Real-time Auction Service Application Based on Frameworks Available for J2EE Platform

Maciej Pilichowski, Bartosz Sakowicz, Jakub Chłapiński

Abstract: The article describes development process and operation of an advanced web application based on modern, popular frameworks available for J2EE platform. Capabilities of Spring, Hibernate, JSF and Rich Faces are considered based on auction service application.

Keywords: Java, J2EE, Hibernate, Spring, JSF, Rich Faces, JQuery

INTRODUCTION

Java Enterprise Edition platform offers much flexibility as far as implementation of certain application is concerned. This results in large number of frameworks that can help developers in their work. Frameworks discussed in the paper mainly worked in different areas however, their integration capabilities were very important as well. In order to create considered auction service, Hibernate, Spring, Java Server Faces and Rich Faces solutions have been used.

AUCTION SERVICE DESCRIPTION

Auction service considered in the article differs from the casual idea of auction (user paying the highest price wins). It provides two main types of auctions.

First one follows the rule, that user who gave smallest unique price wins and pays offered amount of money. Moreover this type of auction has a defined offer cost, i.e. price that a user has to pay for each placed offer. Finally, auction has a prescribed duration time, in which bidders can submit their offers.

Second auction type shares some of the ideas – each placed offer costs defined amount of money and duration time is defined at the beginning of the auction. However, the rule which determines auction winner is completely different. Every offer increases the price of a product by defined amount of money. Person which placed the last bid wins and pays the final price. What is more duration time also increases with every offer.

Second type of auction is much more challenging as development of the service is concerned. This is because of the fact, that duration time is measured in seconds and has to be updated in real-time as users would often like to place their offers very close to the end of auction. It forced usage of AJAX technology (allowing partial update of site content without the necessity to refresh the whole page), which was implemented with Rich Faces framework.

Obviously, site provides functionality common for modern web services like: on-line help, latest news about the service, registration page, survey allowing users to vote for products they like etc.

Also, following requirements have been worked out at the beginning of development process:

- clear design with separation of application layers – different frameworks mainly operate on different layers and such separation allow better evaluation of each of them separately,
- eye-catching user interface – used tools shall allow creation of interface that would attract potential users,
- real-time update of presented information – taking into account specification of second auction type, time to end and current price have to be refreshed at least every second,
- fast database access – a lot of information stored in the database must be presented to many users at the same time, moreover this data is a subject to frequent changes,
- short development time.

Fig. 1 presents main page of the auction service.

APPLICATION BASE

Spring is a starting point of development [5,8]. All other technologies are added to the project by integration with this framework. It is also a central point of application operation, as all the objects (beans) used, are created and maintained within Spring framework. Configuration used in this particular application differs from the standard one because some tasks which commonly are performed with Spring are passed to other specific frameworks. This happened in case of dispatcher servlet, which was replaced with its
counterpart provided by Java Server Faces – this resulted in following lines added to the WEB.XML file:

```xml
<servlet>
    <servlet-name>
        Faces Servlet
    </servlet-name>
    <servlet-class>
        javax.faces.webapp.FacesServlet
    </servlet-class>
    <load-on-startup>
        1
    </load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>
        Faces Servlet
    </servlet-name>
    <url-pattern>*.htm
</servlet-mapping>
```

Faces servlet is the core of all JSF based applications. It has access to the JSP pages that belong to the application and is responsible for building the component tree for them. Presented mapping obviously passes all the requests made to the server ending with suffix ".htm" to Faces servlet.

It replaces dispatcher servlet that comes with Spring MVC. Hence, the design pattern common for Spring MVC applications was not applied as the presentation layer completely belongs to the Java Server Faces. A lot of advantages can be derived from such a solution. First of all, there is no need to write controller for each model-view pair. Secondly, one can benefit from JSF navigation rules, that clearly describe the flow of control within the application. Finally, this change does not mean that MVC design pattern was abandoned. There still is a clear division between model (managed beans), view (JSP pages) and controller (Faces servlet and navigation rules).

Combining Spring and JSF technologies is a very common case and Spring framework provides necessary tool for further integration – SpringBeanFacesELResolver. It is a JSF el-resolver that delegates to the Spring root Web Application Context, resolving name references to Spring-defined beans [1]. Configuring this class in faces-config.xml file as the el-resolver, makes service layer beans managed with Spring accessible for JSF. This means, that they can be a part of JSF expressions and as a result they can be used directly on JSP pages.

**PERSISTANCE LAYER**

Persistence layer have been implemented based on Postgresql database. In order to make it cooperate with object oriented model, Hibernate framework has been used [3,4,7]. Following tasks have been realized by means of this solution:

- communication with database,
- mapping between tables and objects,
- CRUD (create, read, update, delete) operations,
- advanced queries invocation.

Hibernate and DAO design pattern that have been applied (Data Access Object – pattern that delegates certain layer of application to be the only one that interacts with database) saved a lot of time in each of the areas mentioned above. Mapping between objects and tables have been implemented with adding annotations to Java files. Developers did not have to access database directly, thus whole model could have been realized in an object oriented fashion. Simple operations like storing object, getting it by identifier etc. were implemented with only one line of code. But, probably the biggest strength of Hibernate was the possibility to form advanced database queries based on object oriented model. This is done with Hibernate Criteria and Hibernate Projections mechanisms [3]. Their exemplary usage within auction service is presented below:

```java
public List getActiveProductNames() {
    Criteria criteria = getSession()
        .createCriteria(Product.class);
    criteria.setFetchMode("shipping", FetchMode.EAGER)
        .add(Restrictions.eq("isactive", true))
        .setProjection(Projections.projectionList()
            .add(Property.forName("shipping.id")
                .as("shipping")));
    return criteria.list();
}
```

**SERVICE LAYER**

Methods belonging to objects from this layer perform business logic operations. These commonly involved data manipulation which is allowed by DAO layer. Usually, each service object has a member which is a corresponding data access object (for example BidderService class has a BidderDAO member). Dependency injection has been used for the purpose of object management, i.e. Spring framework initializes proper DAO component and injects it into a service object.

Moreover, explicit definition indicating object, which is to be injected is not necessary with Spring, instead auto-wiring feature have been applied. Each field that is to be initialized with this tool is preceded with @Autowired annotation, that makes Spring look for the component with name fitting the field type in the base package, defined earlier with the component-scan property. Spring provides also suitable annotations to indicate that a class belongs to service tier - @Service. Additionally @Transactional annotation has been used to assure transactions support. Service objects are the only ones visible for presentation tier. Because of this, any method that either returns or receives data from the user interface has to be implemented as a service method.

In simpler applications DAO and service layers can be merged into one. However, in case of this project such a division was fully justified and resulted in clear distinction between database access and any other actions performed with business objects. Complexity of service methods varies, though some of them simply encapsulate one DAO method.
while others can use many of them in a chain, and introduce business logic tasks in between.

PRESENTATION LAYER

Presentation layer for the application being a subject of this article have been based on the Java Server Faces and Rich Faces solutions [6].

JSP pages were used as a container for user interface components provided by other frameworks. In general they consist of HTML tags and scriplets which hold Java code allowing dynamic content. However, in the considered application, HTML language was used only for the purpose of proper arrangement of the objects on the page and scriplets were not present at all.

Java Server Faces technology provides a set of common user interface components. Due to the fact, that application’s look was designed in a highly customized way, only the basic UI components like outputText or graphicImage were placed on JSP pages. Still, functional tags from the JSF library like, form, loadBundle etc. where used extensively in the whole application. Generally, Java Server Faces technology provided a solid base for the presentation tier of a target application. Due to the fact, that it offers much more than only user interface components, appeared to be irreplaceable even in cases, in which they are coming from other tool. This is because of features like input field validators, navigation rules or multiple language support, that was be easily implemented in the considered auction service. Although, for this project UI components from Rich Faces framework have been used, the ones offered by JSF shall be enough for most of developers’ needs.

The user interface itself has been mainly implemented with Rich Faces framework because of its large set of rich user interface components, that surpasses the one that comes with JSF. It offers WYSIWYG editor, file upload component, tables with appendable data sources and many others. Moreover, those components are much more responsive as they take advantage of AJAX technology, with no need for the developer to write JavaScript code. What is also very important, those components are highly customizable as far as their look and feel is considered. Fig. 2 and Fig. 3 present data table component with default [2] and auction service specific look.

AJAX support was one of major arguments for choosing Rich Faces framework. Due to application specification, some of the information presented to the user had to be refreshed every second. Rich Faces provide a component designed for this purpose in the a4j library called poll [2]. This component periodically sends AJAX request to the server, and updates the page with data coming from the response. Usage of this component is very simple. Once it is placed inside a form (this is a requirement), its two parameters have to be set – interval and reRender. Interval parameter specifies time interval between requests sent in milliseconds, while reRender parameter content is a comma-separated list of identifiers of components to be re-rendered. Poll component have been introduced on each page, that needed real-time update of presented data, its implementation for refreshing table with latest bids from Fig. 3 looks as follows:

```xml
<a4j:form>
  <a4j:poll id="latestBidsPoll" interval="1000"
    reRender="latestBidsDataTable" />
</a4j:form>
```

During development process component performed very well. Data presented to the end user was updated properly. If some small lags happened they were probably caused by the poor performance of the hardware. After the application has been placed on the server for the purpose of tests it appeared that lags are growing fast, proportionally to the number of service users. Cause of this problem have been investigated, and results were surprising. With help of a Firefox web bowser with Firebug plug-in it was found out, that not only information that was to be re-rendered is sent within the response, but for example also whole table definition tags (<table>, <tr>, etc.). This resulted in large amount of data transferred, even if a small component with a few characters was to be updated. With only 10 users, service would not be able to send necessary data on time. Thus, AJAX support provided by the Rich Faces will have to be replaced with other technology before the application can be delivered to the users.

VII. CONCLUSIONS

To start with, all frameworks offer a lot of flexibility and they did not cause any problem as far as their integration is considered. Spring’s initial configuration does not seem to be costly when compared to benefits resulting from usage of

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chevrolet</td>
<td>S-10</td>
<td>43845</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>S-10</td>
<td>28766</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>S-10</td>
<td>36657</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>S-10</td>
<td>26467</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>S-10</td>
<td>25721</td>
</tr>
</tbody>
</table>

Fig. 2: Default look of data table component

Fig. 3: Data table component customized for auction service
this technology. Dependency injection and support of different life-cycles for components have been extremely useful during development.

Hibernate fully deserves its leading position among solutions for database access. This has been proved during the development of the auction service application. With Hibernate one can easily access data without SQL knowledge. Object oriented criteria API allows complicated queries to be implemented in a clear and easy way. Moreover, no noticeable lags did occur even in case of complex operations. Though Hibernate can be recommended as a very good ORM tool.

JSF is a technology that needs no introduction, because it is present on the market for a very long time. Its usage in this project proves that it still is a framework, that cannot be overestimated. Tag library and custom tag creation capability facilitate creation of the user interface. Navigation rules allow clear definition of flow of control in a web application, and the auction service application benefits from it.

Rich Faces is a quite new framework, but it is gaining more and more users thank to a large set of attractive and fully customizable user interface components. They also make application much more responsive due to AJAX support. However, this particular feature did not perform good enough to be implemented in a final auction service. User interface components were displayed and refreshed properly, but amount of data transferred form the server was not acceptable. As a further development a research have to be done in order to find alternate solution (eg. JQuery library).

ACKNOWLEDGEMENTS

One of the authors is a scholarship holder of project entitled “Innovative education ...” supported by European Social Fund.