ijsfontein

System maintenance and developement

Cyberdam 2.3

Plaats	Amsterdam
Datum	18 februari 2008
Auteur	Simon Groenewolt
Functie	Programmer

Document	Cyberdam 2.3 System Maintenance.doc
Versie	1.0

Index

1 Introduction

This document describes Cyberdam 2.3 from a maintenance and development point of view. It is intended for java developers with some knowledge about web applications, the Spring framework and Hibernate. The core components and the database will be discussed.

2 System Components

Cyberdam 2.3 has been created using a number of libraries. We'll discuss the most important parts of the system.

2.1 Java

The system has been programmed and tested on java 5. It uses annotations for some of the functionalities – especially the Java Persistence API Annotations used for Object Relational Mapping with Hibernate. Enums and generics are also used.

2.2 Tomcat

The default war file has been made to work on Tomcat version 6. Tomcat has been chosen because it is one of the most well known open-source servlet containers.

2.3 The Spring Framework

The Spring framework is at it's core a dependency injection framework, but combined with the Spring MVC framework and the Spring Web Flow system it is basically a lightweight alternative to a full J2EE system. See http://www.springframework.org/

2.4 Spring MVC

The standard Model-View-Controller system to use with the base Spring system. It provides conventions for setting up controllers and views, and provides base classes for most important tasks. Almost all web pages in Cyberdam are handled by Spring MVC.

2.5 Spring Web Flow

Spring Web Flow is a system to enable multi-step web-based edit sessions – it is being used in the game model creation part of the site. The current implementation saves the game on every navigation action within the flow – this should be noted since it is uncommon for flow setups.

2.6 Hibernate

Hibernate 3 has been chosen as the object-relational mapper for the project, it integrates nicely with the Spring framework. Classes are configured for Hibernate using annotations, but they still have to be specified in the properties of the sessionFactory bean in the applicationContext.xml

2.7 MySQL

MySQL has been chosen because it is well known and free.

3 System description

3.1 Java package layout

This is the layout of the java classes.

package	description
nl.cyberdam.domain	All domain objects - these objects are

	mapped to the database using Hibernate.
nl.cyberdam.multilanguage	database backed multilanguage implementation
nl.cyberdam.service	the service and data access layer.
nl.cyberdam.service.dwr	contains a class that can be used by DWR (http://getahead.org/dwr) to enable/disable some user settings. Only used on the session pages.
nl.cyberdam.tools	user csv file importer and password helper file (to generate an encrypted password for direct insertion in the database)
nl.cyberdam.util	locale resolver, login/logout listeners, utility classes
nl.cyberdam.validation	validators that are not web-specific
nl.cyberdam.web	all controllers for handling web requests
nl.cyberdam.web.customeditors	custom editors for several datatypes, used by controllers in edit forms
nl.cyberdam.web.flow	controller like objects used for the game model create/edit flow
nl.cyberdam.web.util	currently only the playground xml converter class.
nl.cyberdam.web.validation	web-layer specific validators

3.2 Spring configuration

Spring is configured using a number of xml files – these are all located in the WEB-INF directory.

file	configures
applicationContext.xml	Cyberdam core services and hibernate setup
applicationContext-acegi-security.xml	security specific settings
applicationContext-datasource.xml	(jndi) lookup code for database and email connections
applicationContext-email.xml	email sending service and threadpool
applicationContext-jmx.xml	jmx (java management) settings to expose hibernate data
dispatcher-servlet.xml	all controllers
webflow-config.xml	webflow specific configuration

3.3 Database Layout

Use `cyberdam_model.mwb` and Mysql Workbench to explore the database structure. Alternatively have a look at db_simplified.pdf – that provides a simplified view on the database schema (users, language system and logging were left out to cut back on the

interconnections). These files can be found in the `dbscripts` dir.

3.4 Views

All views are located in WEB-INF/jsp with the convention being that a request to “test.htm” will be handled by a controller named “TestController.java” using the view “test.jsp”.

4 Extending the system

The 'project' directory contains an eclipse project – get eclipse at: http://www.eclipse.org/
Extensions of the system most probably will happen in some part of the system that is spring related: a good reference can be found here:
http://static.springframework.org/spring/docs/2.5.1/reference/mvc.html
We will discuss a number of common scenarios.

4.1 A new domain object

New domain objects can be modeled like the ones already present in the nl.cyberdam.domain package. All ORM is done using JPA (http://java.sun.com/javaee/reference/) annotations with a couple of Hibernate specific additons, for example the list annotation used in the GameModel class. Note that a class needs to be added to the Hibernate configuration in applicationContext.xml if it should be handled by Hibernate.

4.2 A new page

Adding a new page consists of creating a new Controller (http://static.springframework.org/spring/docs/2.5.1/reference/mvc.html#mvc-controller) similar to the ones in nl.cyberdam.web and adding a view in the WEB-INF/jsp directory. The controller should be added to the WEB-INF/dispatcher-servlet.xml spring configuration file. Most controllers are mapped to web request by their classname, and no additional configuration is needed if you follow that pattern. If you want to map the controller to another request you can use the urlMapping bean available in the dispatcher-servlet.xml.

4.3 New system logic

Adding service objects should be done in the nl.cyberdam.service package. Service objects are generally instantiated by Spring and therefore should be configured in one of the xml files. They can then be used by controllers by implicitly or explicitly wiring them.

4.4 A note about multilanguage

Cyberdam 2.3 contains a multilanguage system. In webpages multilanguage strings can be retrieved using a spring tag <spring:message code=”multilanguage.code” />, in code it can be retrieved using the bean 'messageSource' available in the spring context – this bean implements the nl.cyberdam.multilanguage.MultiLanguageSource interface. Defaults for some multilanguage keys have been defined in WEB-INF/classes/messages.properties.

5 System Tuning

The system has a couple of points where it can be tuned.

5.1 Java and Tomcat

Using the Tomcat startup parameters the amount of memory and garbage collection settings for the Java Virtual Machine can be configured. See: http://tomcat.apache.org/tomcat-6.0-doc/index.html for Tomcat specific configuration options.

5.2 EhCache

Hibernate has been configured to be able to use the EhCache system for both the Hibernate second level cache and the query cache, configuration of the caches can be done using WEB-INF/classes/ehcache.xml

5.3 MySQL

MySQL is running as it's own process – please refer to http://dev.mysql.com/doc/#refman for tutorials. This doesn't fall within the scope of this document.

5.4 Logging

Logging is configured by editing WEB-INF/classes/log4j.properties Excessive logging may make the system slow.