Оглавление

[Постепенное понимание технологий GWT, ExtGWT, Spring, Freemarker…. 1](#_Toc321396826)

[1.создаем проект с использованием GWT и Spring framework-ов 2](#_Toc321396827)

[2.создаем в папке web/WEB\_INF папку для библиотек lib и копируем туда все нужные библиотеки 2](#_Toc321396828)

[3.при подключении библиотеки gxt.jar не забыть про файлы css/ gxt-all.css и flash/ swfobject.js, которые необходимо положить в папку web. В файлах \*.html не забыть про строку 2](#_Toc321396829)

[4. Точку входа в GWT настраиваем в файле jsp/index.jsp 2](#_Toc321396830)

[5. Ускорение процесса компиляции, задаем параметр сборки только под конкретный браузер(IE,Mazila) следующим образом: 2](#_Toc321396831)

[6.подключаем библиотеку логирования log4j 2](#_Toc321396832)

[7. Интеграция GWT и Spring (пример - смотри архив build\_4\_add\_GWT&Spring) 3](#_Toc321396833)

[8. Добавление аннотаций (пример - смотри архив build\_5\_add\_Annotation@) 6](#_Toc321396834)

[9. Добавление ModuleController (build\_7\_add\_ModuleController) 9](#_Toc321396835)

[10. Добавляем слой БД. (build\_7\_add\_DB\_Part1) 10](#_Toc321396836)

[10. Добавляем библиотеку frieemarker для шаблонов Html страниц (build\_8\_add\_freemarker) 12](#_Toc321396837)

[11.Добавляем в приложение Интерфейс для констант - com.google.gwt.i18n.client.Constants (build\_9\_add\_constants) 13](#_Toc321396838)

[12.Добавляем ресурсы с иконками (build\_10\_add\_icons) 14](#_Toc321396839)

[13. Добавляем слой БД (продолжение) (build\_10\_add\_UserServerSeervices) 14](#_Toc321396840)

[14 Добавила GWTUserService (из proj\_kav)( build\_12\_add\_GWTUserService) 15](#_Toc321396841)

[15 Добавила зависимость доступа к БД от сборки и другие файлы \*.properties (build\_13\_add\_properties) 16](#_Toc321396842)

[16. Разбираюсь с Spring Security(build\_14\_add\_security) 17](#_Toc321396843)

[17.Вопрос про использование аннотации @Autowired (build\_15\_add\_@Autowired) 18](#_Toc321396844)

[18.Добавляю главную страницу из proj\_kav (MainPage) (build\_16\_add\_mainWindow) 18](#_Toc321396845)

# Постепенное понимание технологий GWT, ExtGWT, Spring, Freemarker….

### 1.создаем проект с использованием GWT и Spring framework-ов

### 2.создаем в папке web/WEB\_INF папку для библиотек lib и копируем туда все нужные библиотеки

все библиотеки из проекта удаляем и добавляем те которые нужны из этой папке.

### 3.при подключении библиотеки gxt.jar не забыть про файлы css/ gxt-all.css и flash/ swfobject.js, которые необходимо положить в папку web. В файлах \*.html не забыть про строку

<link rel="stylesheet" type="text/css" href="resources/css/gxt-all.css" />

А в файле Interactive.gwt.xml

<inherits name="com.extjs.gxt.ui.GXT" />

### 4. Точку входа в GWT настраиваем в файле jsp/index.jsp

добавить строку

<script type="text/javascript" src="pst.arm.Interactive/pst.arm.Interactive.nocache.js"></script>

### 5. Ускорение процесса компиляции, задаем параметр сборки только под конкретный браузер(IE,Mazila) следующим образом:

(http://alextretyakov.blogspot.com/2011/11/gwt.html)

В файле Interactive.gwt.xml

<!-- hlv compile for Firefox only TODO delete it for work project-->

<set-property name="user.agent" value="gecko1\_8"/>

В проекте в свойствах в GWT (последный пункт ) на этой закладке опция argument

Задаем параметры компиляции -[draftCompile](http://mojo.codehaus.org/gwt-maven-plugin/compile-mojo.html#draftCompile) (отключает оптимизацию кода при компиляции приложения)

### 6.подключаем библиотеку логирования log4j

Создаем папку src/resources в свойствах проекта подключаем эту папку. В ней создаем файл с настройками log4j.properties.

В файле web.xml приложения подключим описаный выше обработчик событий:

<context-param>

<param-name>log4jConfigLocation</param-name>

<param-value>classpath:log4j.properties</param-value>

</context-param>

<listener>

<listener-class>

org.springframework.web.util.Log4jConfigListener

</listener-class>

</listener>

Теперь есть возможность использовать логирование с помощью log4j в своих сервлетах и jsp-страницах.

Ниже приводится пример сервлета использующего логирование с помощью log4j:

import org.apache.log4j.Logger;

public class TestLogServlet extends HttpServlet {

private static Logger logger=Logger.getLogger("simple");

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws IOException{

logger.info("log message from TestLogServlet");//( logger.warn(…),logger.debug(…), logger.error(…)

response.setContentType("text/html; charset=windows-1251");

PrintWriter out=response.getWriter();

out.print("<html><body>");

out.print("Log4j test servlet");

out.print("</body></html>");

}

}

JSP-страница, использующая log4j, могла бы выглядеть следующим образом:

<%@ page import="org.apache.log4j.Logger" %>

<% Logger logger=Logger.getLogger("simple"); %>

<html>

<body>

<%

logger.info("log message from TestLog JSP-page");

%>

Log4j test JSP-page

</body>

</html>

Если создать лог для **StatisticLogger:**

private static Logger log = Logger.getLogger("StatisticLogger");

То все еще будет писаться в файл, на который есть ссылка в файле с настройками compile-akozhin.properties для поля **log4j.appender.Stat.File** например я задала d:/web\_java\_proj/unicuspst/logs/arm.statistic.log

### 7. Интеграция GWT и Spring (пример - смотри архив build\_4\_add\_GWT&Spring)

Для вызова GWT RPC запросов

Создаем RPC сервис

**Файл cliеnt/service/ GWTService.java**

package pst.arm.client.service;

import com.google.gwt.user.client.rpc.RemoteService;

import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;

@RemoteServiceRelativePath("service/gwtservice")

public interface GWTService extends RemoteService {

public String myMethod(String s);

}

**Файл cliеnt/service/ GWTServiceAsync.java**

package pst.arm.client.service;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface GWTServiceAsync {

public void myMethod(String s, AsyncCallback<String> callback);

}

Файл server/service/GWTServiceImpl.java

package pst.arm.server.service;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import org.apache.log4j.Logger;

import org.springframework.stereotype.Service;

import pst.arm.client.service.GWTService;

public class GWTServiceImpl extends RemoteServiceServlet implements GWTService {

private static Logger logger=Logger.getLogger("GWTServiceImpl");

public String myMethod(String s) {

logger.warn("GWTServiceImpl::myMethod");

// Do something interesting with 's' here on the server.

return "Server says: " + s;

}

}

Файл server/service/ GwtRpcController.java (свзывает Spring и GWT)

package pst.arm.server;

import javax.servlet.ServletContext;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.web.context.ServletContextAware;

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.mvc.Controller;

import com.google.gwt.user.client.rpc.IncompatibleRemoteServiceException;

import com.google.gwt.user.client.rpc.RemoteService;

import com.google.gwt.user.client.rpc.SerializationException;

import com.google.gwt.user.server.rpc.RPC;

import com.google.gwt.user.server.rpc.RPCRequest;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

import org.apache.log4j.Logger;

public class GwtRpcController extends RemoteServiceServlet implements

Controller, ServletContextAware {

private static Logger logger=Logger.getLogger("GWTServiceImpl");

private ServletContext servletContext;

private RemoteService remoteService;

private Class remoteServiceClass;

public ModelAndView handleRequest(HttpServletRequest request,

HttpServletResponse response) throws Exception {

super.doPost(request, response);

return null;

}

@Override

public String processCall(String payload) throws SerializationException {

logger.warn("GwtRpcController::processCall");

try {

RPCRequest rpcRequest = RPC.decodeRequest(payload,

this.remoteServiceClass);

// delegate work to the spring injected service

return RPC.invokeAndEncodeResponse(this.remoteService, rpcRequest

.getMethod(), rpcRequest.getParameters());

} catch (IncompatibleRemoteServiceException ex) {

getServletContext()

.log(

"An IncompatibleRemoteServiceException was thrown while processing this call.",

ex);

return RPC.encodeResponseForFailure(null, ex);

}

}

@Override

public ServletContext getServletContext() {

return servletContext;

}

public void setServletContext(ServletContext servletContext) {

this.servletContext = servletContext;

}

public void setRemoteService(RemoteService remoteService) {

this.remoteService = remoteService;

this.remoteServiceClass = this.remoteService.getClass();

}

}

В web.xml

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>\*.htm</url-pattern>

**<url-pattern>/pst.arm.Interactive/service/\*</url-pattern>**

</servlet-mapping>

А в dispatcher\_servlet.xml

<bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">

<property name="alwaysUseFullPath" value="true" />

<property name="mappings">

<props>

<prop key="index.htm">indexController</prop>

**<prop key="/pst.arm.Interactive/service/gwtservice">sampleController</prop>**

</props>

</property>

</bean>

**<bean id="sampleController" class="pst.arm.server.GwtRpcController">**

**<property name="remoteService">**

**<bean class="pst.arm.server.GWTServiceImpl" />**

**</property>**

**</bean>**

Вызов метода myMethod() из клиента

GWTServiceAsync service = GWT.create(GWTService.class);

service.myMethod(txtUserInput.getText(), callback);

### 8. Добавление аннотаций (пример - смотри архив build\_5\_add\_Annotation@)

Конфигурирования компонентов mvc с помощью аннотаций. Благодаря этому значительно уменьшается количество конфигурационной информации в XML-файлах.

**1.Для использования этой возможности, в dispatcher-servlet.xml необходимо включить**

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:aop="http://www.springframework.org/schema/aop"

**xmlns:context="http://www.springframework.org/schema/context"**

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-3.0.xsd

http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx-3.0.xsd

**http://www.springframework.org/schema/context**

**http://www.springframework.org/schema/context/spring-context-2.5.xsd**">

**<context:component-scan base-package="pst.arm.server" />**

**<!-- Define DefaultAnnotationHandler specifically so we don't get railroaded by the other HandlerMappings -->**

**<bean class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping"/>**

**1.Далее создать файл котроллера**

package pst.arm.server.common.web;

import org.springframework.web.servlet.ModelAndView;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import java.io.IOException;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import javax.servlet.http.HttpServletRequest;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

//@RequestMapping("/index.htm")

public class MainController

{

protected final Log logger = LogFactory.getLog(getClass());

@RequestMapping("/index.htm")

public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

logger.warn("Returning pageName: ");

return new ModelAndView("index");

}

}

**3. Предыдущий п.7 тоже немного перепишем**

Создадим файл arm/server/common/web/ GWTController.jar

package pst.arm.server.common.web;

import javax.servlet.ServletContext;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.log4j.Logger;

import org.springframework.web.context.ServletContextAware;

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.mvc.Controller;

import com.google.gwt.user.client.rpc.RemoteService;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

//import ru.spb.iac.archives.arm.server.common.service.UserService;

/\*\*

\* Simple spring controller that merges GWT's {@link RemoteServiceServlet},

\* the {@link Controller} and also implements the {@link RemoteService}

\* interface so as to be able to directly delegate RPC calls to extending

\* classes.

\*

\* @author Amigos Team

\*

\*/

public class GWTController extends RemoteServiceServlet implements

ServletContextAware, Controller, RemoteService/\*, GWTSerializer\*/ {

private static final Logger log = Logger.getLogger(GWTController.class);

private static final Logger statLog = Logger.getLogger("StatisticLogger");

private static final long serialVersionUID = 5399966488983189122L;

//protected UserService userService = null;

@Override

public ModelAndView handleRequest(HttpServletRequest request,

HttpServletResponse response) throws Exception {

try{

//if( userService != null ){

// statLog.info( "SERVICE " + request.getRequestedSessionId() + " " + userService.getCurrentUser().getUserName() + " " + getClass().getSimpleName() + " "+request.getRemoteAddr());

//}

doPost(request, response);

return null;

}catch (Exception ex) {

log.error("handleRequest error "+ex);

//ex.printStackTrace();

return null;

}

}

private ServletContext servletContext;

@Override

public void setServletContext(ServletContext servletContext) {

this.servletContext = servletContext;

}

@Override

public ServletContext getServletContext() {

return servletContext;

}

}

**И тогда класс в GWTServiceImpl.java можно унаследовать от GWTController добавив аннотацию @Service**

package pst.arm.server.service;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import org.apache.log4j.Logger;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import pst.arm.client.service.GWTService;

import pst.arm.server.common.web.GWTController;

@Service("GWTService")

public class GWTServiceImpl extends GWTController implements GWTService {

private static Logger logger=Logger.getLogger("GWTServiceImpl");

@Override

public String myMethod(String s) {

logger.warn("GWTServiceImpl::myMethod");

// Do something interesting with 's' here on the server.

return "Server says: " + s;

}

}

**И файл dispatcher-servlet.xml сократить так**

<bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">

<property name="alwaysUseFullPath" value="true" />

<property name="mappings">

<props>

<prop key="/pst.arm.Interactive/service/gwtservice">**GWTService**</prop>

</props>

</property>

</bean>

### 9. Добавление ModuleController (build\_7\_add\_ModuleController)

**Создаем файл pst.arm.server.common.web /ModuleController.java**

package pst.arm.server.common.web;

import java.io.IOException;

import java.net.URL;

import java.util.Enumeration;

import java.util.HashMap;

import java.util.Map;

import java.util.Properties;

import javax.servlet.http.HttpServletRequest;

import org.apache.log4j.Logger;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.MessageSource;

import org.springframework.ui.Model;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

abstract public class ModuleController {

return "index";//MODULE\_VIEW;

}

}

**Тогда файл pst/arm/server/common/web /MainController .java можно переписать следующим образом**

package pst.arm.server.common.web;

import org.springframework.web.servlet.ModelAndView;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import java.io.IOException;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import javax.servlet.http.HttpServletRequest;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

**@RequestMapping("index.htm")**

public class MainController extends ModuleController

{

}

### 10. Добавляем слой БД. (build\_7\_add\_DB\_Part1)

Добавляем библиотеки **c3p0-0.9.1.2.jar ,** **ojdbc14.jar, ojdbc14\_g.jar**

Добавляем файл **applicationContext-jdbc.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx-3.0.xsd"

xmlns:tx="http://www.springframework.org/schema/tx">

<bean id="dataSourceArm" class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">

<property name="driverClass" value="${arm.env.jdbc.driverClass}" />

<property name="jdbcUrl" value="${arm.env.jdbc.url}" />

<property name="user" value="${arm.env.jdbc.username}" />

<property name="password" value="${arm.env.jdbc.password}" />

<property name="testConnectionOnCheckout" value="true"/>

<qualifier value="dsArm"/>

</bean>

</beans>

**В web.xml добавляем ссылку на этот файл**

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

/WEB-INF/applicationContext.xml,

**/WEB-INF/applicationContext-jdbc.xml**

</param-value>

</context-param>

**В resources добавляем файл с настройками доступа к БД env.properties**

#arm jdbc configuration

arm.db.charset=utf-8

arm.db.dialect=org.hibernate.dialect.Oracle10gDialect

arm.env.jdbc.driverClass=oracle.jdbc.driver.OracleDriver

arm.env.jdbc.url=jdbc:oracle:thin:@192.255.0.6:1521:XE

arm.env.jdbc.username=arm\_passat

arm.env.jdbc.password=passat

**Переносим ссылки на файлы со свойствами(\*.properties) из web.xml в applicationContext.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-3.0.xsd

http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-2.5.xsd">

<context:component-scan base-package="pst.arm.server" />

<context:annotation-config/>

<bean id="propertyConfigurer"

class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="locations">

<list>

<value>classpath:env.properties</value>

<value>classpath:log4j.properties</value>

</list>

</property>

</bean>

<bean id="messageSource" class="org.springframework.context.support.ReloadableResourceBundleMessageSource">

<property name="defaultEncoding" value="UTF-8"/>

<property name="basename" value="classpath:messages"/>

</bean>

</beans>

### 10. Добавляем библиотеку frieemarker для шаблонов Html страниц (build\_8\_add\_freemarker)

**а) Добавить библиотеку freemarker.jar (**FreeMarker — компилирующий обработчик шаблонов)

**б) Создать папку WEB\_INF/freemarker в которую скопировать файл index.jsp** переименовав его в index.ftl и удалив из него перую строку о jsp. А папку WEB\_INF/jsp можно удалить.

**В) В dispatcher-servlet.xml** изменить бин bean id="viewResolver" следующим образом

<bean id="viewResolver" class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver" abstract="false"

scope="singleton" lazy-init="default" autowire="default" >

<property name="requestContextAttribute" value="rc"/>

<property name="cache" value="true"/>

<property name="prefix" value="/WEB-INF/freemarker/"/>

<property name="suffix" value=".ftl"/>

<property name="exposeSpringMacroHelpers" value="true" />

<property name="contentType">

<value>text/html;charset=UTF-8</value>

</property>

</bean>

**Примечание:** В коде проекта proj\_kav в этом биде присутствует еще свойстово " dependency-check :

<bean id="viewResolver" class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver" abstract="false"

scope="singleton" lazy-init="default" autowire="default" **dependency-check="default">**

Я пока не разобралось что оно означало, но в springe 3.0 (а в проекте proj\_kav springe 2.5) этого свойства похоже уже нет. Я его пока убрала.

г) В applicationContext.xml добавить следующий код

<bean id="freemarkerConfig" class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer" abstract="false"

scope="singleton" lazy-init="default" autowire="default">

<property name="templateLoaderPath">

<value>/</value>

</property>

<property name="freemarkerSettings">

<props>

<prop key="datetime\_format">dd/MM/yyyy</prop>

<prop key="number\_format">0.#####</prop>

<prop key="url\_escaping\_charset">UTF-8</prop>

<prop key="default\_encoding">UTF-8</prop>

</props>

</property>

</bean>

### 11.Добавляем в приложение Интерфейс для констант - com.google.gwt.i18n.client.Constants (build\_9\_add\_constants)

Создаем папку pst/arm/client/common/lang в которую кладем файлы \*. .properties и для них кассы \*. Java

В Interactive.gwt.xml добавляем строки

<inherits name="com.google.gwt.i18n.I18N" />

<!-- Available locales -->

<extend-property name="locale" values="ru\_RU"/>

<!--Locale to compile-->

<set-property name="locale" value="ru\_RU"/>

(Можно в файле \*.flt добавить строку

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<link rel="stylesheet" type="text/css" href="/css/gxt-all.css" />

<title>Welcome to Spring Web MVC project</title>

**<meta name="gwt:property" content="locale=ru\_RU">**

</head>

В клиенте теперь можно писать так

BaseConstants constants = (BaseConstants) GWT.create(BaseConstants.class);

lblReadConstant.setText(constants.aboutDialogTitle());

Еще переименовала все свои классы с добавлением Simle

Добавила файл pst/arm/client/common/AppHelper.java который можно использовать с.о

lblReadConstant.setText(AppHelper.getInstance().baseUrl());

Добавляем еще файлы с константами pst/arm/client/config

Configuration.properties, Configuration.java

## 12.Добавляем ресурсы с иконками (build\_10\_add\_icons)

Сами иконки кладем в resources/pst/arm/client/common/images

Файл для работы с иконками src/java/pst/arm/client/common/images/CommonImages.java

Использование:

CommonImages commonImages = GWT.create(CommonImages.class);

buttonReadConstant.setIcon(AbstractImagePrototype.create(commonImages.add()));

Никаких дополнительных настроек не требутся

(имя функции в CommonImages.class должно соответсвовать названию файла с картинкой)

### 13. Добавляем слой БД (продолжение) (build\_10\_add\_UserServerSeervices)

Создаем серверную часть всего для пользователя User (интерфейс клиента пока не трогаем)

1)Добавляем всю pst/arm/client/common/domain/ (структура данных)

2) Добавляем классы для обработок ошибок pst/arm/client/common/exception

SiteException.java, RpcServiceException.java, InfrastructureException.java, BusinessException.java

3) Добавляем фал pst/arm/client/common/ ConfigurationManager.java – он основан на GWT классе com.google.gwt.i18n.client.Dictionary

Справка:

---------------Начало справки----------------

В переменной Dictionary храняться активные javascript объекты

//они создаются в commonGWT.ftl и в ModuleController.java

java.lang.Object

extended by com.google.gwt.i18n.client.Dictionary

Provides dynamic string lookup of key/value string pairs defined in a module's host HTML page. Each unique instance of Dictionary is bound to a named JavaScript object that resides in the global namespace of the host page's window object. The bound JavaScript object is used directly as an associative array.

For example, suppose you define the following JavaScript object in your host page:

var CurrentTheme = {

highlightColor: "#FFFFFF",

shadowColor: "#808080",

errorColor: "#FF0000",

errorIconSrc: "stopsign.gif"

};

You can then use a Dictionary to access the key/value pairs above:

public void useThemeDictionary() {

Dictionary theme = Dictionary.getDictionary("CurrentTheme");

String highlightColor = theme.get("highlightColor");

----------------Кнец справки----------------------

4) Добавляем фал pst/arm/server/common/ ConfigurationManager.java

5)Добавила файл pst/server/common/domain/ ServerSideUser.java

Он использует библиотеку spring-security.jar, добавляем ее.

6)Добавляем pst/client/modules/admin/domain/ UserSearchCondition.java

7)Добавляем dao прослойку работы с БД

pst/server/common/dao….

Для нее делаем настройку в <bean id="transactionManagerArm" в applicationContext-jdbc.xml

<bean id="transactionManagerArm" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSourceArm"/>

<qualifier value="txArm"/>

</bean>

<bean id="userDAO" class="pst.arm.server.common.dao.jdbc.UserDAOJdbcImpl">

<property name="dataSource" ref="dataSourceArm" />

<property name="transactionManager" ref="transactionManagerArm"/>

</bean>

8)Добавляем сервисы сервеной стороны(обращение к dao объектам, работа с данными)

pst/server/common/service/UserService.java

pst/server/common/service/impl/UserServiceIml.java

Для них делаем настройку в applicationContext.xml

<bean id="userService" class="pst.arm.server.common.service.impl.UserServiceImpl">

<property name="userDAO" ref="userDAO" />

</bean>

9)Написала для понимания, происходящего в сервисной части сервеной стороны, сервис TestService и вызов его из GWTServiceSimpleImpl в методе myMethodGetAllUsers()

### 14 Добавила GWTUserService (из proj\_kav)( build\_12\_add\_GWTUserService)

Скопировала

pst/arm/client/common/service/remote/GWTUserService, pst/arm/client/common/service/remote/GWTUserServiceAsync

pst.arm.server.common.web/ GWTController

pst.arm.server.common.service.gwt/ GWTUserServiceImpl

В dispatcher-servlet.xml добавила

<prop key="/pst.arm.Interactive/service/userService">GWTUserService</prop>

В applicationContext.xml добавила

<bean id="GWTUserService" class="pst.arm.server.common.service.gwt.GWTUserServiceImpl">

<property name="userService" ref="userService" />

</bean>

### 15 Добавила зависимость доступа к БД от сборки и другие файлы \*.properties (build\_13\_add\_properties)

В env.properties изменила

arm.db.charset=utf-8

arm.db.dialect=org.hibernate.dialect.Oracle10gDialect

arm.env.jdbc.driverClass=oracle.jdbc.driver.OracleDriver

arm.env.jdbc.url=jdbc:oracle:thin:@192.255.0.6:1521:XE

arm.env.jdbc.username=arm\_passat

arm.env.jdbc.password=passat

**на**

arm.db.charset=@environment.arm.db.charset@

arm.db.dialect=@environment.arm.db.dialect@

arm.env.jdbc.driverClass=@environment.arm.env.jdbc.driverClass@

arm.env.jdbc.url=@environment.arm.env.jdbc.url@

arm.env.jdbc.username=@environment.arm.env.jdbc.username@

arm.env.jdbc.password=@environment.arm.env.jdbc.password@

добавла в nbproject файл compile-akozhin.properties, где задается **@environment**

а ссылака на compile-akozhin.properties задается в файле nbproject /project.properties:

#build-compile properties

compile.property.file.location.default=nbproject/compile-akozhin.properties

Итого добавила(изменила) следующие файлы , пока не понимаю как все это работает

build.xml

nbproject /project.properties

nbproject /compile-akozhin.properties

nbproject /application-default.properties

nbproject /application-locale-server\_en.properties

nbproject /application-locale-server\_ru\_RU.properties

nbproject / build-compile.xml

nbproject /build-deploy.xml

nbproject /build-integrate-db.xml

nbproject /build-test.xml

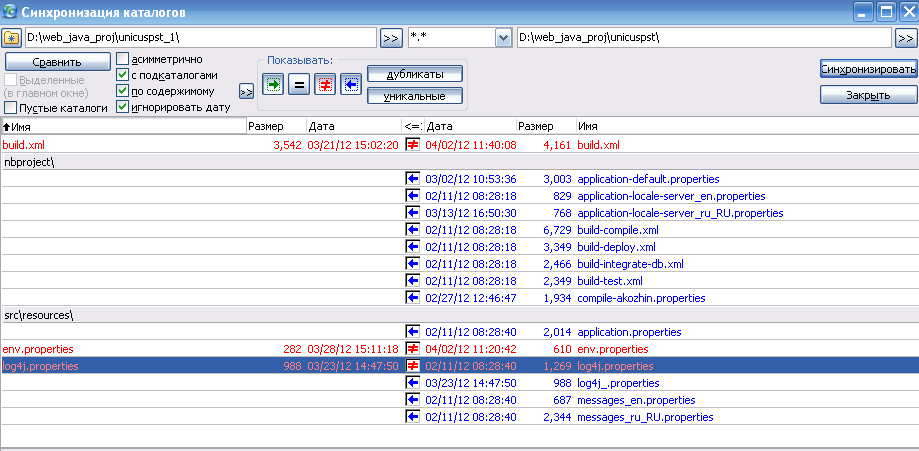
resources/env. properties

resources/log4j. properties

resources/ messages\_en.properties

resources/messages\_ru\_RU.properties

resources/ application.properties



### 16. Разбираюсь с Spring Security(build\_14\_add\_security)

Включить бибилотеки (если не были включены ранее) spring-security.jar, commons-codec-1.4.jar

Добавить файл applicationContext-acegi-security.xml с настройками spring security

Добавить ссылку на этот файл в web.xml

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

/WEB-INF/applicationContext.xml,

/WEB-INF/applicationContext-jdbc.xml

**/WEB-INF/applicationContext-acegi-security.xml**

</param-value>

</context-param>

Добавила файл с экранной формой логина

freemarker/login.ftl

freemarker/common.ftl

freemarker/spring.ftl

перенесла свою страницу Index.ftl в папку freemarker/secure (не обязательно, пытаюсь приблизиться к структуре каталогов проекта proj\_kav) и соответственно перенастроила контроллеры :

в public class MainControllerSimple .java @RequestMapping("/**secure**/index.htm")

в redirect.jst response.sendRedirect("**secure/**index.htm")

в index.ftl <script type="text/javascript" src="**../**pst.arm.Interactive/pst.arm.Interactive.nocache.js"></script>

Скопировала (из proj\_kav) папку с картинками web/ images

Скопировала css/ archives.css

### 17.Вопрос про использование аннотации @Autowired (build\_15\_add\_@Autowired)

(См. <http://www.seostella.com/ru/article/2012/02/12/ispolzovanie-annotacii-autowired-v-spring-3.html>)

Столкнулсь с непониманием того, почему иногда без этой аннотации все работает. Описываем объект в applicationContext.xml , заполненняем его свойства например testService и при вызове метода использующего этот сервирс все отрабатывается. Дело в том, что аннотации или используют и тогда используют для всех концепций или не используют и тогда аннотации вовсе не нужны. Т.е если мы перед сервисом написали аннотацию @Service("GWTServiceSimple") тогда и перед методами надо писать нужные аннотации например @Autowired для автозаполнения в applicationContext.xml, а если перед классом нет @Service то и @Autowired не нужна.

В примере добавила назначение сервиса setTestService для GWTServiceSimpleImpl в applicationContext.xml с использованием аннотаций.ВАЖНО: Для того что бы было возможно в applicationContext.xml создать конкретный объект класса с заполнением его полей эти поля обязательно должны иметь в классе методы **set,**

Пример:

в applicationContext.xml :

<bean id="*testService*" class="pst.arm.server.common.service.impl.TestServiceImpl">

</bean>

<bean id="GWTServiceSimple" class="pst.arm.server.service.gwt.GWTServiceSimpleImpl">

<property name="**testService**" ref="*testService*" />

</bean>

В классе должен быть метод **setTestService**

@Service("GWTServiceSimple")

public class GWTServiceSimpleImpl extends GWTControllerSimple implements GWTServiceSimple {

private TestService **testService**;

…..

@Autowired

public void **setTestService**(TestService testService) {

logger.warn("GWTServiceSimpleImpl::setTestService");

this.testService = testService;

}

### 18.Добавляю главную страницу из proj\_kav (MainPage) (build\_16\_add\_mainWindow)

Добавила ресурсы со свойствами resources/configs/base.properties, которые используются в ModuleController.java - контроллер для отображения модуля на экране. В этом файле основные настройки для компонент модулей.

Properties baseProps = new Properties(); (import java.util.Properties;)

baseProps.load(this.getClass().getResourceAsStream("/configs/base.properties"));

Добавила много связанных классов для обработки Модулей

### 19. Отображение новостей на главной странице. Работа с файлами . Что обслуживает класс CommonServiceImpl (build\_17\_add\_newsInMainWindow)

Новости (Последнии изменения) читаются из файла имя которого задается в параметре application.config.application.resources.changeLogFile файла с настройками compile-\*.prooerties

Будем их хранить в файле statlog/ news.txt

Функцией CommonServiceImpl::public String getChangeLog()

В этой функции используется библиотека commons-io-1.4.jar для чтения содержимого файла

Например для чтения файла можно написать следующий код

import java.io.File;

import org.apache.commons.io.FileUtils;

…

File f = new File("D:/1.txt");

String changeLog = FileUtils.readFileToString(f,"UTF-8");

Еще на главной странице можно отобразить ссылки на дополнительные ресурсы. Они тоже читаются из файла имя которого задается в параметре application.config.application.resources.linksFile файла с настройками compile-\*.prooerties

, будем их хранить в файле statlog/ links.txt

функцией CommonServiceImpl::getPortletLinks

Для вывода окна с новостями и окна со ссылками необходимо что – бы в файле Base.properties

config.modules.main.portlets=links changelog если написать

config.modules.main.portlets=links то будет отображаться только окно со ссылками, если

config.modules.main.portlets=changelog то только новсоти

В классе CommonServiceImpl еще есть функции которые отвечают за обработку статистики. т.е. Сама статистика - это логи сервера приложений, генерируемые и конфигурируемые с помощью log4j в файл arm.statistic.log ссылка на который тоже в compile-\*.prooerties в поле og4j.appender.Stat.File

### 20. Добавляю модуль с отображением статистики. (build\_18\_add\_AdminStatPage)

Статистика должна писаться в файл arm.statistic.log (ссылка накоторый задатеся в compile-\*.prooerties в поле log4j.appender.Stat.File) по определнным правилам, т.к. потом идет разбор этого текстового файл в методе getStatistic() и parseStatFile() класса CommonServiceImpl. (Так что ничем лишним этот файл замусоривать нельзя)

Статистика не работала, так как в файл ничего не писалось. В классе GWTController условие

if( userService != null ){

statLog.info( "SERVICE " + request.getRequestedSessionId() + " " + userService.getCurrentUser().getUserName() + " " + getClass().getSimpleName() + " "+request.getRemoteAddr());

}

Не выполнялось никогда так как userService был всегда нулевой. Добвила аннотацию

**@Autowired**

public void setUserService(UserService userService) {

this.userService = userService;

}

и включила соостветсующий файл

import org.springframework.beans.factory.annotation.Autowired;

теперь UserService автоматически подхватывается и заполнятеся, запись в файл производиться и статистика показывается.

Добавляю все классы виджетов в client/module/admin/stat

Для отображения графиков используется классы библиотеки gxt.jar

com.extjs.gxt.charts.client.model

Для их работы необходимо:

А)Необходимо добавить в файл Interactice.gwt.xml строку:

<inherits name='com.extjs.gxt.charts.Chart'/>

B)Необходим в паку web/ добавить папку chart с файлом open-flash-chart.swf(из ресурсов самой библиотеки gxt.jar папка resources)

И в конце не забыть в панель навигации добавить

import pst.arm.client.modules.admin.stat.AdminStatPage

и строку с вызовом этой страницы.

### 21. Добавляю модуль администратора для создание пользователей(build\_19\_add\_AdminUserPage)

### 22. Добавляю модуль администратора для создание ролей(build\_20\_add\_AdminRolePage)

### 23. Добавляю весь код для создание модуля с данными (гридом, редактированием, поиском) и пример GridData

### Создание страницы с раскрывающимися панелями

Свойсвтво у калсса ContentPanel setCollapsible(true) - озанчает что панель можно будет прятать.

Долго не могла найти метод который позовляет раскрвать(закрвать) панель не по кнопке в правом углу заголовка а по щеочку по загаловку. Нашла этот метод, вот он

setTitleCollapse(true);

x-small-editor x-toolbar x-component x-border x-toolbar-layout-ct

#d0def0

package pst.arm.client.modules.test;

import com.extjs.gxt.ui.client.Style.Orientation;

import com.extjs.gxt.ui.client.Style.Scroll;

import com.extjs.gxt.ui.client.util.Margins;

import com.extjs.gxt.ui.client.widget.ContentPanel;

import com.extjs.gxt.ui.client.widget.Text;

import com.extjs.gxt.ui.client.widget.layout.FlowData;

import com.extjs.gxt.ui.client.widget.layout.RowData;

import com.extjs.gxt.ui.client.widget.layout.RowLayout;

/\*\*

\*

\* @author Admin

\*/

public class RowLayoutExample extends ContentPanel {

public RowLayoutExample() {

setScrollMode(Scroll.AUTOY);

//---------------------------------------------

ContentPanel panel\_1 = new ContentPanel();

panel\_1.setHeading("Панель 1");

panel\_1.setLayout(new RowLayout(Orientation.VERTICAL));

panel\_1.setFrame(false);

panel\_1.setCollapsible(true);

panel\_1.setAnimCollapse(false);

panel\_1.getHeader().setStyleAttribute("backgroundImage", "none");

panel\_1.getHeader().setStyleAttribute("backgroundColor", "#DFE8F6");

panel\_1.getHeader().setStyleAttribute("border", "1px solid #7CA4D9");

panel\_1.setTitleCollapse(true);

Text label1 = new Text("Test Label 1");

label1.addStyleName("pad-text");

label1.setStyleAttribute("backgroundColor", "white");

label1.setBorders(true);

Text label2 = new Text("Test Label 2");

label2.addStyleName("pad-text");

label2.setStyleAttribute("backgroundColor", "white");

label2.setBorders(true);

Text label3 = new Text("Test Label 3");

label3.addStyleName("pad-text");

label3.setStyleAttribute("backgroundColor", "white");

label3.setBorders(true);

panel\_1.add(label1, new RowData(-1,-1,new Margins(4)));

panel\_1.add(label2, new RowData(-1,-1,new Margins(0, 4, 0, 4)));

panel\_1.add(label3, new RowData(-1,-1,new Margins(4)));

add(panel\_1,new FlowData(5));

//---------------------------------------------

ContentPanel panel\_2 = new ContentPanel();

panel\_2.setHeading("RowLayout: Orientation set to horizontal");

panel\_2.setLayout(new RowLayout(Orientation.HORIZONTAL));

panel\_2.setHeight(100);

panel\_2.setFrame(false);

panel\_2.setAnimCollapse(false);

panel\_2.setCollapsible(true);

panel\_2.getHeader().setStyleAttribute("backgroundImage", "none");

panel\_2.getHeader().setStyleAttribute("backgroundColor", "#DFE8F6");

panel\_2.getHeader().setStyleAttribute("border", "1px solid #7CA4D9");

panel\_2.setTitleCollapse(true);

label1 = new Text("Панель 2");

label1.addStyleName("pad-text");

label1.setStyleAttribute("backgroundColor", "white");

label1.setBorders(true);

label2 = new Text("Test Label 2");

label2.addStyleName("pad-text");

label2.setStyleAttribute("backgroundColor", "white");

label2.setBorders(true);

label3 = new Text("Test Label 3");

label3.addStyleName("pad-text");

label3.setStyleAttribute("backgroundColor", "white");

label3.setBorders(true);

panel\_2.add(label1, new RowData(-1, -1, new Margins(4)));

panel\_2.add(label2, new RowData(-1, -1, new Margins(4, 0, 4, 0)));

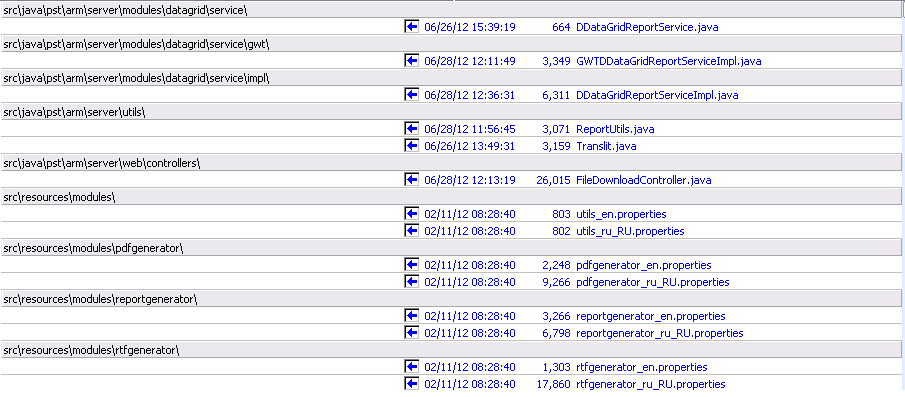
panel\_2.add(label3, new RowData(-1, -1, new Margins(4)));

add(panel\_2,new FlowData(5));

}

}

## Добавила классы и библиотеки связанные с формирование отчетов (см. svn версию 38266)

Используемые библиотеки: iText-2.1.7.jar, iText-rtf-2.1.7, poi-3.8-beta4-20110826.jar, poi-excelant-3.8-beta4-20110826.jar, poi-ooxml-3.8-beta4-20110826.jar, poi-ooxml-schemas-3.8-beta4-20110826.jar, poi-ooxml-schemas-3.8-beta4-20110826.jar, commons-lang-2.4.jar

**Библиотека iText** – формирование word-иx, pdf-их документов

import com.lowagie.\*

**Библиотека POI** - Формирование word-их документво

import org.apache.poi.hwpf.HWPFDocument;

Мне помог ресурс <http://jexp.ru/index.php/Java_Tutorial/PDF/PDF_Stamper>  генерация отчета pdf

## Как заставить аннотацию @Transactional работать.

1.В файле **applicationContext-jdbc.xml** прописать строку

**<tx:annotation-driven transaction-manager="transactionManagerArm"/>**

<bean id="transactionManagerArm" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSourceArm"/>

<qualifier value="txArm"/>

</bean>

У меня после этого перестал собираться проект, в интернете прочла что нужна библиотека **aspectjweaver.jar** после ее подключения проект собирался аннотация @Transactional не работала.

Оказывается еще надо было в файле **applicationContext.xml** написать

<context:component-scan base-package="pst.arm.server" >

**<context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>**

**</context:component-scan>**

<context:annotation-config/>

И в файле **dispatcher-servlet.xml**

<context:component-scan base-package="pst.arm.server**" use-default-filters="false">**

**<context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>**

**</context:component-scan>**

После этого все заработало, теперь транзакцию можно использовать так:

@Override

@Transactional(value = "txArm", rollbackFor = Throwable.class)

public ClientCard insert(final ClientCard domain) {

String sql = ‘’INSERT 1”;

SqlParameterSource namedParameters = new BeanPropertySqlParameterSource(domain);

KeyHolder keyHolder = new GeneratedKeyHolder();

Integer rows = getJdbcTemplate().update(sql, namedParameters, keyHolder, new String[]{"CLIENT\_ID"});

if (!rows.equals(1)) {

return null;

}

domain.setClientId(keyHolder.getKey().intValue());

sql = ‘’INSERT 2";

SqlParameterSource namedParametersRepr = new BeanPropertySqlParameterSource(domain);

Integer rows\_repr = getJdbcTemplate().update(sql, namedParametersRepr, keyHolder, new String[]{"CLIENTS\_REPR\_ID"});

if (rows\_repr.equals(1)) {

domain.setClientsReprId(keyHolder.getKey().intValue());

}

return domain;

}

## Scrum — методология управления разработкой информационных систем для [гибкой разработки программного обеспечения](http://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B1%D0%BA%D0%B0%D1%8F_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8).

Scrum чётко делает акцент на качественном контроле процесса разработки. Кроме управления проектами по разработке ПО Scrum может также использоваться в работе команд обслуживания программного обеспечения (software maintenance teams), или как подход управления разработкой и сопровождением программ

## The Apache Lucene — это [свободная](http://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) библиотека для высокоскоростного [полнотекстового поиска](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA),

написанная на [Java](http://ru.wikipedia.org/wiki/Java). Может быть использована для поиска в интернете и других областях компьютерной лингвистики (аналитическая философия).