Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44

Annales UMCS Informatica AI X, 2 (2010) 51-62
DOI: 10.2478/v10065-010-0052-6

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

# Usage of JSF framework and EJB technology in the creation of corporate applications

Przemysław Dębski*, Barbara Gocłowska†

*Institute of Computer Science, Maria Curie Sklodowska University,*
*pl. M. Curie-Sklodowskiej 1, 20-031 Lublin, Poland.*

**Abstract –** In the following article we describe the architecture of an online store project, which is an application utilising Java Enterprise Edition. Our project is based on a customer expectations model. The choice of technology has been due to its easy expandability by additional modules as well as its functionality which does not require reorganising the existing code.

## 1 Introduction

Over the last years the Internet has become one of the most important media. It has become a means for both static websites based on ordinary HTML and for the powerful applications. Creation of the latter may be a complex and time-consuming process. The implementation of IT based systems across the society has triggered a demand for a variety of tools for convenient creation of complex corporate applications.

There is a number of ready-made solutions available on the IT market. For instance, Enterprise Resource Planning applications support the management of a number of tasks performed by a company or groups of co-operating companies through gathering data or through making it possible to perform operations on the gathered data. Currently, the highest demand for Internet applications is particularly notable in companies (ranging from sole traders to powerful corporations).

*przemekdebski@gmail.com
†gocbar@gmail.com

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44


There are groups of companies, such as online stores, with certain features in common. They have a similar physical structure of a company which can be easily described by means of appropriate tools. One of such tools is the Java Enterprise Edition platform and its component technologies.

### 1.1 Why Enterprise Java Bean?

Enterprise Java Beans is a technology of creating components which perform certain tasks for the benefit of customers. It is a server-based technology, which makes it possible to transfer a significant share of business logic from client applications to server components. Thanks to this it is easier to make changes in the application and maintain it at the same time. EJB is a very complex technology comprising many modules responsible for, among others, data security or storage, providing tools for creating large, solid and portable business applications such as online stores.

Apart from the typical customer service, the technology streamlines financial management and staff management of a company. Moreover, it reminds the user about payment deadlines (Java Messenger Service, e-mail, a very convenient Timer interface) and makes it possible to use Web Services. Web Services can accelerate the speed and reduce the cost of integration with various internal applications and systems, and therefore they are particularly desirable in such applications [1].

### 1.2 Why Java Server Faces?

Java Server Faces is a framework which considerably simplifies and accelerates creation of large web applications. It has been created on the basis of Java Servlet Specification, which has entirely changed the attitude towards web application programming. In its structure JSF resembles window applications such as Swing. Its key features include: view creation based on the tree of ready-made user interface components, event-driven programming and data validation as well as conversion services. The programmer can focus on the implementation of an application's business logic thanks to ready-made mechanisms generating responses, checking data integrity as well as intercepting events.

### 1.3 Facelets or how to simplify the usage of JSF?

JSF can be associated with different technologies for creating view. The basic one is Java Server Pages which was created for different purposes before JSF. Consequently, JavaServer Faces cannot be fully used. An alternative solution is offered by a new Facelets framework created specifically for JSF. Its main advantage is the ability to create view based on templates, which cannot be achieved with JSP

### 1.4 Session Bean as JSF Servlet event listeners

Session beans may serve as event listeners in a JSF-based application. Event listener methods perform certain tasks, such as adding goods to the shopping cart by using

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44


EntityManager from Enterprise JavaBeans 3.0 in order to gain access to the database and return a text value which suggests which view should be generated in response.

## 2 Application of Java Enterprise Edition platform to create utility programs

The Internet is widely used as a medium for data presentation by online stores, among others. Files are dynamically generated after receiving a query from the server, which is linked with the necessity of choosing the right technology. The choices developers make are often based on their own preferences and past experiences. Nevertheless, literature on the subject comprises comparative studies which may be helpful in making such choices. Yet, the choices are not always favourable against the results of such studies. For example, the results of study [2] on "the cost" of using a given web application within various technologies have shown that Java Servlets are one of the most expensive (compared to CGI/C++ among others). The authors of the study have named several reasons for this and one of the most significant is the servlet interpretation by Java Virtual Machine. On the other hand, the authors point to advantages of using servlets, especially with applications which are not highly complex. Moreover, if this is accompanied by the use of JSF and Facelets technologies, thanks to which the work of a team creating an application may be efficiently organised, the choice seems to be justified. It is also possible to look for convenient but faster solutions such as Portlets [3, 4] or even to use code generating frameworks such as SEAM or Ruby on Rails. Some [5] use the UML tool to accelerate the process of creating applications. However, the ability to use metadata as well as the CRUD mechanism of entity beans EJB 3.0 has eliminated such necessity.

## 3 Online store application

Designing a user friendly online store is the most important issue. As [6] indicates "online shopping Web Sites contain a lot of irrelevant information related to new types of products or reduced items". Customers get confused by details and become impatient being unable to intuitively find relevant information. Another common problem is posed by bad navigation due to which it is easy to get lost in cyberspace [7].

Whereas in creating educational platforms [8] the main focus is on tailoring them to the customer's cognitive requirements, in the implementation of service application the main focus is on the customer's expectations, which has been proved by empirical study [9].

(1) Product categorisation – choice of category.
(2) In the case of a large number of categories, requiring more attention than just a glance, for instance in a drop down menu – grouping and sub-categorisation.
(3) Exclusion of the possibility to get lost in cyberspace.
(4) "Memorising" the customer's preferences.
(5) Avoiding information overload on the website.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44


(6) Good timing for loading another navigated webpage (according to research [10] if it takes longer than a minute for a webpage to load, the application user will not browse it).
(7) The website has to provide detailed product information "on request".

We have tried to utilise the above-mentioned customer expectations model in the application described below. To deal with the issue of speed we have used the rendering of single or multiple UI components, which has reduced the number of loaded pages and complied with point 5 above.

### 3.1 Application's functionality

The function of the project is an online photographic equipment store. The application is divided into customer and administrator modules. Each of them comprises a separate web interface created on the basis of JSF Framework + Facelets. The customer module's functionality allows customers to browse the catalogue, add products to the shopping cart, order selected items, express opinions on them as well as browse their shopping history. The administrator's interface makes it possible to create product categories, add or edit items.

The application has been designed in a way which makes it possible to change the store's profile. Each catalogued product comprises items belonging to a given category. Each category has a set of attributes which are typical of the items belonging to it. The administrator can define them, thanks to which the application is universal and may be used for cataloguing products from a variety of fields.

### 3.2 Design and Application

The application's architecture is represented by the picture below. Its core comprises the EJB module which serves as a link between web clients and the database. It can be divided into two basic groups: session beans and entity beans. Web clients only gain access to session beans which make use of the entity beans and these, in turn, serve as a database abstraction.

#### 3.2.1 Entity beans

Entity beans are used to translate the database tables into Java objects. The application comprises 12 entity classes.

AdminUser – store administrator.

Customer – store customer.

Category – item category (Item) or product category (Product).

Item – single item (e.g. a camera or lens); items comprising a catalogued product.

Attribute – an item's attribute (e.g. pixel count for the camera sensor); each category has a given set of attributes.

AttributeValue – attribute value of a particular item; value is of the the kind java.lang.String, and its interpretation is expressed by AttributeValueType.

AttributeValueType – logical type of attribute value; four types have been defined: INTEGER, FLOAT, BOOLEAN, STRING.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44



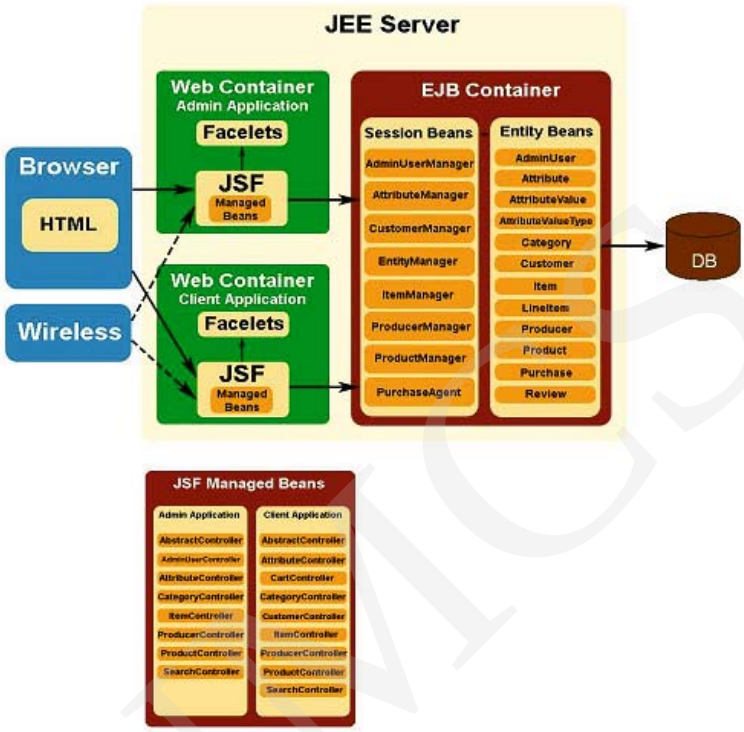


Fig. 1. System architecture.

Producer – producer of an item.

Product – catalogued product, it has its price as well as quantity in stock, comprises one main item (of the same category as the product) and optional additional items, e.g. a camera (main item) + lens (optional additional item).

LineItem – item in an order; it comprises the item, its price at the moment of purchase as well as quantity ordered.

Purchase – order; it comprises items (LineItem), information on the customer, payment method, price and shipping address.

Review – the customer's opinion on a product.

#### 3.2.2 Session beans

Session beans are responsible for the application's business logic and serve as an interface between the client application and the database. Certain operations multiplied in entity classes have been abstracted to a stateless session bean EntityManagerBean.

This bean comprises a set of typical methods used in applications utilising a database. Each of the methods performs a calculation whose result is independent of other methods' results, which justifies the usage of the stateless session bean.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44


The other session beans comprise methods dedicated to particular entity beans. The methods include, among others: searching according to specific attributes of a given entity, saving preceeded by checking whether or not the uniqueness of a certain attribute will be impaired (e.g. producer's name should be unique) etc. The method ItemManagerBean.search() may serve as a good example. Its task is to search all items whose attributes' values have been defined in a template.

```
public List<Item> search(Category category, Map<Attribute, Object>
    attrval) {
  <Item> items = findByCategory(category);
  if (attrval == null || attrval.isEmpty())
    return items;
  List<Item> result = new ArrayList<Item>();
  for (Item i : items) {
    boolean match = true;
    for (Attribute a : attrval.keySet()) {
      boolean attrFound = false;
      Object o = attrval.get(a);
      for (AttributeValue avi : i.getAttributes()) {
        if (! avi.getAttribute().equals(a))
          continue;
        attrFound = true;
        if (o instanceof String) {
          String av = (String)o;
          if(!av.trim().equalsIgnoreCase(avi.getValue()))
            match = false;
        } else if (o instanceof StringRange) {
          StringRange av = (StringRange)o;
          if(avi.getAttribute().getValueType().getType().equals("
              INTEGER")){
            Integer avfrom=null, avto=null, aval=null;
            if (! av.getFrom().isEmpty())
              avfrom = new Integer(av.getFrom());
            if (! av.getTo().isEmpty())
              avto = new Integer(av.getTo());
            aval = new Integer(avi.getValue());
            if (avfrom!=null && avfrom.compareTo(aval)>0) {
              match = false;
              break;
            }
            if (avto!=null && avto.compareTo(aval)<0) {
              match = false;
              break;
```

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44



```
            }
          } else if(avi.getAttribute().getValueType().getType().
              equals("FLOAT")){
            ..............................................
          }
        }
        break;
      }
      if (! attrFound) {
        match = false;
        break;
      }
      if (! match)
        break;
    }
    if (match)
      result.add(i);
  }
  return result;
}
```

The method returns arguments comprising a category and a map of attributes with values according to which items should be searched. Attribute values are saved in the form of a string but each of them takes one logical value of the following four types INTEGER, FLOAT, BOOLEAN or STRING. In the case of last two, the value passed to the map is of type String. In the case of INTEGER and FLOAT the value is of type StringRange.

The reason for using it is the fact that for digital attributes it is possible to set a range of values circumscribing the values of such attributes.

Initially all items belonging to a given category are searched. If the map of template values is empty the method returns the items found and becomes inactive. Otherwise, each of the items is analysed by comparing its attributes with template attributes. For attributes of type BOOLEAN and STRING the analysis involves searching for identical values. For digital types the analysis checks whether their value is higher or equal to the value of from field of an object of type StringRange and whether it is lower or equal to the value of to field. If an item does not have value set for an attribute from the template map or its value does not match the value from the template, the analysis of a given item stops and proceeds to another one. If the analysis of an item is successful, it is copied to a list which will be returned as a search result.

CustomerManagerBean.add()is a method of persisting a new entity bean (in this case Customer) in the database with a prior examination whether the persistent entity will not interfere with uniqueness limit (limits are imposed on email field in Customer entity, among others).

merge()operation of EntityManager persistence service only checks the uniqueness of the primary key. When an attempt is made to persist an entity interfering with the limitations

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44


imposed on a different column, the database will return an error. Java Persistence 1.0 specifications do not define a separate exception for such a situation, therefore in case of failure persistence service will not indicate its cause. The solution offered by add()method is to perform a query checking whether the record with the value of the indicated column and uniqueness limit exists in the database. If the search result is positive the method generates the exception UniqueException, otherwise the entity becomes persistent.

### 3.2.3 Customer applications

Customer modules are in the form of web applications. We have created two separate ones: one for customers and the other for store administrators. They have been built on the basis of JavaServer Faces technology combined with Facelets framework and may be divided into management and presentation layers.

The management layer comprises managed beans, also referred to as controllers (figure 1). Initializing particular controllers is performed declaratively by means of appropriate XML implementation descriptor elements faces-config.xml.

Controllers implement all operations which the application offers its users. Access to those operations is given by means of links and forms on the viewed website. The controller object uses session beans injected by @EJB annotations, thus gaining access to data gathered in the database.

Each controller is an extension of the abstract class AbstractController. This class comprises references to the session component's interface EntityManagerLocal used in all the controllers. Another feature of the class is view which is a String type object storing information on view. AbstractController has one getEntityFromRequest() method to get an entity object according to the HTTP query parameter (parameter value should constitute the primary key of searched entity).

```
public abstract class AbstractController {
  @EJB
  protected EntityManagerLocal em;
  private String view;
  public String getView() { return view; }
  public void setView(String view) { this.view = view; }

  public <T> T getEntityFromRequest(Class<T> objtype, String param) {
    T obj = null;
    String p = FacesContext.getCurrentInstance().getExternalContext()
        .
    getRequestParameterMap().get(param);
    if (p != null) {
      int id = Integer.parseInt(p);
      obj = em.get(objtype, id);
    }
    return obj;
```

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44



```
    }
}
```

ProducerController from the store administrator application may serve as an example of a controller. This class comprises a number of methods utilised in the InvokeApplication phase of the JSF query processing cycle. Their function is to initiate parameter states so that their values may be displayed or set by means of forms on the viewed webpages. For instance, the createSetup() method has to perform the following tasks:

(1) it creates a new object of type Producer whose attributes will be set by means of a form on the viewed webpage,
(2) it sets the value of view parameter to CREATE, which means displaying the webpage with the form in order to add information concerning the producer,
(3) it returns the value of type String, it is navigational information whose task is to define the page returned as a search result.

After filling in the form, sending it to the server, passing all the validation phases and being converted in the InvokeApplication phase, the method save() is performed. The method does the following:

(1) by invoking the method save()of an entity bean, it tries to save a new record, reflecting the producer entity, in the database,
(2) in case of failure it generates a message to the user and returns an empty writing informing JSF about displaying the same view,
(3) if the operation of persisting succeeds, the parameter view will assume the value LIST – i.e. displaying the list of all the producers – and the navigated value will be returned.

Facelets framework has been used to create the presentation layer because it allows for creating view in a convenient way based on templates. XHTML has served as a standard for creating each component file adding up to the ultimate appearance of a webpage. Each page makes use of a template defined in the layout.xhtml file which is presented in the code snippet below (certain elements responsible for the appearance of the webpage have been removed for the sake of clarity):

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   .........
  xmlns:ui="http://java.sun.com/jsf/facelets">
.........
<body>
  <!-- elementy definiujace uklad i wyglad strony -->
  <div id="center_content">
     ............
    <ui:insert name="mainbody"/>
```

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44


```
      ...........
    </div>
    <div id="footer">
      ........
    </div>
  </body>
</html>
```

The document describes general appearance and common elements for each page (menu, footnote) as well as separates a variable section with the help of <ui:insert name="mainbody"/> whose content varies from page to page. The documents describing separate pages define the appearance and content of this section only. Thanks to the usage of templating mechanism it is much easier to maintain the code and to make changes. If we wanted to, for instance, change the position of certain elements on the webpage, we would only have to edit the template file and the changes would be automatically introduced in all the subpages utilising it. We would be deprived of such a possibility if we had used the standard way of creating view in JSF, namely JSP.

The content itself varies on the majority of pages and it depends on the user's action. In such cases the main document of a webpage – depending on the view parameter of the suitable controller – imports the document defining the content of the webpage. An extract from the file producers.xhtml from the administrator application may serve as an example:

```
<ui:composition template="/WEBINF/templates/layout.xhtml">
  <ui:define name="mainbody">
    <h1>Katalog :: Producenci</h1>
    <c:if test="#{producerController.view == 'LIST'}">
      <ui:include src="/WEBINF/templates/producers_list.xhtml"/>
    </c:if>
    <c:if test="#{producerController.view == 'CREATE' ||
        producerController.view == 'EDIT'}">
      <ui:include src="/WEBINF/templates/producers_edit.xhtml"/>
    </c:if>
  </ui:define>
</ui:composition>
```

The element <ui:composition> points to the template used by the webpage. <ui:define> points to the variable section of the template. Its content is the same as the content of the section. In this case the content is not directly set in <ui:define> element but rather divided into two separate files. If the requested view is the product list (LIST), the file producer_list.xhtml is imported, if the administrator wants to create a new product (CREATE) or make changes in an existing one (EDIT), the file producers_edit.xhtml is imported.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44


## 4 Summary

The application has been designed and implemented in a way which enables it to be used as a template for convenient creation of online stores. We have achieved this, among others, thanks to utilising general entity classes which we have transposed into the tables of a relation database. Such indispensable elements of successful online retailing as a shopping trolley service, credit card validation service and many others are also present. However, the most important element is strict adherence to "desirable expectations" according to the store customer model. Data protection is also a notable aspect. We have not used typical technologies such as Java Authentication and Authorization Service [11], but focused on the methods offered by the PhaseListener interface: beforePhase() and afterPhase(), which, combined with the roles assigned to particular groups of application users, supervise redirecting unlogged and unregistered users to the webpages they are permitted to view. The architecture of the application allows for easy extension by an accounting module or online store staff management module as well as to create such modules as a service module, etc.

## 5 Future work

Further work on the application will concern utilising the rule system for giving discounts to customers, dealing with their interests and including external data if customers are interested in equipment which is out of stock.

We are also considering adding customer support while shopping. For instance, it could include suggestions to increase the practical value of a product purchased.

Due to our concern about the effectiveness of online retailing, we are of the opinion that i t is necessary to conduct a survey on customer "satisfaction" based on the model described in section 3.

## References

[1] Huang Y., Chung J-Y., A Web services-based framework for business integration solutions, Electronic Commerce Research and Applications 2 (2003): 15–26.

[2] Aptea V., Hansenb T., Reeser P., Performance comparison of dynamic web platforms, Computer Communications 26 (2003): 888–898.

[3] Coronato A., d'Acierno A., De Pietro G., Automatic implementation of constraints in component based applications, Information and Software Technology 47 (2005): 4970–509.

[4] Minbo Lia, Jianguo Wangb, YokeSan Wonga, Kim Seng Leea, A collaborativeapplication portal for themould industry, Int. J. Production Economics 96 (2005): 233–247.

[5] Coronato A., d'Acierno A., De Pietro G., Automatic implementation of constraints in component based applications., Information and Software Technology 47 (2005): 497–509.

Pobrane z czasopisma Annales AI- Informatica **http://ai.annales.umcs.pl**
Data: 23/12/2024 14:42:44


[6] Dominguea D., Stutta A., Martinsa M. et al., Supporting online shopping through a combination of ontologies and interface metaphors, Int. J. Human-Computer Studies 59 (2003): 699–723.

[7] Brusilovski P., Adaptative navigation suport In educational hypermedia: the role of student knowledge level and the case for meta-adaptation, British Journal of Educational Technology 34(4) (2003): 487–497.

[8] Gocłowska B., Łojewski Z., Intelligent tutorial system LISE, Annales Informatica 5 (2006): 441–452.

[9] Lohse G. L., Spiller P., Quantifying the effect of user interface design features in cyberstore traffic and sales (Los Angeles, 1998).

[10] Gocłowska B., Futa G., Analysis of functionality of distance learning platform moodle, Informatica (2006).

[11] Merz M., Enabling declarative security through the use of Java data objects, Science of Computer Programming 70 (2008): 208–220.

[12] Schalk C., Burns E., JavaServer Faces: the complete reference (McGraw-Hill, 2007).

UMCS