Master Thesis Project
Design and implementation of an Intranet

Net Entertainment AB

Stefan Ekenkrantz
c00sez@cs.umu.se

Internal supervisor: Stefan Johansson
External supervisor: Anders Callertun
Examinator: Per Lindström
Abstract

This master thesis describes the process and the results of developing an Intranet for Net Entertainment AB (a member of the Cherry Group) using JavaServer Pages. It defines the goals and requirements for the project and how they were accomplished. It also contains a study in Java2 Enterprise Edition and JavaServer Pages as well as a comparison between JavaServer Pages and other techniques to build dynamic web pages. At the end of the report the resulting pages of the Intranet are presented.
## Contents

1 Introduction  
1.1 Background ............................................. 1  
1.2 Disposition of this report .......................... 1  

2 Problem description and goals  
2.1 General functionalities requirements ........ 1  
2.2 Functional requirements specification ........ 2  
2.3 Delimiters ........................................... 2  
2.4 Open issues ......................................... 2  
2.5 Project organization ............................... 2  
2.6 Acceptance criteria ............................... 3  
2.7 Project close ..................................... 4  

3 In-depth study of JavaServer Pages  
3.1 The Java2 Enterprise Edition .................... 5  
3.2 J2EE Applications .................................. 6  
3.3 Servlets ............................................ 9  
3.3.1 Structure of the Servlet ..................... 10  
3.3.2 The Servlet lifecycle ....................... 10  
3.3.3 Basic Servlet example ....................... 11  
3.4 JavaServer Pages .................................. 12  
3.5 JSP Tags ............................................. 13  
3.6 Enterprise JavaBeans ............................. 14  
3.6.1 Entity Beans .................................. 14  
3.6.2 Session Beans ................................ 15  
3.6.3 Message-driven beans ....................... 15  
3.7 Databases and JSP .................................. 15  
3.7.1 JDBC Drivers .................................. 17  
3.8 Security and JSP .................................... 18  
3.8.1 Authentication .................................. 19  
3.8.2 Confidentiality and Data integrity ........ 19  
3.8.3 Security credentials ......................... 20  
3.9 Comparison of technologies for creating dynamic web content ........................................ 21  
3.9.1 Common Gateway Interface .................. 21  
3.9.2 JavaServer Pages .............................. 21  
3.9.3 mod perl ........................................ 22  
3.9.4 Active Server Pages ......................... 22  
3.9.5 Hypertext Preprocessor ...................... 22  
3.9.6 Conclusions .................................... 22  

4 Accomplishment ........................................ 24  
4.1 Schedule of project ................................ 24  
4.2 Changes and actual distribution of time ........ 25
5 Implementation
5.1 Software tools used during the project .................................. 27
5.2 Java classes ................................................................. 27
5.3 Design of database ......................................................... 29
  5.3.1 The employees table ............................................... 29
  5.3.2 The companies table .............................................. 30
  5.3.3 The worksatcompany table ...................................... 31
  5.3.4 The groups table .................................................... 31
  5.3.5 The ingroup table .................................................. 31
  5.3.6 The eventtypes table ............................................. 31
  5.3.7 The events table ................................................... 31
  5.3.8 The messages table .............................................. 31
  5.3.9 The backofficestaff table ....................................... 32
  5.3.10 The supportedlanguages table ................................ 32
  5.3.11 The knowslanguage table ...................................... 32
  5.3.12 The shifts table .................................................. 32
  5.3.13 The boschedule table .......................................... 32
  5.3.14 The casinos table .............................................. 32
  5.3.15 The promotions table .......................................... 33
  5.3.16 The inpromotion table ........................................... 33
  5.3.17 The projects and inproject tables ............................ 33
  5.3.18 The customers table ............................................ 33
  5.3.19 The contacts table ............................................. 33
  5.3.20 The companyaccounts table ................................... 33
5.4 The groups in the Intranet ............................................ 34
5.5 Security model ........................................................... 34
5.6 A typical request example ............................................ 35

6 Future work ............................................................... 36

7 Acknowledgements .......................................................... 36

A Net Entertainment Intranet user’s guide .................................. 41
A.1 The login prompt ........................................................ 41
A.2 The menu ................................................................. 41
A.3 The jump menu .......................................................... 42
A.4 News page ............................................................... 42
A.5 Customer contacts page ............................................... 43
A.6 Company accounts page .............................................. 43
A.7 Other pages in the documents menu ................................ 43
A.8 Staff listing page ........................................................ 45
A.9 Edit staff page .......................................................... 45
A.10 Intranet groups page .................................................. 46
A.11 Vacations page .......................................................... 46
A.12 Promotions page ....................................................... 49
A.13 Jour page ............................................................... 49
1 Introduction

1.1 Background

The project started with that Net Entertainment AB in Stockholm decided to implement an Intranet for their employees. The project runs as an in-house development project at Net Entertainment and as a Master Thesis Project for Stefan Ekenkrantz, student at Umeå University, between February 1st 2005 and June 30th 2005. It will result in a release 1 of the solution. If time permits, functionality from what has been defined as release 2 will be implemented as well (according to prioritisation). The goal with the Intranet is to make information available to all staff members of Net Entertainment in order to make the company more effective. The Intranet will primarily serve as a source of information.

1.2 Disposition of this report

The remaining report is organized as follows. Section 2 defines the problem description, goals and all the functional requirements for the project. Section 3 of the report contains an in-depth study of JavaServer Pages, which will help the reader to understand how Java2 Enterprise Edition(J2EE) applications are designed. How the project were accomplished is described in Section 4, and Section 5 contains a report of the results of the project. In Appendix A, there is a user’s guide for the implemented Intranet.

2 Problem description and goals

This section describes the purpose, goals and requirements of the project. The main parts are Section 2.2 which holds a brief table which describes what the requirements of the project are, and Section 2.6 and 2.7 which defines what must be fulfilled before the project can be closed down.

2.1 General functionalities requirements

Before the design of the Intranet started, some basic functionality requirements for the Intranet was set up. These where:

- The Intranet should be a very simple solution. Functionality should be very restricted.
- The functionality for editing information in the pages of the Intranet is to be decided by the Project group.
- The Intranet should be password protected.
2.2 Fuctional requirements specification

Table 1 describes the requirements for the project, and were set up by Viktoria Nilsson, project manager at Net Entertainment AB. The different pages are described very briefly. The column Administration feature defines for each page whether it should be managed by an administrator or not. The release 1 and release 2 defines how important it is that the page is implemented. The most important pages will be implemented in the first release, while others will be implemented in the second release. Notice that “NE” in the table is an abbreviation for Net Entertainment AB.

2.3 Delimiters

The Intranet will have a few limitations. The ones specified before the start of the project was:

- Only one language supported: English.
- No graphical interface will be provided. User interface will be similar to the backend/admin interface of the Casino Module¹.

2.4 Open issues

The one and only open issue for the project before it started was:

- Should the Intranet be accessible over the Internet, or only from within the internal network?

2.5 Project organization

There are a few persons involved in this project. Below is a table which describes what role they have in this project.

<table>
<thead>
<tr>
<th>Steering committee</th>
<th>Pontus Lindwall, Anders Callertun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project owner</td>
<td>Anders Callertun</td>
</tr>
<tr>
<td>Reference group</td>
<td>Pontus Lindwall, Annelie Jonsson, Viktoria Nilsson</td>
</tr>
<tr>
<td>Project manager</td>
<td>Stefan Ekenkrantz</td>
</tr>
<tr>
<td>Systems developer</td>
<td>Stefan Ekenkrantz</td>
</tr>
</tbody>
</table>

¹Casino Module is a product of Net Entertainment AB.
Table 1: Functional requirements specification for the Intranet.

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
<th>Administration feature</th>
<th>Release 1</th>
<th>Release 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Budget</td>
<td>A compact version of NE’s budget. This will give all employees the possibility to follow what the company should achieve.</td>
<td>No</td>
<td>Link to document</td>
<td></td>
</tr>
<tr>
<td>Policies</td>
<td>Relevant policies, such as staff policy.</td>
<td>No</td>
<td>Links to documents</td>
<td></td>
</tr>
<tr>
<td>Forms</td>
<td>Downloadable forms for travel, expenses, car mileage and time reports.</td>
<td>No</td>
<td>Links to documents</td>
<td></td>
</tr>
<tr>
<td>Company Accounts</td>
<td>Account information for FedEx, UPS accounts etc.</td>
<td>No</td>
<td>Static html page</td>
<td>Database connected</td>
</tr>
<tr>
<td>Staff members</td>
<td>Contact list of all staff members</td>
<td>Release 2</td>
<td>Static html page</td>
<td>Database connected</td>
</tr>
<tr>
<td>Customer contacts</td>
<td>Business and technical contact person for each customer.</td>
<td>Release 2</td>
<td>Static html page</td>
<td>Database connected</td>
</tr>
<tr>
<td>Projects</td>
<td>Open and closed projects. Divided in project name, status, time due, responsible, developers.</td>
<td>Add, edit and delete projects</td>
<td>Database connected</td>
<td></td>
</tr>
<tr>
<td>Assignments</td>
<td>Status on each assignment, customer, responsible developer, planned test and launch dates. Divided in project name, status, time due, responsible, developers.</td>
<td>Add, edit and delete projects</td>
<td>Database connected</td>
<td></td>
</tr>
<tr>
<td>Time reporting</td>
<td>Simple time reporting tool for projects (project name, activity, hours, date).</td>
<td>Not specified</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Product releases</td>
<td>Planned product releases and deployment dates. List of features, responsible, downtimes etc.</td>
<td>No</td>
<td>Static html page</td>
<td></td>
</tr>
<tr>
<td>Service windows</td>
<td>Dates for planned service windows, incl. responsible and activity</td>
<td>No</td>
<td>Static html page</td>
<td></td>
</tr>
<tr>
<td>Promotions</td>
<td>Ongoing and upcoming promotions in the casinos run by NE.</td>
<td>Not specified</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Jour</td>
<td>Jour schedule for NE.</td>
<td>No</td>
<td>Link to document</td>
<td></td>
</tr>
<tr>
<td>Vacations</td>
<td>Vacation schedule for NE employees.</td>
<td>No</td>
<td>Link to document</td>
<td></td>
</tr>
<tr>
<td>Backoffice schedule</td>
<td>Schedule for backoffice employees.</td>
<td>No</td>
<td>Link to document</td>
<td></td>
</tr>
</tbody>
</table>

2.6 Acceptance criteria

The system should be accepted when the acceptance test has been completed and the number of errors does not exceed the ones in the table below.
2.7 Project close

<table>
<thead>
<tr>
<th>Severity</th>
<th>No. of errors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Critical. The Intranet is down or not available.</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Serious. Main feature of the Intranet is not working.</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>Minor. Minor logical error in sub-feature of the Intranet.</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>Cosmetic. Cosmetic issues, spelling or similar.</td>
</tr>
</tbody>
</table>

2.7 Project close

The Steering Committee should close the project when:

- All deliverables have been delivered by the developer.
- The specified deliverables have been approved by the project owner.
- The acceptance criteria for the solution are met.
3 In-depth study of JavaServer Pages

In this section, the technology of JavaServer Pages (JSP) [1, 2, 3, 4, 5] is explained. JSP is a part of a larger technology called the Java2 Enterprise Edition (J2EE) [22]. JSP itself is a technology for controlling the content and appearance of web pages through the use of Servlets. A Servlet is a small server-side program that is called by the web server to generate the page before it is sent to the user who requested it [10]. The result of a request to a JSP is a HTML page. A good reference on JavaServer Pages is [1], from which also most of the theory presented in this section is taken from.

This section starts with describing the J2EE platform, in order to get the reader to understand how JSP fits into this technology (Section 3.1–3.2). After that (Section 3.3–3.4) the relationship between JSP and Servlets is described. In Sections 3.5–3.6, the use of JSP tags and Enterprise JavaBeans are discussed. In Sections 3.7–3.8, it is discussed how databases and security can be used together with JSP. The section end with a brief comparison between and discussion of different existing technologies for creating dynamic web pages.

3.1 The Java2 Enterprise Edition

In 1998, Java2 was released by Sun Microsystems, Inc [11]. There are three different versions [12], each one intended for a specific purpose. The Micro Edition (J2ME), intended for use with embedded devices, such as cellular phones, handheld computers etc. The second one is the Standard Edition (J2SE), which is intended for standard desktop application. The third and largest one is the Enterprise Edition (J2EE), which builds on the standard edition, but has a significant amount of other packages included for use in distributed enterprise applications. Essentially, J2EE is a collection of APIs that can be used to build enterprise applications [1].

One of the aims with J2EE is to provide interfaces in order to simplify for the end developer for tasks that are common in enterprise applications. Instead of writing large amounts of code, the developer uses the services described in the APIs which has been implemented by a vendor.

The APIs which are part of the J2EE are listed below and can be found in [1]. APIs from the standard edition are not included in the list.
### 3.2 J2EE Applications

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servlets/JSP</td>
<td>APIs for the presentation of a web application.</td>
</tr>
<tr>
<td>Enterprise JavaBeans (EJB)</td>
<td>APIs for handling business logic, data access etc. Most of the enterprise application functionality lies in the EJB container.</td>
</tr>
<tr>
<td>Java Messaging Service</td>
<td>API for a messaging framework.</td>
</tr>
<tr>
<td>Java Transaction API</td>
<td>Defines how transactions can be managed.</td>
</tr>
<tr>
<td>Java Activation Framework Connector</td>
<td>Provides support for MIME types. Enables connection of a J2EE enterprise server to an enterprise information system.</td>
</tr>
<tr>
<td>Java Mail</td>
<td>API enabling emails to be received and sent.</td>
</tr>
<tr>
<td>SOAP with attachments API</td>
<td>API for creating and processing SOAP messages. Provides declarative and programmatic security for Java applications.</td>
</tr>
<tr>
<td>Java Authentication and Authorization service (JAAS)</td>
<td>API for locating and naming resources.</td>
</tr>
<tr>
<td>Java Naming and Directory Interface (JNDI)</td>
<td>API for database access.</td>
</tr>
<tr>
<td>Java Database Connectivity (JDBC)</td>
<td>API for interaction with remote methods. Enables invocation of remote procedures with SOAP and WSDL. Provides XML registry access.</td>
</tr>
<tr>
<td>Java API for XML Registries</td>
<td>Provides XML registry access.</td>
</tr>
<tr>
<td>Java API for XML Processing</td>
<td>For the SAX, DOM, XSLT processing of XML.</td>
</tr>
</tbody>
</table>

These interfaces can be implemented by vendors who want to develop different types of products. Some vendors only implement some of the APIs while others decide to implement the whole application framework. If the whole application framework is implemented, this implementation will be referred to as a **J2EE application server** [1]. To verify application servers, Sun has developed a verifier called **J2EE Compatibility Test Suite** [23]. Some of the J2EE application servers available today are:

- Apache Tomcat,

- IBM Websphere,

- Sun Java System Application Server,

- Ironflare Orion,

### 3.2 J2EE Applications

Enterprise applications often consist of several layers. The three most common are the **Presentation layer**, **Business layer** and the **Data layer** [1] (see Figure 1). In a web application, the presentation layer often consists of e.g., JSP/Servlets (see Section 3.3–3.4), Java Beans (Section 3.6), and Tag libraries (Section 3.5).
The business layer contains the business logic of the application. This logic often uses the J2EE APIs implemented in the J2EE server (these APIs are listed in Section 3.1). The data layer consists of essential data for the application, which is usually stored in a database, XML file(s) or any other type of file(s) containing data which is used by the application. Both the presentation layer and the business layer reside in what are referred to as containers. Containers enable communication between layers, and provide services to the components that exist within them. J2EE application components always use the protocols and methods of the container when interacting with each other and other services in the platform. The presentation logic is in a web container, and the business layer is within an EJB container.

Figure 1: Application layers.

A typical J2EE product provides an application client container, applet container, web container and an EJB container. All J2EE-compliant containers must provide the following services:

- Java Messaging Service.
- Transactions.
- Java Naming and Directory Interface.
- Java Authentication and Authorization Service.

The EJB container (see Figure 2) must also support

- Entity Beans for Data access,
- Session Beans for Business logic,
- Message-driven Beans for Asynchronous messaging,

while the Web container (see Figure 3) must support

- Servlets,
- JSPs,
- JavaBeans,
- Tag Libraries.
Figure 2: The EJB container.

Figure 3: The Web container.
3.3 Servlets

Since the beginning of the Internet, web pages have changed from being a collection of static pages, to become highly dynamic and personalized. For web developers, the work today consists of building such dynamic pages. A collection of approaches has evolved throughout the years. Java applets [14] were introduced by Sun in the beginning of the 1990s and was an attempt to try to make web pages more dynamic. A Java applet runs on the client machine without any reliance of the server [1]. The only way to get server-side processing was to use Common Gateway Interface (CGI) programs [15], which were written (mostly) in the Perl programming language [16]. The main problem with CGI had to do with performance. Every request from the client started a separate process (with its own address space) in the server application. When the number of clients started to increase there would soon be a large number of processes running at the server, which clearly affected the performance of the service.

Later Netscape, Apache and Microsoft published APIs for their servers which enabled developers to write multithreaded server programs that were much more efficient (visualized in Figure 4). The drawback though, was that an application written for Apache (for instance), could not run on any other server. Since the thought of a server-independent application would be very attractive, it did not take long before a such a method was developed. This is what we today call a Servlet [1, 24]. A Servlet is an application which applies to the “Write Once, Run Everywhere” philosophy of the Java family, i.e., it uses the Java Virtual Machine technology (see Figure 5).

![Figure 4: Contrast between the CGI and the multithreaded model.](image)


3.3 Servlets

![Servlet Diagram]

Figure 5: The Servlet processing model.

3.3.1 Structure of the Servlet

To understand how JSP works, you first need to know how the Servlet is working. The HttpServlet class implements the Servlet interface in the `javax.servlet.*` package. The HttpServlet class contains the following methods:

```java
protected void init(ServletConfig config)
protected void doGet(HttpServletRequest req, HttpServletResponse res)
protected void doPost(HttpServletRequest req, HttpServletResponse res)
protected void doTrace(HttpServletRequest req, HttpServletResponse res)
protected void doHead(HttpServletRequest req, HttpServletResponse res)
protected void doPut(HttpServletRequest req, HttpServletResponse res)
protected void doDelete(HttpServletRequest req, HttpServletResponse res)
protected void getLastModified(HttpServletRequest req)
protected void destroy()
protected void service(HttpServletRequest req, HttpServletResponse res)
```

3.3.2 The Servlet lifecycle

When a request for a JSP comes to the application server, this will initiate an execution of the Servlet corresponding to the requested JSP. What really happens from the moment of the incoming request until the response is sent back is shown in Figure 6 and described below.

The course of events starts when a request for a JSP is received by the server from a client. If there is no instance of the Servlet already running on the server, a new instance will be created from the JSP and the constructor and `init()` methods are executed. After that, the `service()` method is executed, which redirects the request to the appropriate “do” method. `Http` supports several different commands. The most common ones are the `GET` and the `PUT` commands. So, if the request is a `GET`, then the `doGet()` method is invoked
in the Servlet. When this method has returned, the response (a HTML file) is sent back to the client.

3.3.3 Basic Servlet example

In Example 3.1, you can see a basic implementation of an extended HttpServlet. When a class is extended, new methods can be added which are not available in the parent class. If a method that already exists in the parent class is redefined, the new method will be used instead of the one in the parent class. The doGet() method in the example returns a simple html page to the client who is making the request.
Example 3.1

```java
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.IOException;

public class MyServlet extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        ServletOutputStream out = res.getOutputStream();
        res.setContentType("text/html");
        out.println("<HTML>");
        out.println("<HEAD><TITLE>MyServlet</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("Hello World.");
        out.println("</BODY>");
        out.println("</HTML>");
    }
}
```

3.4 JavaServer Pages

So, what is the relationship between JavaServer Pages (JSP) [2, 3, 4, 5] and Servlets one could wonder. A JSP looks very similar to a regular HTML document at first sight. The thing that differs is that it can also contain Java code (scriptlets). When the JSP is to be executed for the first time it is compiled into a Servlet (and after that into a class file) [1]. For example, a JSP page that would produce a Servlet like the one in Example 3.1 would look like:

```html
<HTML>
<HEAD><TITLE>MyServlet</TITLE></HEAD>
<BODY>
Hello World.
</BODY>
</HTML>
```

All Java code which is included in a JSP must be embedded in between the `<% %>` tags to be recognized as code. To prevent a large mix of HTML and Java code (which sometimes can be confusing), Tag Libraries can be used. Tag libraries are described in Section 3.5.
3.5 JSP Tags

A common design goal in web applications is to try to separate the display code from the business logic. JSP tags [25] make this possible, which results in that you won’t have to clutter the JSP page with Java code. Instead of writing code blocks of Java in the JSP, the tag libraries allow for you to make custom HTML-like tags which are mapped to a Java class which performs the business logic. This is done by creating a Tag class, in which you put the business logic. So, when you use a specific tag in the JSP, this tag invokes the corresponding code in the Java class. Additional to the benefit of separating the display code and business logic, this also simplifies for web authors who does not know Java. With JSP tags they can find a simple solution to write dynamic pages without learning the Java programming language. One of the intentions with JSP Tags is to help simplify JSP authors’ lives. Authors can easily use libraries that have been developed by others (and of course write their own if they want to), to get a greater functionality on their pages. The Java Standard Tag Library (JSTL) [26], which is commonly referred to as the standard tag library, contains four tag libraries for different purposes. The Core library [27] offers tags for simple Java control commands such as if, for, switch, and print. So instead of cluttering a JSP with unnecessary Java code, one could use the \(<c:if>, <c:forEach>, <c:choose>, and <c:out>\). The other three libraries are the XML [28], Internationalization-capable Formatting [29] and SQL [30] libraries.

However, the possibility to build custom tag libraries can be very useful. In the list below, you can see four major advantages [1].

- Custom tags make it easier to write and maintain JSPs. This is because the custom tags have an XML-like syntax that is easy for page authors who are familiar with HTML to learn. The resulting JSPs are cleaner since they are not cluttered with scripting code.

- One of the major benefits of custom tags is that they are very reusable. Thus, you can write a custom tag of your own that provides a specific functionality. After it has been tested and debugged, it can be used over and over again.

- The previous point leads to this one: Custom tags make the development of new Web applications quicker. After a page author has learned how a tag works, that knowledge can be reapplied to new JSPs. Also, since the custom tag has been used before, new errors are not introduced as would be the case with the development of a new custom tag.

- If you decide that you want to change the way that the custom tag works, that change is automatically reflected in all the JSPs that use the tag. You do not need to modify any of the pages that use it. It is also possible to extend a tag with new attributes, rather than completely change the way it works. In this way, existing pages continue to use the old form of
the tag until modified, but new pages can immediately use the new form with the extra attributes.

3.6 Enterprise JavaBeans

JavaBeans [17] where originally designed for making it easy to write new user interface components that could be integrated into existing development environments. However, they are a big asset when writing JSPs, since they help to encapsulate the business logic. The Enterprise JavaBeans (EJB) container in a Java application server holds three different types of JavaBeans [1, 31]: Entity Beans, Session Beans, and Message-Driven Beans (see Figure 7). In this section, these three types are described and their differences explained.

![EJB Container](image)

Figure 7: The EJB container.

3.6.1 Entity Beans

An entity bean could simply be described as a Java object, with a few special properties [8]:

- **It is permanent.** Regular Java objects come to life when a program creates them. Later, when the program terminates, the objects are lost. An entity bean on the other hand, continues to live until it is destroyed.
A program can create an entity bean and then terminate. When it is restarted again, it can continue using the same entity bean.

- **It is network based.** While standard Java objects are used by one program only, the entity bean can be used by any program/JSP in the network. The only thing a program needs to know is the location of the bean.

- **It is executed remotely.** When you call a bean’s method, the local thread stops running and the control is passed over to the bean. When the method returns from the bean, the local thread continues its execution.

### 3.6.2 Session Beans

Session beans differ from the entity beans in that they are not permanent objects. They only live throughout a user’s session. When the session ends, the bean is destroyed automatically. They are also not in general shared between programs.

### 3.6.3 Message-driven beans

A message-driven bean is an enterprise bean that listens for messages sent to the J2EE application. It allows for messages to be processed asynchronously. It can be compared with an event listener, with the difference that it receives messages instead of events. Messages can be sent to the application by any J2EE component within the application, or by any application supporting the Java Messaging System (JMS) protocol [18].

### 3.7 Databases and JSP

An important property which makes Java useful is that it has a standard way to let Java applications access databases. The Java Database Connectivity (JDBC) technology is the industrial standard for database-independent connectivity Java and most of the databases available on the market today [6, 7]. The JDBC API makes it possible to do three things [6]:

- Establish a connection with a database or access any tabular data source.
- Send SQL statements.
- Process the results.

JDBC also allows programmers to update multiple data items with a single command, or even access multiple database servers within a single transaction [7].
The technology also supports reuse of connections, so that every new command does not have to create a new connection. This is called connection pooling [1] (see Figure 8).

Figure 8: Connection pooling.
3.7.1 JDBC Drivers

To connect to a specific database, a JDBC driver is needed. Drivers are available in four different types [1, 6, 7, 32], see Figure 9. Type 1 and 2 are intended for programmers writing applications, while type 3 and 4 are typically used by vendors of middleware or databases.

Figure 9: The different types of JDBC drivers.

**Type 1: JDBC-ODBC bridge**
This driver type provides access via one or more Open Database Connectivity (ODBC) drivers [7]. ODBC is a mechanism that Windows applications can use to access databases. It provides a uniform interface to many of the database vendors for applications running on the Windows platform. ODBC must in this case be loaded on each client machine. The drawback of this method is that performance suffers because there are some overhead associated with the translation between JDBC and ODBC.

**Type 2: Partial Java driver**
This type of driver converts JDBC calls on the client API for, e.g., Oracle, or other Database Management Systems (DBMS) [7]. It can be explained as being a hybrid, containing Java code, but also containing native code from the database vendor [1]. Just as type 1, this type of driver requires that some binary code is loaded on every client machine.
Type 3: Pure database driver for database middleware
The Pure database driver for database middleware converts JDBC calls into the middleware vendor’s protocol, which is then translated to a DBMS protocol by a middleware server [6]. So, the client is actually communicating with the middleware, and the middleware with the DBMS. The middleware server often supports several different databases.

Type 4: Direct-to-database pure Java driver
The Direct-to-database pure Java driver uses the network protocol used directly by the DBMSs, allowing a direct call from the client machine to the DBMS server [6]. This is clearly the type of driver with the highest performance. Pure java drivers are available for nearly all the major database implementations available today. These drivers are written only in Java. The drawback of this style is that the client machine needs one driver for every single DBMS it needs to access [7].

3.8 Security and JSP
There are different types of security issues depending on whether you are a user on a specific site, or if you are involved in developing/running the site. Here follows a list of the main issues that a user/developer must consider [1].

- When sensitive information is passed over the Internet, how do I know that no one will intercept it? (Confidentiality)
- How do I know that the web-site really is the web-site it say it is? (Authentication of the server)
- How do I know that my data has not been modified on the way? (Data integrity)
- How do I restrict access to resources? (Access control)
- How can I be sure that users are who they say they are? (Authentication of users)
- How is the security credentials (usernames, passwords etc.) passed through to different parts of an application?

In J2EE, security can be implemented using two different approaches [1]: Declarative security and Programmatic security. Declarative security is when the deployer of an application declare security information for different types of resources such as EJB, Web Folders, or any other type of resource. In this case, the developer of the application does not need to implement any security. The opposite method is Programmatic security, which means that all security is hard-coded into the application. This makes the developer responsible for all security functions.
3.8.1 Authentication

The validation of users is called client authentication. There are many ways to authenticate users, but in a web application there are four ways that can be used declaratively [1]: HTTP Basic, Digest, Form-based, and HTTPS Client authentications. The security-constraints are placed in the application’s deployment-file (web.xml) in the web container of the application server.

**HTTP Basic Authentication** is the most simple authentication method. The username and password are sent from the client in clear-text over the HTTP/1.0 specification. This makes it very insecure, and is not recommended.

**Digest Authentication** is more secure, since in this case the client never send its username and password in clear-text. Instead, a hash is created with information of the username, password, the requested URL, a nonce (a number sent from the server), and the HTTP method being used. When the server gets the hash it create its own hash using the same information. If the two hashes are the same the user is validated.

In the two methods above, the user gets a popup-window when trying to access a page in the secured area. In **Form-based Authentication**, the user instead uses a standard html form to fill in the username and password. The username must be sent as a `j_username` parameter, and the password as `j_password`. This method is nothing to recommend when not using the https protocol, because the username and password are sent in clear-text.

The last method one could use is **HTTPS Client Authentication**. This method uses the benefit of Secure Socket Layer (SSL) [19], which use both symmetric and asymmetric key encryption, as well as digital certificates and digital signatures. The essential thing to know is that SSL used in a proper way provides a very reliable connection between the client and server, which supports both confidentiality and data integrity.

The drawback of these four declarative methods are that they only can be used for either allowing or denying access to a protected area. If we instead want to customize the appearance of a specific page depending on who made a request for it, we need to use programmatic security. This means that the application developer would simply check, within the JSP, what type of account the logged on user has and customize the information based on that.

3.8.2 Confidentiality and Data integrity

As described in Section 3.8.1, secure authentication of the client can be achieved by using Digest, Form-based (with SSL enabled), Programmatic (with SSL en-
abled), or HTTPS Client Authentication. What about the other issues listed in Section 3.8? To solve the remaining issues, we should take a more detailed look at SSL. As pointed out in Section 3.8.1, SSL supports confidentiality and data integrity. To trust this method, one must know a few things about it. SSL fully supports confidentiality, i.e., no one is able to intercept the data sent between the server and the client, since it is heavily encrypted. But there is one thing to have in mind: are we sure that the server we communicate with really is the one we intend to communicate with? To make sure, one must get hold of the server’s public key in a secure way. Preferably one would contact the Certification Authority (CA)\(^2\) which is responsible for keeping this server’s public key. To get hold of the right public key is critical to ensure a secure connection. When connecting to a server with SSL, a certificate is used. In this certificate, the public key is declared. In order to trust the communication, one must check whether this key is identical to the key that has been received securely. If this is the case, one can be sure that the server actually is the correct server, and that no one will be able to intercept the data. If not, this server is not the one that you intended to communicate with, although it may look like it is.

To ensure data integrity SSL uses a message digest mechanism [21]. This can be explained as a fingerprint of the data sent. It is impossible to prevent the data to be altered along the way between the server and the client, but if the data has been altered on the way, SSL will detect this and dismiss this data. SSL calculates the fingerprint before it is sent to the network and appends it to the encrypted data that should be sent. When the data arrives at the receiving part, SSL recalculates the fingerprint of the data, and then compares it to the one which were appended. If both of the fingerprints match, the receiver can be sure that the data has not been altered on the way. If they does not match, the content has been altered somewhere along the way, and the package is dismissed. It is practically impossibly for someone to alter the data and the fingerprint so that the new fingerprint is valid for the altered data.

3.8.3 Security credentials

The developer of an application must consider how sensitive the credentials passed within the system are, in order not to expose these unintendedly. A simple solution to this is to keep sensitive information in a hashed format [20]. For example, passwords do not need to be kept in their original form. Instead it can be stored (in a database, for example) in a hashed format. The benefit will be that no one is able to get hold of the users passwords, even those who have full access to the database. When the user logs on to the system, the password is hashed by the application, and then compared with the hash stored in the database. If they are identical, the user is authenticated.

\(^2\) A CA is a trusted place where public keys are kept.
3.9 Comparison of technologies for creating dynamic web content

Some of the most popular technologies for developing dynamic web content today are JavaServer Pages (JSP), Common Gateway Interface (CGI), mod perl, the Hypertext Preprocessor (PHP), and Microsoft Active Server Pages (ASP). These different technologies are compared throughout this section. Important issues such as portability, performance, and other characteristics for the different technologies are discussed.

3.9.1 Common Gateway Interface

Common Gateway Interface (CGI) [15] is supported in many of the web servers available today. It was the first technology ever to create dynamic server-side applications. CGI applications are not portable between different platforms in general. In order to develop portable content, one must be very careful when designing CGI applications. If all operating system specific libraries are avoided, the application can be moved to another platform. As described in Section 3.3, CGI has performance issues when many users use the application at the same time, since it does not support multithreading. For every new session, a separate process is created (and a new address-space is allocated for each one). When the number of users start to increase, the allocated memory increases very fast. From the very beginning, only the Perl programming language was supported, but nowadays many of the major languages are supported, among these are C/C++, Perl, Fortran, and Visual Basic. An issue with using the Perl programming language is that it is not strongly typed, so type errors will not show up until run-time. Neither is it an optimal language to use when it comes to team development, since it is not object-oriented. It can be hard to distribute the implementation.

3.9.2 JavaServer Pages

One of the major benefits of using JavaServer Pages (JSP) [2, 3, 4, 5] has to do with portability. Since, JSP take advantage of the Java Virtual Machine (JVM) [33], which runs on all major operating systems available, it is easy to move an application between different application servers (both commercial and free ones), as well as between different operating systems. That the JVM benefits from systems using several CPUs is another nice feature. Many features can be easily added into a web application using the different parts of the Java2 Enterprise Edition, such as Java Messaging service, Java Mail etc. In contrast to the other technologies listed in this section, JSP is not only designed to be used by programmers. When using JSP Tags, it becomes very easy to use for developers that only have the experience of HTML codes. Since Java is object-oriented, it becomes very suitable for team development of JSP applications. The Java programming language is the only language supported for JSP.
3.9.3 mod perl

*mod perl* is a project to integrate the *Perl* programming language into the *Apache Web Server*\(^3\). This means that it can only run on one specific web server. Also, an application written for an Apache instance on one operating system must be rewritten in order to work on an Apache instance set up on another operating system. *mod perl* is supposed to be a lot faster than the traditional CGI-scripts. In contrast to Java, the Perl language does not support strong typing, so type errors can be hard to find. Thus, the code must be carefully written. Since Perl has now become an object-oriented language, also mod perl applications are suitable for team development. Obviously, the programming language supported by *mod perl* is Perl.

3.9.4 Active Server Pages

*Active Server Pages* (ASP) [35], is a quite new technology, developed by Microsoft. It has good performance, and can be written in *JavaScript*, *VBScript* and several other scripting languages. Just as JSP, ASP supports multithreading, which makes it suitable for web applications that are to be very busy receiving requests. The limitation for ASP is that it is only available for the *Microsoft Internet Information server* (IIS) [36], which can only run in a *Microsoft Windows*\(^4\) environment. This makes this the most expensive technology to use of the ones listed in this section.

3.9.5 Hypertext Preprocessor

The *Hypertext Preprocessor* (PHP) [37] is a script language that has become very popular. It can be used with many of the most common web servers and databases available today. Applications written in PHP are portable between different web servers and platforms. PHP uses the *Perl* programming language. Many developers today use PHP together with the *Apache Web Server* and the *MySQL*\(^5\) database, which together provides a powerful, complete, and free-of-charge platform to use for building and running web applications on.

3.9.6 Conclusions

Deciding what technology to use for creating dynamic web content is not that easy. One must consider all the factors described above. Naturally, for many developers, the choice often is related to which programming languages that are familiar. Obviously, the size of the application that is going to be developed should also play a big role, since portability and the ease of team development are often critical in large projects. For large applications one would preferably

---

\(^3\)Information about the Apache Web Server can be found at http://httpd.apache.org/.
\(^4\)Microsoft Windows is a registered trademark of Microsoft Corporation.
\(^5\)Information about MySQL can be found at http://www.mysql.com
use JSP or ASP because they are very suitable for team development and there exist excellent development tools for them. Another important factor is the cost for developing and running an application using the various technologies. The performance of the technology is also one thing to be considered. For “heavy” applications a developer might not want to use CGI, because of the performance issues it has.
4 Accomplishment

This section simply group the different activities of the project into phases in order for the reader to see what actually happened throughout the project. The timeline in Figure 10 shows how time was distributed for the different phases along the project.

4.1 Schedule of project

The project was initially divided into the four phases of the project described below. The actual accomplishment of these phases can be seen in Section 4.2.

Phase 1: Inception

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project definition</td>
<td>Define scope and schedule of project.</td>
</tr>
<tr>
<td>In-depth study on J2EE</td>
<td>Read-up and documentation of J2EE/JSP.</td>
</tr>
<tr>
<td>Existing implementation</td>
<td>Evaluate what has already been developed for the Intranet and how this applies to the requirements.</td>
</tr>
<tr>
<td>Detailed specification of the requirements</td>
<td>Interviews with reference group and requirement owners in order to make a detailed specification of the requirements of Release 1 and outline the requirements of Release 2. Results in the Functional requirements specification table.</td>
</tr>
<tr>
<td>Implement staff list</td>
<td>Implement a first Intranet feature in order to apply theories as well as existing setups.</td>
</tr>
</tbody>
</table>

Phase 2: Elaboration

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sketches</td>
<td>Draw sketches illustrating the user interface of the Intranet for all required pages.</td>
</tr>
<tr>
<td>Systems design</td>
<td>Visualize the Systems architecture (server setup etc.), database model and class diagrams.</td>
</tr>
<tr>
<td>Define content responsibilities</td>
<td>Define which employee is responsible for delivering the content for each page.</td>
</tr>
<tr>
<td>Design of security model</td>
<td>Develop and implement support for authentication and access control.</td>
</tr>
</tbody>
</table>

Phase 3: Construction

Since it is easier to develop one feature at a time, this phase is defined as an iterative process for each feature in the Intranet.
4.2 Changes and actual distribution of time

Since an internal supervisor was not assigned (by the department) to the project in phase 1, the “in-depth study” was not carried out during phase 1 (simply because the content of the study could not be decided without one). Instead it was carried out within the scope of phase 3. This was a little bit of a setback, because the knowledge of JSP increased quite a bit during this study.

Another problem was the interviews in phase 3. Many of the employees was not sure of what functionality they really wanted in the part they were responsible for. In many cases, ideas had to be be implemented and delivered by the developer, and then be evaluated by the responsible employee. It would have saved a lot of time if the responsible employees knew (in advance) what functionality they wanted. However, there was enough time to implement the second release (and even further in most of the pages).

There were some features that were not implemented at all of those specified in Table 1. The budget page was not implemented, since no official budget has been released by the company. But since this feature was only defined as a link to a document, it will not be any challenge to add for the developers at Net Entertainment AB when the budget is available. The assignments feature was merged into the projects feature, since the definition of the functionality for this feature is very similar to the functionality in the projects feature. The last feature that was not implemented is the time-reporting tool, since Net Entertainment AB has decided to buy a commercial software product for this purpose (with a much wider functionality than the one defined for this project). The product releases page and the projects page are still under development, and
will probably be ready before the end of June 2005.

Regarding the open issue described in Section 2.4, it was decided that the Intranet should be accessible over the Internet, simply because the customer support company (located in Costa Rica) should also be able to access the Intranet. This should not be any problem, since the Intranet uses SSL, and is password protected.

The time distribution of the project is visualized in Figure 10.

![Figure 10: Time distribution of the project.](image)
5 Implementation

The results of the project are presented in this section. Section 5.1 describes the software tools that have been used during the project. The implemented Java classes are described in Section 5.2, while the design of the database is explained in Section 5.3. In Section 5.4, the administrative groups of the Intranet are presented. The security model of the Intranet application is defined in Section 5.5, and finally in Section 5.6 an example of what happens when a client is requesting a JSP in the Intranet is presented.

5.1 Software tools used during the project

Throughout the project, the Eclipse editor (version 3.1.0) has been used for developing/writing the source code for the Intranet. During the development phase of the project, an Apache Tomcat application server (J2EE application servers are discussed in Section 3.1) was run locally on the developer’s laptop in order to execute the Intranet application. A Microsoft SQL Server\(^6\) database was also set up at the local machine and used together with the Tomcat instance. Thus, making it possible to develop and test the Intranet without an Internet connection (very useful on the train between Stockholm and Uppsala). At the time of deployment, an Ironflare Orion application server was set up to run the Intranet at a dedicated server within the network at Net Entertainment. A multipurpose Microsoft SQL Server database located at another server at the same network was set up to be used by the Intranet.

5.2 Java classes

Two classes are used by the Intranet on different occasions. The Intranet class contains security methods as well as some good-to-have methods. The most frequently used methods are the checkSSL, which makes sure that SSL encryption always is enabled, and the memberOf method, which JSPs use to check whether the logged on user is a member of any administrative group in order to customize the output of the page. The SqlUtil class is used for creating database connections and to execute and get the result of SQL queries. Notice that transactions are supported, which makes it safe to execute series of queries before doing a commit. All methods for these two classes are defined and shortly described below.

\(^6\)Microsoft SQL Server is a registered trademark of Microsoft Corporation.
5.2 Java classes

The Intranet class

- **private static void checkSSL(HttpServletRequest, HttpServletResponse)** – This method is included in all JSP in the Intranet. It simply redirects a http request to a https request.

- **public static String getDay(int)** – Converts an integer representing the day of the week to a two-character string. 1 returns “Su”, 2 returns “Mo” and so on.

- **public static String toMonth(int m)** – Converts an integer representing the month of the year to a string with the name of the month. 0 returns “january”, 1 returns “february” and so on.

- **public static String toShortMonth(int m)** – Same as above, but in short format. 0 returns “Jan”, 1 returns “Feb” and so on.

- **public static String removeChars(String)** – Replaces all occurrences of ’ ‘ with ‘ ’ in order to secure SQL commands.

- **public static boolean memberOf(String, String)** – Checks if a user belongs to a specified group.

- **public static String avoidNull(String)** – Converts null strings to an empty string.

The SqlUtil class

- **public SqlUtil()** – Constructor. Sets the right SQL driver and which datasource to use.

- **public SqlUtil(boolean)** – Constructor. Sets the right SQL driver and which datasource to use. If the boolean parameter is set to false, auto-commit is switched off (for SQL transaction use).

- **public void commit()** – For committing a transaction.

- **private Object[][] getMatrix(String)** – Loads the resultset of a SQL command into an Object matrix.

- **private Object[][] getMatrix(String, int)** – Same as getMatrix, but the integer limits the number of lines loaded into the matrix.

- **public void executeQuery(String)** – Executes a SQL query. Calls getMatrix.
- public void executeQuery(String, int) – Executes a SQL query with a maximum number of rows. Calls getMatrix.

- public void executeUpdate(String, int) – Executes a SQL update.

- public String getString(String) – Returns the value of a SQL cell (in String format) at the current line, where the columns name equals the String parameter.

- public boolean next() – Moves the cursor to the next line in the Object matrix. Returns false if there are no more lines.

- public void beforefirst() – Moves the cursor to its initial position in the Object matrix.

- public boolean first() – Moves the cursor to the first line in the Object matrix. Returns false if there are no first lines.

- public void close() – Closes the connection with the database.

5.3 Design of database

The heart of the Intranet is within the database. Here lies all information that is presented in the JSPs of the Intranet. Throughout Sections 5.3.1–5.3.20, the role of all tables in the database is explained. The SQL server used for this project is Microsoft SQL Server. Also the Microsoft Enterprise Manager has been used during the development in order run SQL queries, as well as to create and edit tables and to set their indexes and foreign key relationships. In Figure 11, all tables of the database are shown. Foreign key relationships are marked out with the key line connectors, which run between the tables.

5.3.1 The employees table

The employees table is a central part of the database (as one can see in the diagram in Figure 11). It contains fields for username and password which are used when logging on to the Intranet. It also contains first name, last name and other optional information about the employee such as address, URL to a picture of the employee and instant messaging accounts. The valid field is used by the Intranet administrator when he/she wants to disable an Intranet user (rather than removing all occurrences of this user in the database). Since the username field is a foreign key to many of the other tables of the database, it can be an advantage to be able to disable an account instead of deleting all information about this user. For example, if an employee is being re-employed, all information can easily be restored.
5.3 Design of database

Figure 11: Database diagram.

5.3.2 The companies table

The *companies table* holds the data of the companies which are included in the Cherry Group. Each company has a unique identifier, *companyname*, and some general information such as address, phone numbers etc.
5.3.3  The worksatcompany table

The *worksatcompany table* specify which company an employee works for. An employee can be involved at several companies in the Cherry Group, therefore the same *username* can exist several times in the table. For each company a person works at, they have a position, an email and a phone number.

5.3.4  The groups table

Many of the pages of the Intranet should be available for some type of administration, such as adding, editing or removing information. Some pages of the Intranet are to be administrated only by a specific group of users, while other pages can be administrated by several groups. In this table the available groups are specified. See Section 5.4 for more details about the different groups a user can belong to.

5.3.5  The ingroup table

In the *ingroup table*, existing users and groups are connected to each other. A user can be a member of none, one or several groups.

5.3.6  The eventtypes table

The *eventtypes table* specify which types of basic events that exist in the Intranet. Currently three types of events can occur in the Intranet, these are the: emph[jour], *vacation* and *system events*.

5.3.7  The events table

The *events table* contains information about all the events added into the Intranet. *Username* refers to which employee the event applies to, *type* must be a *eventtype* from the eventtypes table, and *starttime* and *endtime* specify during which period of time the event takes place. *Description* holds additional information about the event, *color* is used to distinguish different types of system events, and *systemid* is used as a unique identifier for system events.

5.3.8  The messages table

The *messages table* stores what is shown on the news page. It holds the messages posted by the users in the Intranet, as well as information about who posted a specific message and at what time the message was posted.
5.3.9 The backofficestaff table

The “Backoffice” company (located in Costa Rica) is responsible for customer support in the casinos that Net Entertainment runs. Their employees need to have some more detailed information about themselves in order to get a usable schedule on the backoffice schedule page. In the backofficestaff table, every employee in the backoffice is assigned a color. This color is used as background color behind their names in the backoffice schedule page, in order to make it easier to see when an employee is working.

5.3.10 The supportedlanguages table

The supportedlanguages table contains all the languages that are available at all casinos that Net Entertainment runs. These are mapped to the backoffice employees in the knowslanguage table.

5.3.11 The knowslanguage table

Every employee in the backoffice can handle one or several languages. In the knowslanguage table, every employee is mapped to the language(s) he or she masters. When viewing the backoffice schedule, the schedule can be filtered on a specified language using this table.

5.3.12 The shifts table

The employees in the backoffice works in shifts. In the shifts table, the different types of shifts are stored. It contains an id of the shift (e.g. “Afternoon”) and what time the shift starts and ends.

5.3.13 The boschedule table

The boschedule table holds the backoffice schedule. It contains a username (the employee who is working), the workingday (which day the employee works) and a shiftid (foreign key from id in the shifts table) corresponding to the shift the employee works that day. It also contains a comment which is visible in the schedule.

5.3.14 The casinos table

In the casinos table, all the casinos managed by Net Entertainment AB are listed. All casinos are assigned a color to be used in the promotion page.
5.3.15 The promotions table

The *promotions table* is used for storing all the promotions in the casinos. A promotion is a time-limited “happening” in a casino. Every promotion has a unique *id* (an integer), a *header* (short description) and a *description* field. Every promotion also has a responsible employee, a *startdate* and an *enddate.*

5.3.16 The inpromotion table

The *inpromotion table* stores which promotion from the *promotions table* is assigned to which casinos. Since one promotion can run on several casinos at the same time, this table is used for connecting the casinos to a specific promotion.

5.3.17 The projects and inproject tables

By the time this report is written, the outline of the projects page has not been fully decided. The main features though (as can be seen in the database diagram in Figure 11), are to be able to add projects and to add employees into these projects. This feature will be implemented when the outline has been decided by Net Entertainment AB.

5.3.18 The customers table

The company’s customers are listed in the *customers table.* Each customer has a unique *name.* The other columns are used for address information about the customer company.

5.3.19 The contacts table

Each customer company can have one or several contact persons. These are listed in the *contacts table.* Each contact person must be connected to a company *name* from the customers table. For every contact person the table holds information about the name of the contact, *title* (such as technician, CEO etc.) as well as their *phone number* and *email.*

5.3.20 The companyaccounts table

The *companyaccounts table* holds information about the company’s accounts at external companies such as FedEx, UPS etc.
5.4 The groups in the Intranet

Below, the different groups available today are listed and explained. The groups are specified in the groups table described in Section 5.3.4. Groups can easily be added, and members added and removed, by the members of the Intranet administrators group.

- **Intranet Administrators** – A member of this group has full permission in the Intranet. They can, e.g., add users into the system, edit which group(s) they belong to, which company/companies they work at, and set a new password if the user has forgotten it. These are the most privileged users. They have full permission.

- **Administrators** – A member of this group has the permission to add users into the system and to set which company/companies they work at. This group is responsible for administrating the vacation schedule and the company accounts page.

- **Backoffice managers** – This group is responsible for adding/removing staff in the backoffice company, and to administrate their work schedule at the backoffice schedule page.

- **Customer responsible** – This group is responsible for keeping the customer contacts page up to date.

- **Project leaders** – Responsible for managing the project pages.

- **Jour managers** – Responsible for managing the jour schedule.

- **System administrators** – Responsible for managing the system events page.

- **Enable login** – Users that are included in this group are allowed to login on the Intranet.

5.5 Security model

When a user tries to log on to the Intranet, the login page checks if the username and password are valid, that the account is not invalidated, and that the user is a member of the enable login group. If everything is ok, the username is assigned to the session variable called **user** and the user is redirected to the news page. JSP that have administrative features always check if the current user has administrative permissions using the **Intranet.memberof** method, which returns a boolean assinged **true** if the specific user is included in the specified group. The output of the page is then customized accordingly.

Every request to a page using the http protocol is always redirected to the same
page, but using the https protocol for security reasons. The certificate used was set up at Net Entertainment AB and is a sub-certificate to the company’s own root certificate.

5.6 A typical request example

To get at picture of what happens in the system when a request comes in, a brief example is presented. The example contains the most common actions taking place in the Intranet when a JSP is requested. Let us say a client makes a request to see the \texttt{a.jsp} page. This is what happens (see also Figure 12):

1. A request for \texttt{a.jsp} comes in.

2. The JSP invokes the \texttt{Intranet} class in order to:
   
   (a) Determine if the request is sent over SSL. If not, the request will be redirected to the same page using SSL.

   (b) Determine if the user has been signed in. If not, the client will be directed to the login page.

   (c) Determine if the user has administrative rights for this page, i.e., the user is a member of a specified group. The \texttt{Intranet} class uses a \texttt{SqlUtil} object to check this in the database. If this user is not allowed to view this page, the user will be denied to see the information that should have been presented.

3. The JSP continues to execute if (at least) both 2a and 2b are true.

4. The JSP invokes the \texttt{SqlUtil} class in order to fetch the data from the database that will be presented on the requested page.

5. The \texttt{SqlUtil} object connects to the database in order to fetch data from the database.

6. The results are sent back to the \texttt{SqlUtil} object.

7. The results are sent back to the JSP.

8. The response is presented to the client (in html).
6. Future work

After the presentation at the Department of Computing Science in June 2005, the work on the project will continue until the end of June by the author. Additional pages might be added during that time. After that, developers at Net Entertainment AB will manage the Intranet and continue with future development when necessary.

7 Acknowledgements

I would like to thank my external supervisor at Net Entertainment, Anders Callertun for supporting me in my work. I would also like to thank Patrik
Nilsson, Henrik Liljekvist, Dan Meiling, David Ekholm, and Johnny Aspelin at Net Entertainment. At the Department of Computing Science I would like to thank my supervisor Stefan Johansson.
References


[30] JSTL SQL. Sun Microsystems, Inc.,

[31] Enterprise JavaBeans Technology. Sun Microsystems, Inc.,


[33] Java Virtual Machine - All about JVMs,

[34] mod perl: Welcome to the mod perl world,

[35] ASP.NET Web: The Official Microsoft ASP.NET Site,

[36] Internet Information Services 6.0 Product Information. Microsoft Corporation,

[37] PHP: Hypertext Preprocessor,
A Net Entertainment Intranet user’s guide

This guide describes the implemented features of the Intranet. It is written in order to get the reader to understand how things appears in the browser, and as a user’s guide for the end user. A picture for the most important pages are available.

A.1 The login prompt

To access the Intranet, users must log in from the page shown in Figure 13, which is reached from https://intranet.netentertainment.com. All pages in the Intranet checks if the user has logged on. If not, they are redirected to this page. If the login is valid, the user is redirected to the page that was requested, or to the news page if no specific JSP was requested.

![Login prompt](image)

Figure 13: A user at the login prompt.

A.2 The menu

The top menu shown in Figure 14, which appears when the user has logged on, contains links to the different pages available. It is built up using html tables and java scripts. It was originally developed by Anders Callertun (the external supervisor of this project), but has been modified by the author for use in the Intranet. The contents of the menu and the pages in the Intranet are described in Sections A.3–A.17.
A.3 The jump menu

The jump menu makes it easy for employees to jump between the Intranet and the company’s Internet casinos. It supports automatic logins at the web administration part of each casino. If the user has the same username and password in the Intranet as at the casinos, the user will not be prompted for this information when jumping to one of the casinos. If the information does not match, access will be denied and the user will be redirected to the login page of the casino. It works the same way in the opposite direction, i.e., from the casinos it is possible to directly jump to the Intranet.

A.4 News page

Under the menu news->add news all users are able to post messages which are published at the news page (Figure 15). Only the last 100 messages are shown. Every posted message contains information about who posted it and at what time.
A.5  Customer contacts page

On the customer contacts page (Figure 16), all contact persons for the company’s customers are displayed. If the current user is a member of the customer responsible group, a link will appear that enable the user to edit the information displayed on this page. It is important to say that the security does not only consist of hiding the URL to the edit pages in the Intranet, the membership of the user is also controlled at the edit page. If the user does not have the right permission, access will be denied.

A.6  Company accounts page

A list of the company’s accounts at other companies is displayed on the company accounts page (Figure 17). It can be viewed by all users which have access to the Intranet. The administrators group has the permission to edit the displayed information.

A.7  Other pages in the documents menu

The remaining links at the documents menu are only links to Microsoft Excel and Word documents, and will not be described since they do not have any special functionality.
Figure 16: A user viewing the contact persons at external companies.

Figure 17: A user viewing Net Entertainment’s accounts at other companies.
A.8 Staff listing page

On the staff listing page shown in Figure 18, all employees in the Cherry Group are listed (sorted by the name of the company they work at). At the top of the page there are links to the anchor of each company, in order to quickly view the employees at a specific company. Basic information for each employee is visualized. To see more information about an employee (such as home address, picture, instant messaging accounts etc.), the link at the employee’s name (or the more info) can be followed. In that case, a page for the specific employee will be presented, showing all known information about the employee.

![Staff listing page](image)

Figure 18: The staff listing page shows all employees in the Cherry group.

A.9 Edit staff page

The edit staff page can only be accessed if the user is a member of one or several of the intranet administrators, administrators or backoffice managers groups. A member of the intranet administrators group will see all users listed in this page, and is able to edit information about all of them (see Figure 19). They are also the only group which has the permission to set the group memberships of all users. The administrators can edit information about all users which are not working at the backoffice. These users are instead handled by the backoffice
managers. The users shown in the list can also be disabled and completely deleted from the Intranet. If the user is disabled, he/she can not log in on the Intranet and is not visible as an employee of any company. However, he/she will not be deleted from any schedules, and all news that the user has posted will remain at the news page. Nothing is removed from the database. If the user is deleted, on the other hand, all occurrences of the user in the tables of the database is removed.

![Intranet administrator can edit the users.](image)

**Figure 19:** An Intranet administrator can edit the users.

### A.10 Intranet groups page

This page was added to make it easy for users to see which users are included in the different Intranet groups (see Figure 20). All Intranet users can access this page.

### A.11 Vacations page

The vacations for the employees are listed on the vacations page (see Figure 21). It has a monthly view, and the user can go backward and forward in time by using the previous month and next month links. The table grows dynamically
Figure 20: A user viewing the Intranet groups.

Figure 21: A user viewing the vacations page.
depending on how many employees have vacation during the current month. For users who have more than one period of vacation during the current month, the periods are merged into the same row. If the current user is a member of the *administrators* group, he/she is permitted to add vacations. In that case a link will appear at the bottom of the page for adding a vacation. When the mouse pointer is held over one of the table cells of a vacation, information about that vacation appear in a yellow box right next to the mouse pointer. When left clicking a colored cell, the same information is presented in its own html page. On this page, members of the *administrators* are also able to remove the vacation event.

When an authorized user wants to add a vacation, a page appear for setting the start date, end date and other required information (see Figure 22). A similar page also appear when adding any other schedule activities described in the following sections. The calendar[9] used in these pages is a JavaScript distributed under the GNU[7] LGPL[8].

![Image](image_url)

**Figure 22:** An administrator adding a vacation.

---

A.12 Promotions page

On the promotions page all “happenings” in the casinos are listed (see Figure 23). Every casino can have several promotions in parallel, and one promotion can run on several casinos at the same time. As in the vacations page, when the mouse pointer is held over one of the table cells of a promotion, information about that promotion appear in a yellow box right next to the mouse pointer. A link for adding a new promotion is visible for promoters at the bottom of the page. If one would click on one of the colored cells, the information will appear as a new html document, as well as a link for removing that promotion from the database if the user is a member of the promoters group.

A.13 Jour page

The jour schedule for the company is visualized on the jour page (Figure 24). It is almost identical to the vacations page, both in design and in functionality. The only thing that differs is the color of the cells and that the jourmanagers group is responsible for the contents of this page.

![Figure 23: A user viewing the promotions page.](image-url)
A.14 Backoffice schedule page

The backoffice schedule page contains the work schedule for the casino’s customer support office (see Figure 25). It shows the schedule for the current week and 14 weeks forward in time. A link at the top of the page makes it possible to go back in time to see the past 15 weeks. Every week shows the shift periods for each day of the week. One cell (a specific shift on a specific day) can contain any number of employees, but an employee cannot appear twice in the same cell. Every employee has their own color, in order to distinguish them easier. The schedule can also be filtered on a specific employee, in order to get one employee’s schedule. Another feature is that it can be filtered on when support is available in a specific language. So, one could easily see when for example Turkish support is available.

The backoffice managers can left click in an empty area of a cell in order to add an employee which is to be working that day and shift. A new page appears with the specific day and shift filled in. All he/she has to do is to choose from a drop-down list which employee is meant to work during that time and write a comment (optional) that also appear in the schedule. Also here (as in the other schedules) an event can be removed by clicking it and then choosing remove on the information page that appears (backoffice managers only). There are also pages for managing the duration of the shifts and for adding new backoffice employees, as well as for editing the schedule color and which languages they know.

Figure 24: A user viewing the jour schedule.
A.15 System events page

Information about planned system events are published on the system events page (see Figure 26). A system event can be: a casino being upgraded and restarted, replacing any of the servers etc. Different types of system events are grouped by color encoding. Additional information pops up when holding the mouse pointer over a colored cell. The system administrators are responsible for setting the contents of this page.

A.16 Register account page

On the register accounts page new Intranet users can be registered into the system (see Figure 27). Only two groups can access this page, the Intranet administrators and the administrators. Information that must be typed in are the first and last name, username and password (which can be changed by the user itself later). Information about which company/companies in the Cherry group the user is employed at should also preferably be entered. Additional information such as personal information (home address, private phone numbers etc.) is optional. Instead it can be typed in by the user itself under Change account details if he/she wants others to be able to see such information.
Figure 26: A user viewing the system events.

A.17 Change account details page

On the change account details page all users can edit basic information for their own account and set a new password (Figure 28).
Figure 27: An administrator or Intranet administrator registering a new account.
Figure 28: A regular user views its account details.