

# Benchmarking Application Servers for Enterprise JavaBeans with the TPC-W Benchmark\*

Markus Richter  
[richter.werth@web.de](mailto:richter.werth@web.de)

University of Applied Sciences Gelsenkirchen, Campus Bocholt

## Abstract

*Since SUN released the Enterprise JavaBeans specification (EJB spec) as Java's server-side software component technology, it has strongly attracted the software developers community. Meanwhile, developers can already choose among a very long list of free or commercial application servers supporting EJBs. However, such a choice is far from being easy since application servers differ significantly with respect to performance and conformity with the most recent versions of Java's J2EE specifications (Java™ 2 Platform, Enterprise Edition).*

*This diploma thesis provides a complete benchmark environment that can easily be used to evaluate application servers. It is based on the well-known TPC-W benchmark that we have implemented using EJBs according to the EJB specs 1.1 and EJB 2.0.*

## Introduction

The work described in this paper is part of the research project CombaSoft (Component based Software Engineering) that investigates the possibilities of large-scale, component-based software development for Java's J2EE platform using EJBs [Mon-Haef]. The main goal of this thesis is to provide a benchmark environment for evaluating application servers with respect to the achievable performance and with respect to their conformity with various aspects of Java's J2EE specifications.

The benchmark environment should consist of an EJB based distributed application that has to be deployed to any application server in order to be benchmarked. Furthermore, a client application is needed that generates a well-defined workload for the application server and does all the performance measurements. We decided to use the TPC-W benchmark from the well-known Transaction Processing Performance Council (TPC) as the basis of our benchmark environment, since it is an exactly defined and widely accepted benchmark for web-based multi-tier applications. "The TPC-W is a transactional web e-commerce benchmark. The workload is performed in a controlled internet commerce environment that simulates the activities of a business oriented transactional web server [TPCW]."

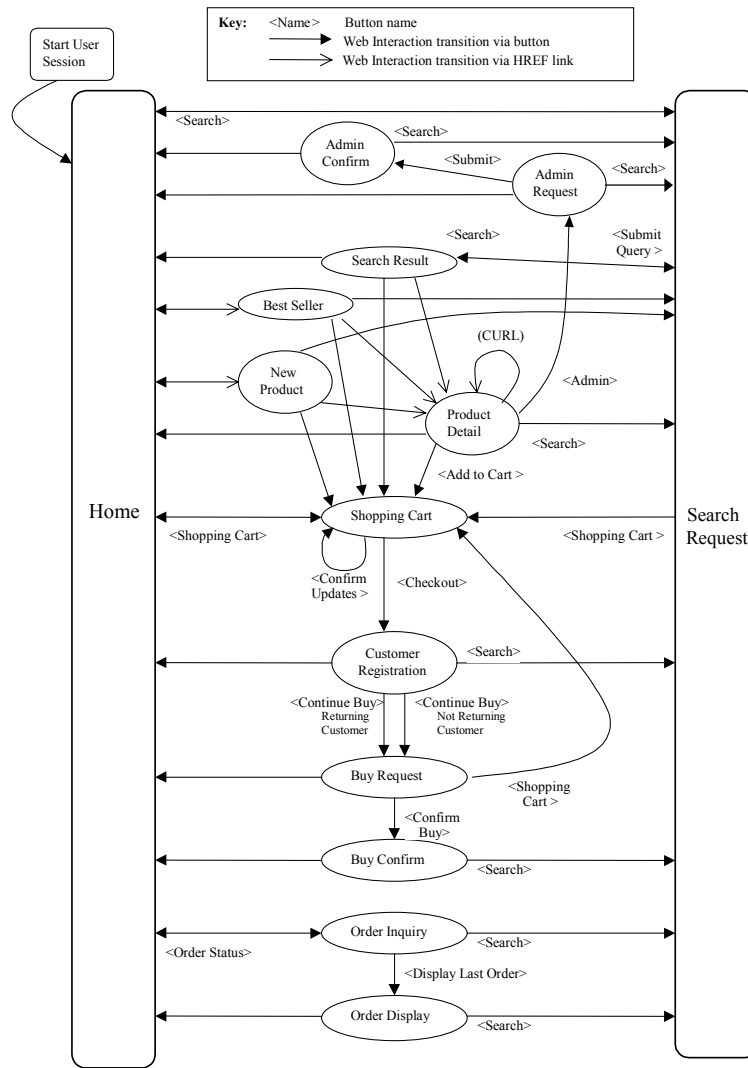
Roughly, the TPC-W benchmark simulates a simple retail online book-shop where customers can search and order books. The whole web application consists of 14 different web pages. Figure 1 gives an overview on required web pages and possible navigation paths (so called web interactions) in the form of a state transition diagram. Any user session starts with the home web page. Each active user session has an associated shopping cart which needs to be maintained by the system. The shopping cart is the logical component where users can add and remove selected books for later purchase. In order to generate the workload for this application several concurrent users are emulated that navigate through this web application and randomly perform possible web interactions. The performance metric reported by TPC-W is the average number of web interactions processed per second.

## Design of an EJB-based implementation

While the TPC-W benchmark exactly specifies the requirements of the underlying application and the way benchmark measurements have to be performed, the implementation details are not specified at all and left open so that different implementations using any available technology and platform can be used. I.E., our main task is to implement the benchmark using the EJB technology and to package our implementation as a portable benchmark environment that can be deployed to different application servers. We decided to base our implementation on the EJB spec 1.1 [EJB1.1], since most current application servers support this version of the specification.

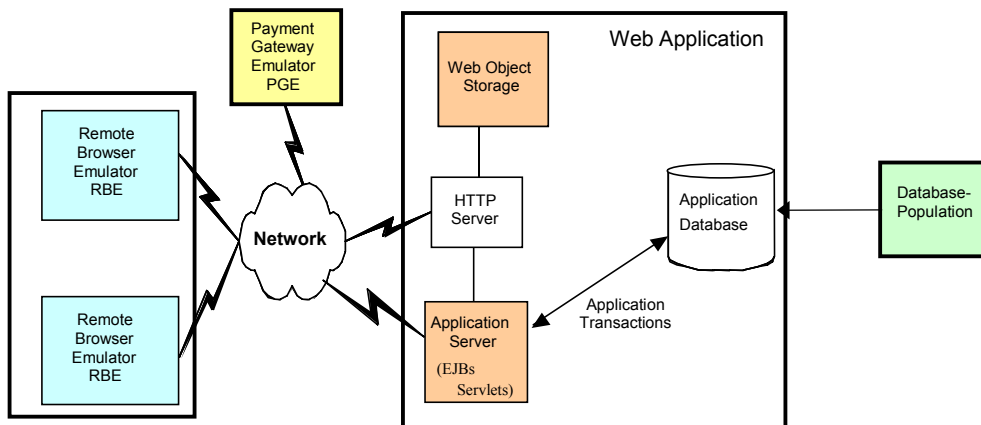
---

\* This work is supported as part of the "Forschungs- und Entwicklungsprojekt CombaSoft" by "Ministerium für Schule, Wissenschaft und Forschung in NRW" and by "Fördergesellschaft Fachhochschule Bocholt e.V."



**Figure 1. Web Interactions of the TPC-W Benchmark [TPCW]**

Figure 2 gives an overview of the architecture of our EJB-based TPC-W implementation. Roughly, the benchmark environment consists of four subsystems. Each of them will be discussed in more detail in the following sections.



**Figure 2. Architecture of the TPC-W Implementation**

The TPC-W benchmark, as the target web application, is designed as a multi-tier application. It consists of three layers. The presentation layer runs on a web server. The business logic layer runs on an application server, which provides the necessary runtime-environment for the server-side business-logic components. The data layer consists of a database. Multi-tier architectures allow pooling from resources and components. Application

servers provide this kind of feature, so that not each client gets its own components, as it would be in a 2-tier architecture.

### Description of Subsystems

The first subsystem deals with the *DatabasePopulation*. To generate the data in the database, the organization TPC provides some tools and programs which **MUST** be used! The main subsystem is the *Web Application*, which consists of EJBs and web pages. The EJB components are a very critical point of the design phase, since all web pages (designed as servlets) and web interactions have to deal with those components. Therefore each database table, as the permanent business data, is represented by a container managed persistence (CMP) entity bean. This provides a high level of flexibility to extend or modify the web application in the future. For example, it might be possible that the address bean should be exchanged with another address bean that contains more fields. Due to this design approach, all other related beans (e.g. customer) do not have to be re-generated.

The needed business processes, such as purchasing of a shopping cart or pricing a shopping cart, are realized by session beans. “Enterprise beans are intended to be relatively coarse-grained business objects (e.g. purchase order, employee record). Fine-grained objects (e.g. line item on a purchase order, employee’s address) should not be modeled as enterprise bean components.” [EJB1.1]. To stress the application server we will not follow this recommendation. While developing the web application we encountered some serious problems regarding the CMP finder methods. With EJB 1.0 or EJB 1.1 there is no way to specify complex finder rules within a single finder method in the Home-Interface of a CMP entity bean. For this purpose, we designed a further stateless session bean (“FinderService”) to do this kind of work.

The relationships between the EJBs are shown in Figure 3.

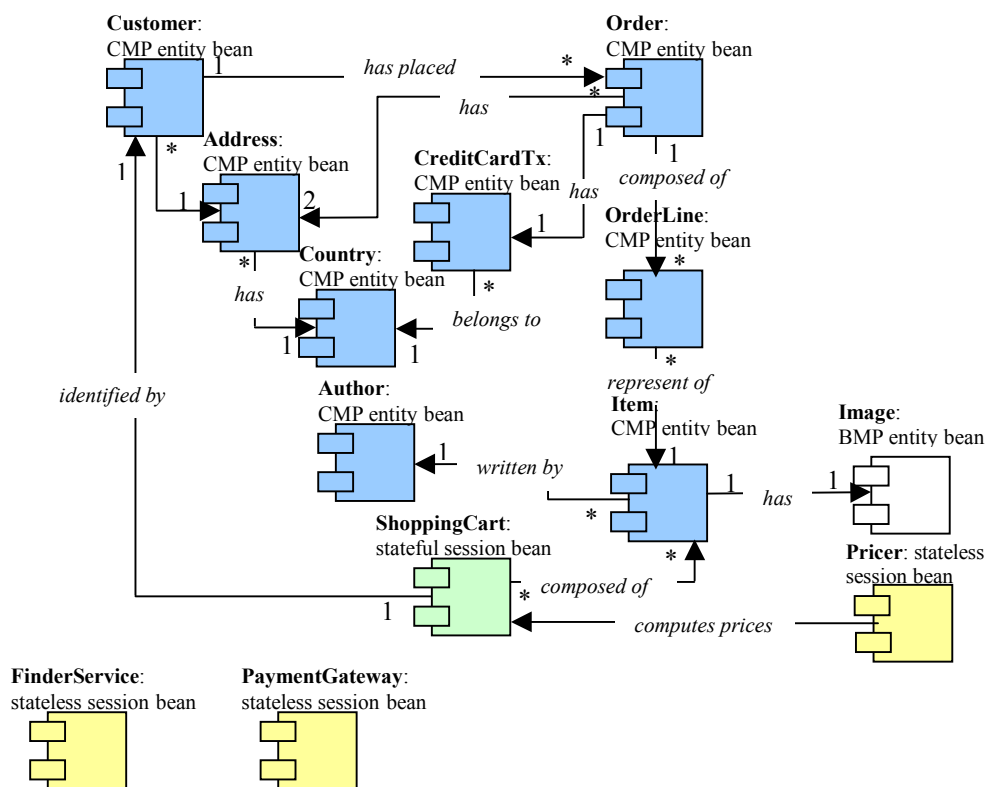


Figure 3. Relationships between the EJBs

An “Emulated Browser” simulates a single user session, just as a user would launch the web pages from a commercial browser, like Netscape or Internet Explorer. Furthermore it automates the navigation process through the different web pages. The *Remote Browser Emulator* is the workload generator. It starts multiple instances of the Emulated Browser to perform concurrency.

And finally we have the *Payment Gateway Emulator* as an external component. The task of this component is to verify a credit card number, generate an authorization ID and send the authorization ID back to the calling process.

### Some Benchmark Scenarios

Within a short period of time there has already been released several EJB specifications. Therefore, we developed different benchmark scenarios in order to evaluate some of the newly available features of the most recent EJB specs EJB 1.1 and EJB 2.0. We were especially interested in realizations of relationships between entity beans. Such relationships can for instance be implemented by storing serialized primary keys of related entity beans within the persistent part of a bean (bean-managed relationships). There are different ways to retrieve an EJB object with a given primary key. On the one hand the lookup can be done directly via JNDI. In EJB 1.0 this was the only way to refer to home objects and causes a lot of problems. For example, if a bean provider wants to call a home object, the developer needs to know the JNDI location. But “what if that JNDI location changes upon deployment, perhaps because the deployment is spread out across multiple domain boundaries?” [Roman].

The introduction of *EJB references*, offered by EJB 1.1, solves this problem. “An *EJB reference* is a deployment descriptor entry that says enterprise bean A uses enterprise bean B. When enterprise bean A is finally deployed, the deployer sees the EJB reference to enterprise bean B, and makes sure that enterprise bean B is available at the correct JNDI location. If enterprise bean B is located in a different enterprise domain, the deployer can add symbolic links within the JNDI tree using JNDI *LinkRef*.” [Roman]

With EJB 2.0 relationships can even be realized via a local home object or – as the most comfortable realization – such relationships can be maintained automatically by the container (container-managed relationships). “The local interfaces for session and entity beans enable the beans to be tightly coupled with their clients and to use pass-by-reference (rather than pass-by-value) semantics. The local interface is a standard Java interface that does not inherit from RMI. A bean uses the local interface to expose its methods to other beans that reside within the same container. It is thus possible to directly access a bean through its local interface without the overhead of a remote method call. Local interfaces provide the foundation for container-managed relationships among entity beans and session beans. The bean uses the local interface to maintain its references to other beans. For example, an entity bean uses its local interfaces to maintain relationships to other entity beans. Using local interfaces, beans can also expose their state and use pass-by-reference to pass their state between related bean instances.” [EJB2.0]

This leads to the following four scenarios:

- Scenario 1: bean-managed relationships by maintaining persistently the object’s primary key and accessing the object directly via JNDI (EJB 1.x)
- Scenario 2: bean-managed relationships by maintaining persistently the object’s primary key and accessing the object via EJB references (EJB 1.1, EJB 2.0)
- Scenario 3: bean-managed relationships by maintaining persistently the object’s primary key and accessing the object via the local interfaces (EJB 2.0)
- Scenario 4: container-managed relationships (EJB 2.0)

Please note that those scenarios are not a complete list of all possible strategies, especially regarding to bean-managed relationships!

The scenarios can be ported to different environments and application servers. In one configuration the application server can run on the same physical machine as the database server, *single server environment*, whereas in another configuration the application server and database server are on two different physical machines, *distributed environment*.

### First Experiences

We completely implemented our design and started first benchmarks in a simple single server environment where database server, application server and web server were all located on the same machine. We used different database management systems (IBM DB2 and Oracle) and tested different application servers (BEA WebLogic, version 5.1 and 6.1 and IBM WebSphere, version 4.01). Besides various benchmark results (see [Richter] for more details), we realized that porting a web application to other application servers isn’t as easy as the philosophy of SUN “Write once, run anywhere™” would suggest! Due to different supported EJB and servlet specifications even code changes are necessary, which require a re-compiling of the application. This contradicts the basic idea of J2EE applications!

### Perspective and future projects

As a result of our work, we provide a benchmark environment that can easily be used to evaluate application servers. This will help to establish a basic understanding of strengths and weaknesses of different products.

Benchmarking is an interesting but also a challenging task with respect to the acceptance of the results (see [Fle-Loe] and [Neumann] for other benchmarking with TPC-W and ECperf). Therefore, we will also use other available benchmark environments (see e.g. ECperf [ECperf]) in order to compare benchmark results obtained by different benchmarks on identical platforms. This will help to improve benchmark environments and to increase the acceptance of benchmark results.

As you can see, the Enterprise JavaBeans technology still changes quite dynamically. It remains to be seen how the application servers let the things simmer!

### References

- [ECperf] SUN: ECperf specification 1.0, Final Release (<http://java.sun.com/j2ee/ecperf/download.html>)
- [EJB1.1] SUN: EJB Specifications 1.1, *Enterprise JavaBeans™ Specification, v1.1* (<http://java.sun.com/products/ejb/docs.html>)
- [EJB2.0] SUN: EJB Specifications 2.0, *Final Release and EJB Specification 2.0, Proposed Final Draft 2* (<http://java.sun.com/products/ejb/docs.html>)
- [Fle-Loe] Marcus Flehmig, Henrik Loeser (Uni Kaiserslautern), *Ansätze der Nutzung von Erweiterbarkeitsmechanismen zur Anwendungintegration in ORDBS – Eine qualitative und quantitative Evaluierung*, BTW 2001 Datenbanksysteme in Büro, Technik und Wissenschaft, Springer, März 2001
- [Mon-Haef] Richard Monson-Haefel, *Enterprise JavaBeans™*, Second Edition, O'Reilly, March 2000
- [Neumann] Olaf Neumann (TU Dresden), *Vergleich von EJB-Applicationservern*, JavaForum Stuttgart 2001
- [Richter] Markus Richter, „Benchmarking Application Servers for Enterprise JavaBeans with the TPC-W Benchmark“, Diploma Thesis, February 2001
- [Roman] Ed Roman, *Mastering Enterprise JavaBeans™ and the Java™ 2 Platform, Enterprise Edition*, Wiley, 1999
- [TPCW] TPC-W Benchmark Specification 1.5 (<http://www.tpc.org/tpcw>)