of enterprise data. Sources can include RDBMS data as well as non- relational data from sources such as LDAP or IMAP servers. Zope supports most common relational databases, including Oracle, Sybase, MySQL and ODBC compliant databases (Switching to Zope 2004).

Zope's object-oriented design allows us to enforce a clean separation of data and presentation. Database programmers can work on SQL Method objects, while content managers can simply call the SQL Methods and use the results in their content. This object model makes it easy to integrate data from multiple data sources. Advanced data access features even allow us to define object behaviour for database results, turning flat relational records into "smart data" in our Zope application.

## 3.3 Development Environment

The development environment is another differentiator between the platforms.

**J2EE:** Applications written in the Java programming language can run in their own windows, unlike applets, which run inside a web browser. Applications are usually larger and more full-featured than applets, and it is important that there is an effect and robust method to deploy them widely, while ensuring that the correct version of the Java platform is available to them. Java webstart is designed for just this purpose (Armstrong et al. 2004, Altendorf et al. 2002).

With Java webstart, a user launches an application simply by clicking on a web page link. If the application is not present on the client computer, Java webstart automatically downloads all necessary files for the application, including a new version of the Java platform if needed. It caches the application on client computer. On subsequent launches of the application, Java webstart will check the network for an updated version of the application, and if there isn't an update, it will launch the cached version.

On Microsoft Windows, Java webstart is automatically installed when the Java Runtime Environment or Java 2 SDK is installed. The Java webstart also comes with the Java 2 SDK and the Java Runtime Environment for Solaris and Linux. It can be installed manually. We can also use JNLP technology for Java application deployment (Altendorf et al. 2002).

**Zope:** The bulk of web application development in Zope is accomplished via a web-based interface. One logs into a Zope application and edits their DTML, Python scripts, etc. inside of their web browser. The upshot is that Zope development is feasible on almost any platform. The downside is that most web browsers make for poor text editors. However, a developer can use a text editor, which has built-in support for FTP (emacs, BBEdit, etc.) or WebDAV in addition to the basic web interface (Lerner 2002a).

## 3.4 Server-Side Sessions

There are a number of problems that arise from the fact that HTTP is a "stateless" protocol. Session is very important for many web applications. For example, when we add an item to our shopping cart, the server needs to know what's already in our cart or when we move from one page to another. The page that takes credit card number and shipping address, the server remembers this entire information and transfers one page to another. There are three typical solutions to handle this session data. These are using cookies, URL rewriting, or hidden form fields.

**J2EE:** Java Servlets provide an outstanding technical solution: the HttpSession API. This is a high level interface built on top of cookies or URL-rewriting. In fact, on many servers, they use cookies if the browser supports them, but automatically revert to URL-rewriting when cookies are not supported or explicitly disabled. Using sessions in Java Servlets is quite straightforward. It involves looking up the session object associated with the current request, creating a new session object when necessary, looking up information associated with a session, storing information in a session, and discarding completed or abandoned sessions (Armstrong et al. 2004, Sheil & Monteiro 2002).

**Zope:** Zope by default provides no support for managing per-user-session data on the server. Several add-on products are available, which provide different forms of sessions (Isaacson 2002, Lerner 2002a, Switching to Zope 2004).

However, the Zope community is aware of this limitation and working to overcome the sessions handling problem.

## 3.5 Learning Curve and Documentation

There are significant differences between J2EE and Zope learning curve and documentations.

**J2EE:** J2EE is an established technology. It is very popular for developing web-based applications. It is developed and maintained by Sun Microsystems Inc. Java has a very large user community and huge learning resources. J2EE documentation is written following standards.

**Zope:** Zope has some problems, the first and biggest one being that the learning curve can be rather steep. Even for an experienced web developer, Zope requires that the developer learn nearly all of the concepts from scratch, changing almost all of the habits that they have acquired over the years. It can be a surprise and a reason to be cautious. Using Zope might slow things down during the initial startup period. Another problem we have noticed is that it does not have enough documentation or it is not well organized or evenly written (IBM 2004).

Comparing J2EE and Zope, the Zope learning curve is steeper. However, a newcomer in this field needs to spend a significant amount of time learning either J2EE or Zope technology.

## 3.6 Safe Delegation

A successful web site requires the collaboration of many people in an organization: application developers, SQL experts, content managers and often even the end users of the application. On a conventional web site, maintenance and security can quickly become challenging. How much control do we give to the content manager? How does giving the content manager a login affect our security? What about that SQL code embedded in the ASP files he / she will be working on - code that probably exposes our database login? We presented how these technologies handle the specified issue.

**J2EE:** In J2EE, when we try to access a protected web resource, the web container activates the authentication mechanism that has been configured for that resource. We can specify the following authentication mechanisms:

- HTTP basic authentication
- Form-based login authentication
- Client certificate authentication
- Mutual authentication
- Digest authentication

If we do not specify one of these mechanisms, the user will not be authenticated. Figure 3.4 shows what happens if we specify HTTP basic authentication. With basic authentication, the following occurs (Armstrong et al. 2004):

- A client requests access to a protected resource.
- The web server returns a dialog box that requests the user name and password.
- The client submits the user name and password to the server.
- The server validates the credentials and, if successful, returns the requested resource.

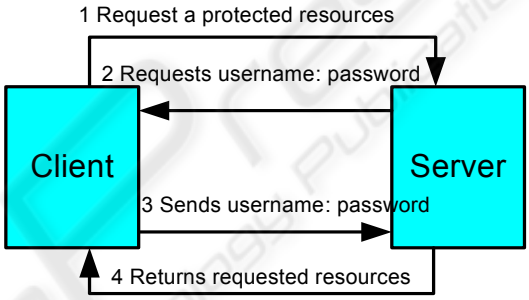


Figure 3.4: HTTP basic authentications (Armstrong et al. 2004)

HTTP basic authentication is not particularly secure. Basic authentication sends user names and passwords over the Internet as text that is uu-encoded (Unix-to-Unix encoded) but not encrypted. This form of authentication, which uses Base64 encoding, can expose user names and passwords unless all connections are over SSL. If someone can intercept the transmission, the user name and password information can easily be decoded.

Form-based authentication is not particularly secure either. Client-certificate authentication is a more secure method of authentication than either basic or form-based authentication. It uses HTTP over SSL, in which the server and, optionally, the client authenticate one another using public key certificates. Secure Socket Layer (SSL) provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection (Armstrong et al. 2004, J2EE 2004).

**Zope:** Zope addresses collaboration issues with its strong object-based design. It enforces separation of data and presentation. It provides a flexible security model designed specifically to handle the unique business models of the web (Lerner 2002a).

Zope manages users with "User Folders," which are special folders that contain user information. Several Zope add-ons are available that provide extended types of User Folders that get their user data from external sources, such as relational databases or LDAP directories (Lerner 2002a).

In addition to strong encapsulation and security, Zope also provides other useful collaboration features. No matter what, something will ultimately go wrong. Once there is a problem, instead of running for the backup tape or spending hours for debugging, the site manager can simply use "undo" option and changes to the site back to a point before the problem started. The manager can also avoid this kind of problem altogether, he/she can work in "Versions," which are private views of the object system (Lerner 2002a, Zope community 2004).

Zope makes "undo" easy and flexible. We do not need to worry about, do any experiment and us undo option if it does not work. On the other hand, to make any small change in Java Servlet code, we had to change the code, compile and run to see the effects. Sometimes it is even harder to investigate what went wrong. Zope preview feature made our life even easier; just make any changes and view the effects right away and use the Undo button if anything goes wrong.

## 3.7 Testing and Debugging

Software Testing is the most expensive phase in software development life cycle. Depending on software types, typically, it is about 40 % to 70% of total cost of software (Kim 2003). We can reduce the overall software cost significantly, if we can reduce testing expenses only. Ostrand and Weyuker (2002) studied on a large AT&T software system. They found early unit-testing method exposed 73% of total faults. It can reveal maximum bugs at the early development stages. Both J2EE and Zope technologies are using unit testing effectively. JUnit has a plug-in for both technologies. However, only unit testing is not enough testing for an application. We have to do requirements and design testing, integration testing, system testing, acceptance testing etc.

**J2EE:** There are many testing tools available for testing a Java program. Some of them are free, and some are not. Sun offers two lines of development tools designed for professional developers. The Sun Java Studio line offers a complete, integrated development environment (IDE) for Java technology encompassing J2ME, J2SE, and J2EE technologies. Sun's tools are Sun ONE Studio, Sun Java Studio Standard and so on. Besides Sun, we have many popular Java testing tools available such as JUnit, GJTester, Cactus, JUnitPerf, Jemmy, Clover, TrueJ etc(Armstrong et al. 2004).

**Zope:** Zope's debugger allows us to peek inside a running process and find exactly what is going wrong. Unit testing allows us to automate the testing process to ensure that our code still works correctly as we change it. We can setup a debug mode and can debug code. Zope uses the JUnit plug-in for unit testing. It is also integrated with Python debugger which is a very simple command line debugger (Zope community 2004).

Our experience with our prototype testing, we had an advantage in J2EE components. The main reason was that we are familiar with Java testing and debugging tools. There are several commercial and non-commercial tools were available to us where as in Zope we experienced few problems for testing and debugging programs. Python command line debugger was not very useful to us. We found the error message was generic and hard to understand. Comparing J2EE, Zope do not have enough tools for requirements testing, design testing, system testing etc.

# 4 RESULT

In this paper, we compared several key features between J2EE and Zope technologies. To do this study, we designed an online prototype for "managing research information" and implemented it in both technologies. We found that depending upon a web application's requirements and features involved, either technology can be preferable over the other one.

Here are some key points that we concluded from our study:

- Downloading and installing one Zope file includes all components that we need for a web application. On the other hand, in J2EE, we need to install different components separately and need to configure as well.
- Zope uses Python language, which is not popular like Java. Java has larger user community and more learning resources than Python. We found very few universities where Python is considered as the primary programming language.
- In Zope, Python script (DTML), can be written inside a web browser.
- Software testing in Zope is difficult. J2EE has many more established testing tools and technology than Zope.

Table 5.1: Compare features between J2EE and Zope technologies

| Features | J2EE | Zope |
|---|---|---|
| Language used | Java | Python |
| Delegate control | Not easy | Very Easy |
| Documentations | Very good | Poor |
| Learning curve | Steep | Steeper |
| Platform stability | Stable | Less stable |
| HTML generation | Use JSP and Servlets | Use DTML and ZPT |
| Server-side session | Servlets provide great support | No support by default |
| Different testing tool availability | Many tool available | Not many |
| Undo option | No | Yes |
| New version | Not easy | Very easy |
| Search Engine | Not available | Built-in search engine |
| Components Availability | Most of the components are not open source and not FREE | All components are open source and FREE |

- Zope has problems with standards and documentations, where J2EE has an advantage on these issues.
- Zope is specially designed to safely delegate control to design expert, database expert, and content managers. On the other hand, in J2EE, it is very difficult to control and maintain projects.
- Zope by default provides no support for managing per-user-session data on the server. Java Servlets offers excellent session tracking technology.

Depending on the application and nature of the problem, J2EE technology is superior over Zope and vice-versa. Zope is very efficient and useful for creating roll-based management system e.g. creating and maintaining a portal. J2EE does a very good job for shopping cart applications, online transactions, and many other web applications.

In the Table 5.1, we summarized features between J2EE and Zope technologies.

# 5 CONCLUSIONS

In this paper, we have compared several key features in J2EE and Zope technologies. We found that based on application requirements and features required, either technology can be superior over other one. To do our study, we designed an online "managing research information" prototype and implemented in both technologies. We compared several key features including applications request-handling, content managements, safe delegation, security, session handling, testing facilities, availability etc.

We have found that Zope has some advantages over J2EE. Zope includes a built-in web server, a content management system and a search engine. It is free to download, does not need any special IDE or any software beside a browser, as a developing environment. It has a built-in "undo" facilities. In Zope, it is very easy to create a new version of the software. On the other hand, Zope suffers from lack of standards, documentations and proper management. It does not have proper session handling technology and Zope components are not stable like J2EE components. The learning curve is steeper for Zope than J2EE technology. We believe, Zope architecture needs to change to handle session data properly and add better testing mechanism.

We have also found that J2EE components are well-documented and built on standards. J2EE has a very large number of users community and learning resources. It has excellent facilities for handling session data, testing facilities, online transactions, concurrency controlling system and other services. Most of the components are not free; it does not have an easy "Undo" facilities. J2EE applications are not easy to make a new version as Zope. J2EE does not allow a site manager to safely delegate control to a design expert, a database expert and a content manager as Zope.

Having walked through the main features and issues that any enterprise technology must address and evaluated J2EE and Zope for each one, what do we think about our respective causes or as a manager which technology we would use for our next web application? We would say that it depends; based on application requirements or features involved in the

application, Zope can be superior over J2EE or vice versa. For a customize application or where frequent changing necessary (e.g. portal), Zope will be a better choice. However, for a large web application where session handling and testing issues are crucial (e.g. online banking software), J2EE will be more preferable.

Our future work includes developing a prototype in both technologies where we can observe features such as online transactions, concurrency control, session handling, huge traffic handling, outside attack vulnerabilities etc. We would also like to contribute to Zope community to overcome Zope weaknesses including session handling and testing facilities.

# REFERENCES


Altendorf, E., Hohman, M. & Zabicki, R. 2002. 'Using J2EE on a large, Web-based project', *IEEE Software*, Volume: 19 , Issue:2, pp. 81-89 (March-April 2002).

Armstrong, E., Ball, J. & others. 2004, 'The J2EE 1.4 Tutorial', http://java.sun.com/j2ee/index.jsp, (June 17, 2004).

Brebner, P. & Gosper, J. 2003 'J2EE infrastructure scalability and throughput estimation', *ACM SIGMETRICS Performance Evaluation Review*, Volume 31 Issue 3 (December 2003).

Deitel, H.M. & Deitel, P. J. 2002, *Java How to Program*, 4th ed, Printice Hall, New Jersey

'Developing for the J2EE Tomcat Platform, Masslight inc'. 2004, http://j2ee.masslight.com/index.html, (July 5, 2004).

Farley, J. 2002, 'Java 2 Platform, Enterprise Edition (J2EE) versus .NET'. http://java.sun.com/ developer/ community/chat/JavaLive/2002/jl0115.html, (July 30, 2004).

Hunter, J. 1998, *Java Servlet Programming*, O'Reilly & Associates inc.

Hutcheson, L.M. 2003, *Software Testing Fundamentals Methods and Metrics*, Wiley Publishing Inc.,USA, pp. 66-67

IBM. 2004, http://www106.ibm.com/developerworks/java/ library/j-pj2ee3.html, (July 15, 2004 ).

Isaacson, P. C. 2002, 'Web Development with Zope', *The Journal of Computing in Small Colleges,* Volume 18, Issue 1.

'Java 2 Platform, Enterprise Edition (J2EE)' 2004, http://java.sun.com/j2ee/index.jsp,(July 30, 2004)

Joseph, W. 2003, 'E-services: The Web Services Debate: J2EE vs. .NET', *Communications of the ACM*, Volume 46, Issue 6, (June 2003).

Kim, Y.W. 2003, 'Efficient use of code coverage in large-scale software development', *IBM Centre for Advanced Studies Conference, Proceedings of the 2003 conference of the Centre for Advanced Studies conference on Collaborative research*, pp. 145 – 155

Lau,T.C., Lu, J., Hedges,E. & Xing, E. 2001, 'Migrating E-commerce database applications to an enterprise Java environment', *Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*, (November 2001).

Lerner, R. M. 2002a, 'Introducing Zope', *Linux Journal*, Volume 2002, Issue 94.

Lerner, R.M. 2002b, 'Databases and Zope', *Linux Journal*, Volume 2002, Issue 97.

Liebmann, E. & Dustdar, S. 2004, 'Web technologies and applications (WTA): Adaptive data dissemination and caching for edge service architectures built with the J2EE', *Proceedings of the 2004 ACM symposium on Applied computing*,(March 2004).

Lloyd, B. 2004, 'An Introduction To Zope', http://www.zope.org/Resources/ZopeIntro/, (May2, 2004).

McDonough, C., Pelletier, M. & Hathaway, S. 2004, 'The Zope Developer's Guide', http://www.zope.org/ Documentation/ Books/ZDG, (June 15, 2004).

MySQL Developer site. 2004, http://dev.mysql.com/, (June 15, 2004).

Naughton, P. & Schildt, H. 1999, *Java 2: The complete Reference*, 3rd Ed, McGraw-Hill, NY

Ostrand , T. J. & Weyuker, E. J. 2002, 'The Distribution of Faults in a Large Industrial Software System', *ACM SIGSOFT Software Engineering Notes*, Volume 27 , Issue 4 ,(July 2002).

Perry, E. W. 2000, *Effective Methods for Software Testing*, John Wiley & Sons, Inc., 2nd Ed, USA pp. 66- 67,123

'Servlets and JavaServer'. 2004, The Johns Hopkins University, http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/index.html, (July 2, 2004).

Shadgar, B. & Hoyer, I. 2004, 'Adapting Databases and WebDAV protocol', *International World Wide Web Conference, Proceedings of the 13th conference on World Wide Web*, NY, USA, Pages: 612 – 620.

Sheil, H. & Monteiro, M. 2002, 'Rumble in the Jungle: J2EE versus .Net', http://www.javaworld.com/ javaworld/jw-06-2002/jw-0628-j2eevsnet_p.html, (July 26, 2004).

'Switching from PHP to Zope/Python (Technology)'. 2004,http://www.kuro5hin.org/story/2004/3/21/184222/896, (July 25, 2004).

White, S., Fisher, M. & others 1999, *JDBC API Tutorial and Reference*, 2nd Ed, Addision-Wesley, USA.

Zope community.2004, http://www.zope.org, (May 5, 2004).