International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 06 | June 2021 www.irjet.net p-ISSN: 2395-0072

Framework migration from Visual Basic 6 (VB6) to ReactJS

Hrithik M R[1], Dr. Sharvani G.S[2]

[1]*Department of Computer Science and Engineering, R.V. College of Engineering, Bengaluru, India*
[2]*Associate Professor, Department of Computer Science and Engineering, R.V. College of Engineering, Bengaluru, India*

Abstract - *With Windows 7 end of support and VB 6.0 IDE not being supported on Windows 10 we have huge risk with respect to the continue development environment for Practice Management and EDI plugin code developed in VB6. Given the challenges outlined above for using a code converter like Mobilize to migrate the Visual Basic 6.0 (VB6) code, taking a step back to assess options for migrating the VB6 code is warranted. At a high level, "Convert VB6 to a REST service-based architecture with light weight UI frontend" approach has been identified. The paper discusses about the existing systems, steps involved in conversion, architecture to address this problem.*

Key Words: Visual Basic, ReactJS, Spring framework, REST API, Practice Management

1.INTRODUCTION

There are many different organizations which provide health care in the United States. Most health-care facilities are owned and operated by private companies. Private health insurance and public health insurance are used to provide healthcare coverage (e.g., Medicare, Medicaid). Unlike most other industrialized countries, the United States does not have a universal healthcare system. To improve clinical and financial outcomes, organization collaborates with healthcare organizations across the care continuum. They have built an application, which is used for healthcare administration. This product is an application which is fully integrated Electronic Health Record (EHR) and Practice Management (PM) system. Everything from billing to clinical processes may be customized with this product to meet our unique practice requirements. This software is used in US hospitals for medical administration.

An electronic health record (EHR) is a comprehensive database of a patient's clinical data and medical history, including demographics, vital signs, diagnoses, prescriptions, treatment plans, progress notes, issues, vaccination dates, allergies, x-rays, and test and laboratory results.

Practice management is concerned with a medical practice's day-to-day operations. Users can enter patient information, schedule appointments, keep track of insurance payors, complete billing tasks, and generate reports. This application has various modules like chart, scheduling, registration, administration, and billing.

1.1 Scope and Motivation

As per new US regulatory rules, many data elements for nationwide, interoperable health information exchange have already been supported by this application. But there are few new data elements that need to be supported as part of new regulatory rules i.e., previous name, previous address for a patient. So, there is need for enhancement to add new functionalities in Registration module. Hence this project adds the extra functionalities and aligns the application with new regulatory rules.

2. LITERATURE REVIEW

In 2017, Having an efficient software application with less response time is very important. The selection of technologies, language and framework plays an important role. In [1], by Hongjun Li, the author has explained about how to use RESTful Web service frameworks in Java to develop RESTful Web service and described about several Restful web service frameworks RestEasy, Restlet, Struts 2, Grails, Axis2, sqlRest. The limitation of this paper was it did not compare between the frameworks. The authors of [2] by Urjita Thakar, Amit Tiwari, and Sudarshan Varma demonstrate how to consume SOAP based as well as RESTful services in composition while maintaining the advantage of RESTful services' lightweight nature. The work related to selecting appropriate component services, detecting faulty component services, and replacing them was not discussed in this study. K Munonye and P Martinek compared and evaluated the performance of REST API implementations based on the Microsoft.Net framework with the Java SDK in their paper [3]. The paper's limitations were that the same workstation served as both the server and the client, that only two metrics were examined in this study: code complexity and response time, and that the dataset was small. The authors of [4] Zhixiang Niu, Cheng Yang, and Yingya Zhang, explained how to design a system based on JSON data format and RESTful web services that has high data exchange efficiency, simple interfaces, ease of modification and expansion, and good cross-terminal support. The limits described in this study are to improve the system's performance and enhance the performance by utilizing some other new technologies. The authors of [5] HaiTao Wang and BaoXian Jia, explain how framework integration can improve system development efficiency and reduce coding workload. The paper's weakness was that it did not explain how to improve system performance, user access speed, or system framework security. Yuxiang Hou introduces the design and implementation of the framework for Spring, Spring MVC, and MyBatis in the development of Web applications in [6]

and it simplifies the development process and workload of the system to improve the system's expansion and convenience of deployment. Zhang Zhenyou, Gu Wei, and Cao Zhi demonstrate how Hibernate technology makes the system dynamically linked to multiple heterogeneous database systems in their paper [7]. The limitation of this paper is there was no information of query optimization. The authors of [8] Qinglin Wu, Yanzhong Hu, and Yan Wang, explained how to create, test, and optimize data persistence layer methods to improve data access efficiency and ensure the quality of Web applications.

3. METHODOLOGY

3.1 Existing systems:

The steps to build a social distancing detector include:

Methodology diagram clearly explains the workflow of the project as to what is exactly done at each stage of the project.

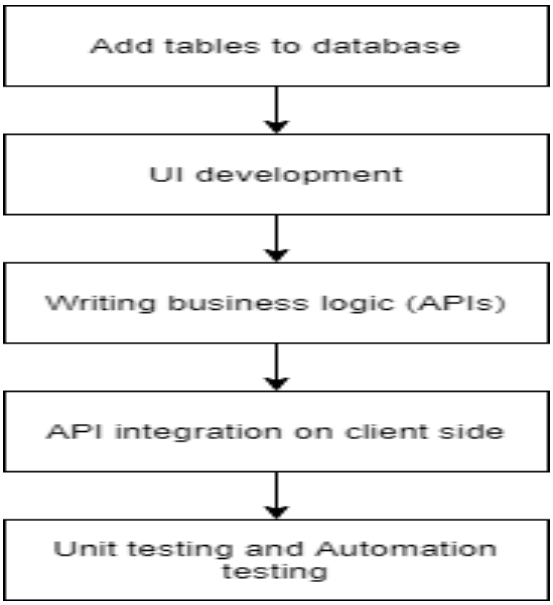


Fig 3.0: Methodology diagram for enhancement of application

All the steps mentioned in the above diagram are explained below:

Add tables to database:
This is the first step. To record the new data of the patient, new tables with required attributes are added in database which is MSSQL server.

UI development:
The user interface is created to add, edit and display the patient's previous demographics using WPF. Dialogs with new and edit buttons and grid view to display the data is created.

Writing business logic (APIs):
The business logic to fetch and save data to the database is encapsulated in REST APIs in Java using JAX-RS annotations and Spring-Hibernate integration. All DB operations (fetch, insert, update and delete) are performed through JBoss service layer.

API integration:
On client side The APIs are called on client side (user interface) to display the data to the user or to insert the new data into database.

Unit Testing and Automation Testing:
Unit testing is done by writing Junit test cases using EasyMock framework. Performance testing of web services is done using JMeter. Automation testing of UI and Web services is done using the testing framework, SpecFlow.

Steps involved in converting VB6 to a REST service-based architecture with light weight UI frontend :

1] The main logic of VB modules can be placed inside the REST services by performing access / manucaption operations on the data.

2] The UI part can be rendered in a lightweight JavaScript (ReactJS/Ng/etc.) based controls running inside an embedded Chromium engine wrapped inside COM component conforming to module binding architecture of product.

3] REST endpoints will be accessed for the all the data content which are to be rendered on UI.

4] If there are workflows which require invoking of C# components by the UI content, then this can be achieved by abstracting the invocation path leveraging the product API.

5] There has to be a proper structure maintained for the content in JavaScript, so that whenever there is a complete migration into different format, then the rewrite of the UI should be minimal and primarily focused on routing, context sharing and invoking other modules of product.

3.2 Architecture:

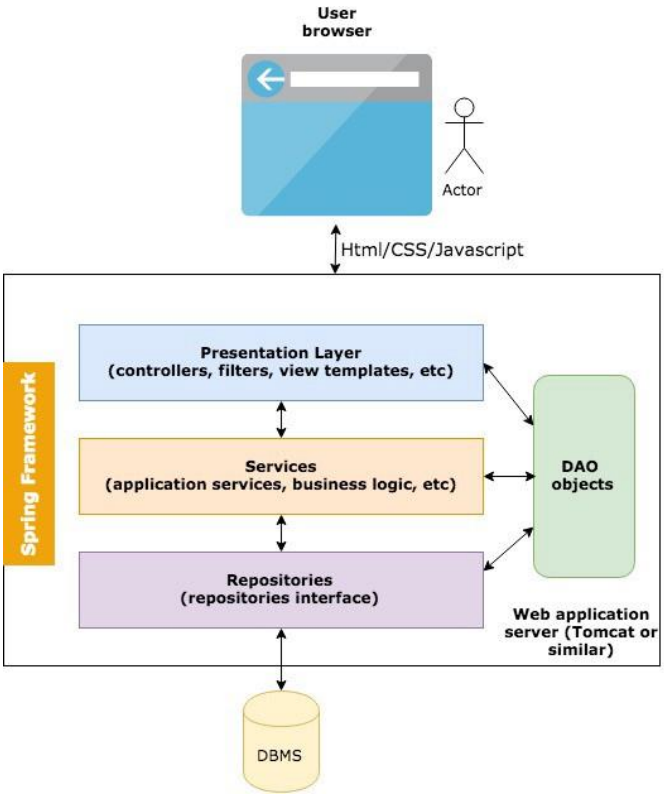


Fig 3.1: Typical Spring Framework architecture

The PM solution has a five-stage workflow that starts from the time the patient schedules an appointment and goes through till the patient checks out and a claim is generated.

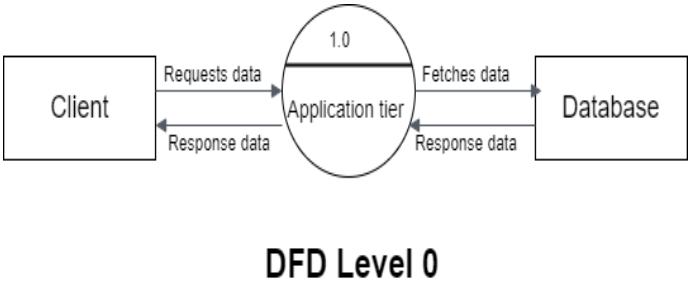
The solution includes five key modules:

• Medical billing to help clinics get paid faster EHR – to ensure accurate and fast management of patient encounters
• Patient communication and engagement modules that help reduce patient no-shows by as much as 8%
• Seamless patient care to improve patient experience and loyalty
• Epocrates – a clinical intelligence module that improves the quality of physician decisions.

The solution is completely web based and runs in data centers. Users do not have to procure servers, firewalls, or other complex equipment. There is no requirement to set up servers, databases, or authentication systems, nor grapple with security issues.

3.3 Data flow diagrams:

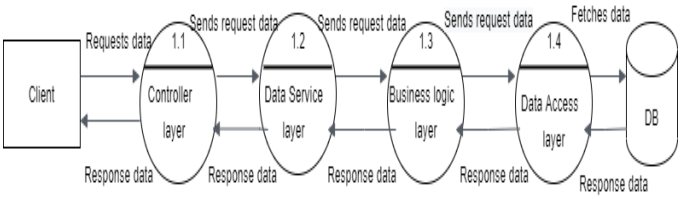
DFD Level 0:



DFD Level 0

Fig 3.2: DFD level 0 for enhancement of application

DFD Level 1:



DFD Level 1

Fig 3.3: DFD level 1 for enhancement of application

Sequence diagram:

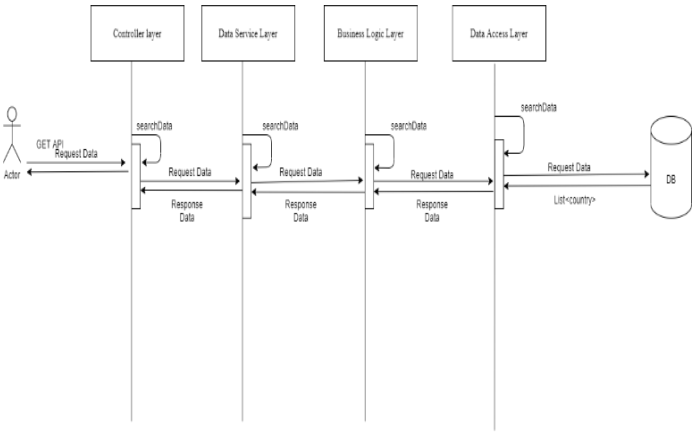


Fig 3.4: Sequence diagram of application

4. IMPLEMENTATION

Step 1] Lay out the foundation:

First, some background. We're not reinventing the wheel here. Facebook did the same thing across their entire ecosystem. It is possible.

The power of ReactJS is that it's a library, not a framework. This lets us work modular, migrating one component at a time until we can (eventually) achieve complete separation of UI and APIs, setting up the stage for a modern architecture. For more background on ReactJS, check out Pluralsight.

The first thing we need to do is identify a self-contained component of the app. This could be a new feature, either something easy or a pain point. You can write this new component using ReactJS.

There are two ways to integrate a new ReactJS component into an existing platform when its UI is rendered in the server and sent as an HTTP response. We have two options,

1] Rendering full pages

If the component that we are creating is a full-page, we can create an entirely new ReactJS site with its own URL and separate it from Spring UI. For this, we need a reverse proxy, which can be configured using Apache, Nginx, etc.

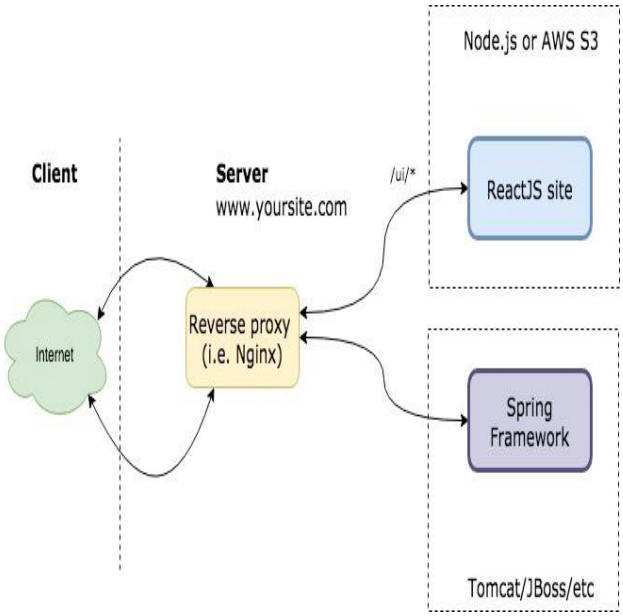


Fig 4.0: Reverse proxy configuration

2] Rendering specific components

In some cases, rendering a full page isn't an option. Sometimes our pages are too complex, and we want to start by migrating only specific components in specific pages.

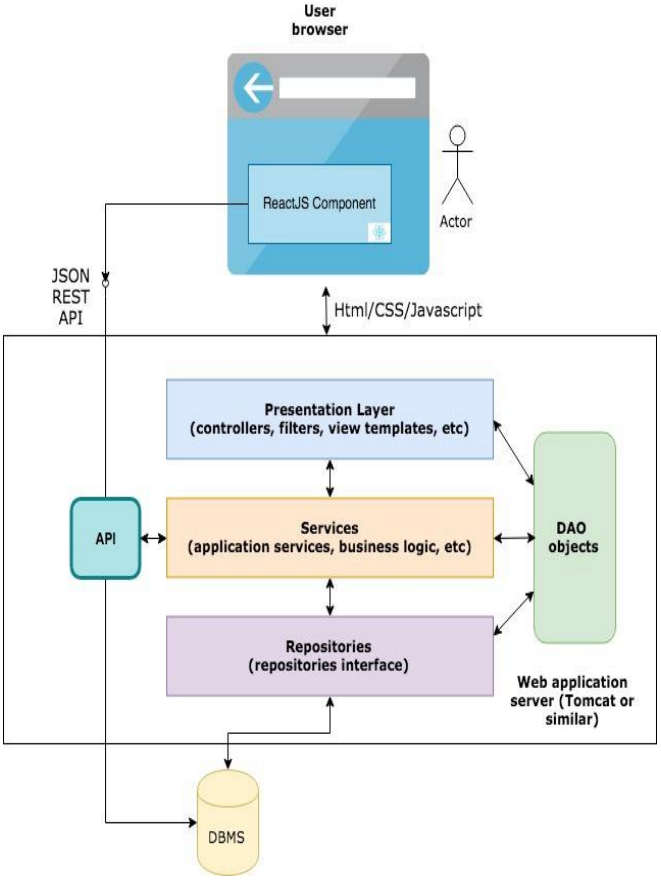


Fig 4.1: Identify one component, build the API for that and create a ReactJS component

Step 2] Iterate and Move Away from Legacy Code:

We must keep strong semantics in our APIs. If we identify that we are building an entirely new module, we can create a new API.

This is a step in the direction of a microservices approach. As we scale, these new APIs can be built using other languages or techniques, connect to different kinds of data modules (like a NoSQL database), or serve needs.

We have to keep clear documentation for each API (I recommend Apiary) and use our existing business logic (services layer) whenever necessary.

Also, we'll be able to create mobile apps and connect them to the same APIs. As a bonus point, we can use React Native and reuse some of our existing ReactJS components.

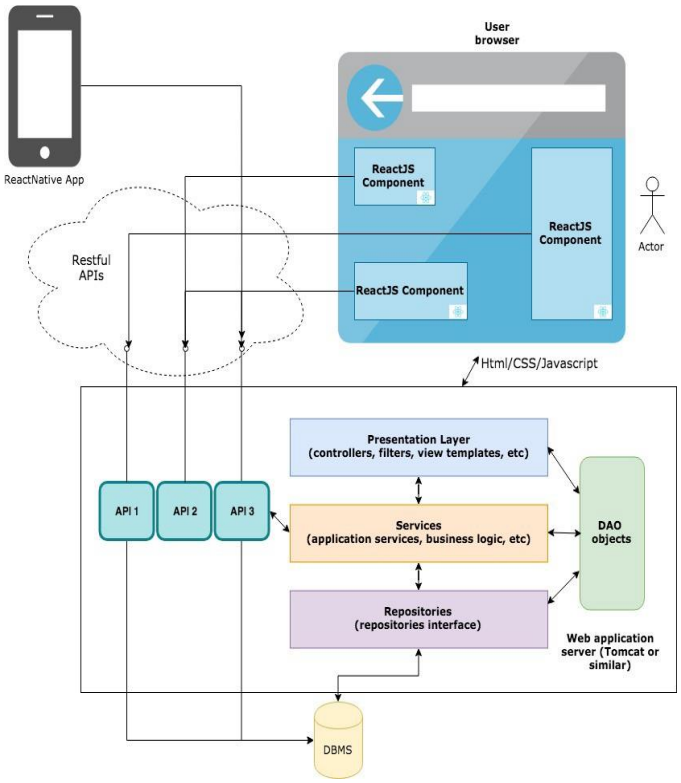


Fig 4.2: Expand APIs layer, create more ReactJS components and mobile apps

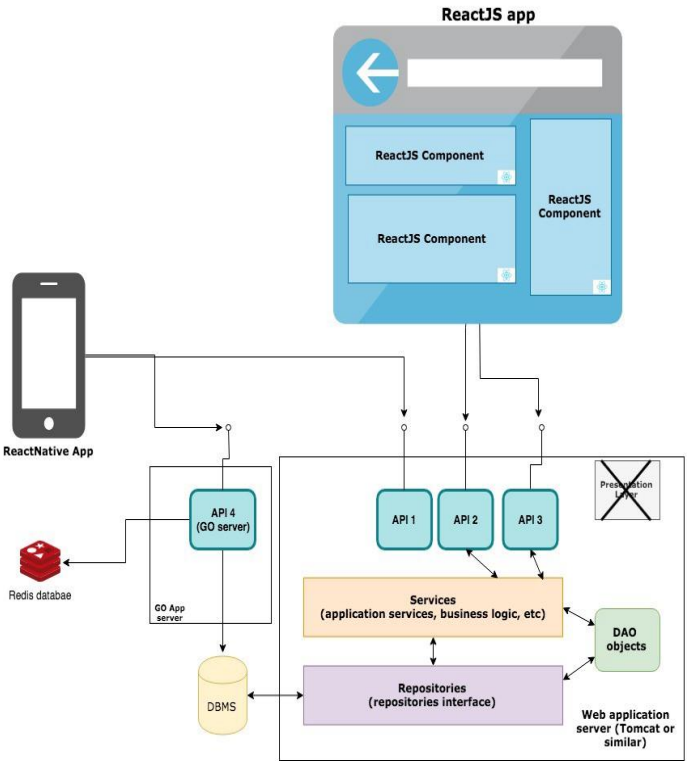


Fig 4.3: A modern API architecture and ReactJS web app.

Step 3] Deprecate Spring framework presentation layer:

Nowadays, software moves fast. If we are running quickly enough with agile methodologies, we'll be able to achieve an architecture that accomplishes all the goals we initially outlined much sooner than we'd expect.

Now, we'll have a web app 100% built using ReactJS, and we are cutting down the dependency to the presentation layer of Spring.

We have all our data exposed via Restful services, and as an example, the diagram contains a new API built using a different language (Go) and a NoSQL database (Redis).

For simplicity, some things are omitted. Any API may or may not access the DBMS, depending on the needs. Different ReactJS components will use different endpoints and different APIs.

It's OK if a component uses more than one API, but if it happens too much, it probably means that your API architecture needs review.

5. CONCLUSION AND FUTURE ENHANCEMENT

5.1 Conclusion:

Enhancement has been done to the registration module of the application. User can record patient's previous demographics such as patient's previous name and addresses and save in the database through API calls. User interface and backend has been modified to add extra functionalities successfully. The application is in align with new regulatory requirements.

Cloud-based solutions can offer benefits not available in other practice management systems, including:

• Continual, non-disruptive, behind-the-scenes upgrades that quickly respond to changes in industry requirements, from changes to individual codes (which can happen often with government payers) to a transition to an entire code set, as will happen with ICD-10.

• The ability to continually gather performance data from within the individual practice as well as from practices across the nation, not only helping to improve performance of the overall cloud-based solution, but also providing performance benchmarks for every practice on the network.

• Interconnection with other practices, hospitals, rehab centers, labs, and health care information sources.

3.2 Limitations:

The application that has been developed is supported by Windows operating system but not Linux operating system. This is the limitation of the project.

3.3 Future Enhancement:

• Upgrade the User interface to more interactive pages and develop components using latest technologies like ReactJS, AngularJS.

• The user interface can be made more interactive. The user interface can be uplifted from WPF to ReactJS to align with new technologies.

• Convert important functionalities into reusable API and integrate them into the existing design for better workflow.

6. ACKNOWLEDGEMENT

We would like to acknowledge the support provided by Teaching and Non-Teaching staff of Department of Computer Science & Engineering, RV College of Engineering through required assistance during the research work.

7. REFERENCES

[1] Abbott, M.L., Fisher, M.T, "The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise", Pearson Education (2009)

[2] Azhar Amir, "Visual Basic 6.0 to .NET Migration- White Paper", Infosys

[3] K. Gowthaman, K. Mustafa and R.A. Khan, 'Source Code Migration to DOT NET Framework: A Re-engineering Application Perspective', International Journal of Computer Information Systems, 2018

[4] Valjakka, Andreas, "A Reengineering Framework for the Migration of a Legacy Front End", Tampere University, Dec 2019.

[5] A. Aguilar, 'Planning a Successful Visual Basic 6.0 to .Net Migration:8 Proven Tips ', International Journal of Computer Information Systems, 2011

[6] Abbott, M.L., Fisher, M.T, "The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise" , Pearson Education, 2018

[7] Abel Avram, "Case Study: Migrating a VB6 Large Application to .NET ", International Conference on Web Engineering, 2011

[8] Cao Ronggui, "Progress and challenges of public hospital reform," Chinese Hospitals, vol. 14, no. 6,2010, pp. 1-5.

[9] Hu Xiao, Zhou Dian,and Wu Dan, "Analyzing the advantages and disadvantages of drugmarkon being cancelled in state-owned hospitals," The Chinese Health Service Management, vol.1, 2011, pp.32-25.

[10] A N Diwedi, "Tracking the healthcare management landscape: implications for integrated healthcare management and practice" Computer Applications, vol. 25, pp. 2817- 2819, Dec. 2005

[11] Gao Ang, Wei Wenxue, "Application of Java data persistence with Hibernate and Struts framework," Computer Applications, vol. 25, pp. 2817-2819, Dec. 2005

[12] Budi Kurniawan, Struts2 design and programming: a tutorial, BrainySoftware.com, 2008.

[13] Zhang Xi, Yu Shuju, "The research on interoperability of Spring framework," Computer Knowledge and Technology, pp. 1-5, Mar. 2008.

[14] Song Hanzeng, Shen Lin, "Simplification of Java database access with Hibernate service," Computer Applications, vol. 23, pp. 135-137, Dec. 2003.