Best Software Development Practices

by Jeff Zhuk

Struts – Spring - Hibernate

DataService & Semantic Frameworks (JavaSchool.com)
We have vast experience, and are ready to share it with you. We customize our training sessions around client needs, and turn case studies into brainstorms and prototyping efforts. We help you to address real-world problems, and show how to apply best practices and technologies in your specific environment, how to master enterprise web services and create Service Oriented Architecture environment while working with Eclipse, JBoss, IBM Web Sphere, Tibco, BEA and Oracle platforms.

We create custom programs for your specific needs, design a curriculum around your requirements, or create one based on your selections from our existing course offerings. Our current 2-3 day course listing includes:

- PROGRAMMING WEB SERVICES and INTRODUCTION TO BPEL
- WEB APPLICATIONS WITH JSP/Struts/JSTL and PORTLET STANDARDS (WSRP/JSR168)
- ADVANCED JDBC
- EJB PROGRAMMING
- ENTERPRISE APPLICATIONS WITH J2EE
- ENTERPRISE APPLICATIONS AND COLLABORATIVE ENGINEERING
- BPEL AND WEB SERVICES FRAMEWORKS
- WRITE ONCE: BUILDING REUSABLE ENTERPRISE COMPONENTS
- THE NEW GENERATION OF SOFTWARE for MULTIPLE CLIENTS
- J2EE DESIGN PATTERNS
- FUNDAMENTALS OF WIRELESS TECHNOLOGIES
- PROGRAMMING WAP APPLICATIONS
- JAVA CARD TECHNOLOGY AND RELATED APPLICATIONS
- J2ME AND SERVER SIDE DEVELOPMENT FOR SMS AND MESSAGING
- SPEECH TECHNOLOGIES FOR VOICE APPLICATIONS
- AN INTRODUCTION TO KNOWLEDGE TECHNOLOGIES
- DISTRIBUTED APPLICATIONS WITH JXTA FRAMEWORKS
- INTEGRATION OF SOFTWARE AND KNOWLEDGE ENGINEERING

Reference book: Jeff Zhuk,
Integration-Ready Architecture and Design.
http://www.amazon.com/exec/obidos/ASIN/0521525837
http://javaschool.com/about/publications.html
- Struts is the major framework for developing Web Applications
  - Extends Servlet-Controller in a function-specific **Action**
  - Uses *struts-config.xml file* to define all function-actions
  - Collects the data from the web forms into specific **ActionForm** classes that keep data state between requests
  - Maps each **Action** to its **ActionForm** in the *struts-config.xml*
  - Introduces a powerful set of tag libraries
  - And more…
Struts Frameworks for Web Application

- **Struts-config.xml**
- **Action-JSP map**
- **Web Tier**
  - A set of specific actions provide data processing and forward control to JSP
- **DAO Beans**
  - Create, Select, Update, Insert, Delete
  - Read SQL and replace variables with field values from a form
- **Database**

- **Client(s)**
- **Web Browser**
- **Http**
- **Post/Get**

- **Submit Form**
- **Display Page**

- **App Sever**

The Struts Action performs common operations. Each specific action extends Struts action and reuse common functionality. Usually Data Access Object (DAO) design...
Application Parameters in the web.xml

- WEB-INF
-  web.xml

```xml
<context-param>
  <param-name>ITSDataSource</param-name>
  <param-value>ITSDispDataSource</param-value>
</context-param>

<context-param>
  <param-name>roles</param-name>
  <param-value>Admin,Developer,Modeler,Configurator,Assembler</param-value>
</context-param>
```
<!-- Standard Action Servlet Configuration -->
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
</servlet>
Application Parameters in the web.xml

- WEB-INF
- web.xml

```xml
<!-- Standard Action Servlet Mapping -->
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

What does this mapping mean?

You can provide a URL to MyAction.class as http://myServer.com:8080/MyApplicationName/MyAction.do
Application Parameters in the web.xml

```
<!-- The Usual Welcome File and Error Page List -->
<welcome-file-list>
    <welcome-file>login.jsp</welcome-file>
</welcome-file-list>

<error-page>
    <error-code>404</error-code>
    <location>/WEB-INF/jsp/DisplayActionException.jsp</location>
</error-page>

<error-page>
    <error-code>500</error-code>
    <location>/WEB-INF/jsp/DisplayActionException.jsp</location>
</error-page>
```
<!-- Struts Tag Library Descriptors -->
<taglib>
<taglib-uri>/WEB-INF/tld/struts-bean.tld</taglib-uri>
<taglib-location>/WEB-INF/tld/struts-bean.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>/WEB-INF/tld/struts-html.tld</taglib-uri>
<taglib-location>/WEB-INF/tld/struts-html.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>/WEB-INF/tld/struts-logic.tld</taglib-uri>
<taglib-location>/WEB-INF/tld/struts-logic.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>/WEB-INF/tld/struts-nested.tld</taglib-uri>
<taglib-location>/WEB-INF/tld/struts-nested.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>/WEB-INF/tld/struts-tiles.tld</taglib-uri>
<taglib-location>/WEB-INF/tld/struts-tiles.tld</taglib-location>
</taglib>
Define Your Plan of Actions in the *struts-config.xml*

App-name.war
- WEB-INF
- struts-config.xml

```xml
<struts-config>

<!-- ==== Global Exception Definitions -->
<!-- Global Exceptions -->
<global-exceptions>
  <exception type="java.lang.Exception" key="none" handler="com.its.actions.WebExceptionHandler" />
</global-exceptions>

Define a unified approach to handling errors with your custom exception handler or use one from the DataService frameworks.
```
Define Your Plan of Actions in the struts-config.xml

As you can see we define a new form on-the-fly using Struts generic class DynaValidatorForm
Define Your Plan of Actions in the **struts-config.xml**

```xml
<!-- ====== Action Mapping Definitions -->

<action-mappings>
   <action name="logonForm"
      path="/login" type="com.its.actions.LoginAction">
      <forward name="success" path="/WEB-INF/jsp/Topics.jsp" />
      <forward name="failure" path="index.jsp" />
   </action>
   <action path="/interpret" type="com.its.actions.ITSAction">
      <forward name="topic" path="/WEB-INF/jsp/Topics.jsp" />
      <forward name="its" path="/WEB-INF/jsp/its.jsp" />
      <forward name="failure" path="index.jsp" />
   </action>
</action-mappings>
```
public class LoginAction extends Action {
    public ActionForward execute(
        ActionMapping map,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response)
    throws Exception {

        // business logics

        return map.findForward("success");
    }
}

Now, look one slide back and figure out that the “success” will pass the control to the ="/WEB-INF/jsp/Topics.jsp"
We are going to write JSP

What do you remember about JSP?

- JSP is a mix of Java and HTML code

- Can use JavaBeans as “backend” helpers

- JSP can use JSTL and custom tags to minimize Java code on JSP pages

- JSP is compiled to servlet at run-time by a JSP engine (usually a part of an application server)
Write the Topic.jsp page

```jsp
<%@taglib uri="/WEB-INF/tld/struts-logic.tld" prefix="logic"%>
<%@taglib uri="/WEB-INF/tld/struts-html.tld" prefix="html" %>
<%@taglib uri="/WEB-INF/tld/struts-bean.tld" prefix="bean" %>
<%@ page import="java.util.*" %>
<%@ page import="com.its.util.*" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<%
String worldName = (String)session.getAttribute("worldName");
%>
<html:html>
<html:errors/>
<html:form focus="text" action="CategoryAdd"
    onsubmit="return validateCategoryForm(this);">
    <html:hidden property="id" />
</html:form>

Find a Java scriplet on the page and compare it with the Struts HTML tags. Tags look much better, right?
```
Continue the Topic.jsp page with more Struts Tags and Internationalization

```html
<table>
<tr>
  <td><bean:message key="category.label.text" /></td>
  <td><html:textarea property="text" cols="100" rows="5" /></td>
</tr>
</table>

<!-- more … -->
<tr>
  <td>
    <html:submit><bean:message key="category.label.submit" />
    <html:submit>
  </td>
</tr>
</table>
</html:form>

Note the “<bean:message” tag. This tag refers to messages stored in the application.properties file. In this file the key “category.label.text” has the value, for example, “Category”. The beauty of this approach is in possibility to have multiple files, like application.en.properties, application.es.properties, and etc. with values written in different languages.
Displaytag Custom Library

<%@ taglib uri="http://displaytag.sf.net" prefix="display" %>
<%@ taglib uri="/WEB-INF/tld/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/tld/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/tld/struts-logic.tld" prefix="logic" %>

<display:table name="list_categories" requestURI="/CategoryList.do"
class="list">
<display:column property="id" sortable="true" />
<display:column property="text" sortable="true" />
<display:column property="type" sortable="true" />
<display:column url="/CategoryView.do" paramId="id" paramProperty="id">
  <bean:message key="category.label.mod" />
</display:column>
<display:column url="/CategoryDel.do" paramId="id" paramProperty="id">
  <bean:message key="category.label.del" />
</display:column>
</display:table>

Download the Displaytag library
from http://displaytag.sf.net
Example from the Login.jsp Page

```xml
<logic:messagesPresent>
    <UL>
        <html:messages id="error">
            <LI style="font-weight: bold; color: red;"><bean:write name="error"/></LI>
        </html:messages>
    </UL>
</logic:messagesPresent>
<html:form action="/login.do?action=logon" method="post" focus="username" onsubmit="return checkRegForm(this)" >
    <table align="center">
        <tr><td align="right"><bean:message key="login.label.username" />: </td>
            <td align="left"><html:text property="username" /></td></tr>
        <tr><td align="right"><bean:message key="login.label.password" />:</td>
            <td align="left"><html:password property="password" /></td></tr>
        <tr><td align=right><html:submit property="submit" value="Login" /></td>
    </table>
</html:form>
```

Note, 3 Struts tag libraries: “logic”, “bean”, and “html”.

Look at the form tag and find out a parameter used in the POST method.
Struts Summary

• Extends Servlet-Controller ....

• Uses *struts-config.xml file* to ....

• Collects the data from the web forms into ...

• Maps each *Action* to its ... in the ...

• Introduces a powerful set of ...
Struts Summary

• Extends Servlet-Controller in a function-specific **Action**

• Uses *struts-config.xml* file to define all function-actions

• Collects the data from the web forms into specific **ActionForm** classes that keep data state between requests

• Maps each **Action** to its **ActionForm** in the *struts-config.xml*

• Introduces a powerful set of tag libraries

• And more…
Displaytag Custom Library
Makes Easy Creating Tables
With Sorted Columns

<table>
<thead>
<tr>
<th>Id</th>
<th>Category.text</th>
<th>TopicName</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>General</td>
<td>UML</td>
</tr>
<tr>
<td>2</td>
<td>General</td>
<td>OOA</td>
</tr>
<tr>
<td>4</td>
<td>Try</td>
<td>demo</td>
</tr>
</tbody>
</table>

Certified Integration Architect Exam
Here is the Login.jsp Page

Register to Enter the Certified Integration Architect Exam and Training Facilities

or Login to proceed to the next step.

Login Name: jzhuk
Password: ********

Login  Cancel
public User getUserById(int id, String psw) {
  try {
    DataManager dm = DataManager.init(); // locate DB connection pool
    Connection conn = dm.getConnection(); // get individual connection
    PreparedStatement prpStmt = conn.prepareStatement("SELECT username, roleID from users where userID=? and psw=?");
    prpStmt.setInt(1,id);
    prpStmt.setString(2,psw);
    User user = new User();
    ResultSet rs = prpStmt.executeQuery();
    while (rs.next()) {
      String username = rs.getString(1);
      int roleID = rs.getInt(2);
      user.setName(username);
      user.setRoleID(roleID);
    }
    rs.close();
    prpStmt.close();
  } catch (SQLException e) {
    e.printStackTrace();
    return null;
  }
  return user;
}
Struts – Spring - Hibernate

**Struts**
- MVC
- Action
- ActionForm
- Custom Tags

**Spring**
- Hibernate Connection and Session Management
- Business Services

**Hibernate**
- Object-to-Relational Mapping
- Generating DAO classes and SQL based on DB schema
- Persistence Layer

DB
Hibernate

• Uses OO query language called HQL

• Uses objects instead of tables and fields instead of columns

• Provides object-to-relational mapping for most DBs

• Separates data layer from business logics

• Uses DB connection info to retrieve DB schema

• Generates DAO beans with data fields mapping table columns

• Generates Insert/Update/Delete/Select statements for DB tables
Hibernate Synchronizer

Select Windows – Preferences – Hibernate Synchronizer ... and the miracle happens: Hibernate connects to the DB, retrieves the schema, and generates DAO classes and SQL for basic operations on DB tables.
<beans>
  <!-- PERSISTENCE DEFINITIONS -->
  <bean id="myDataSource"
       class="org.springframework.jndi.JndiObjectFactoryBean">
    <property name="resourceRef"><value>true</value></property>
    <property name="jndiName">
      <value>jdbc/javatest</value>
    </property>
  </bean>

  <!-- Connect to Hibernate and match your "dataSource" definition -->
  <bean id="mySessionFactory"
       class="org.springframework.orm.hibernate.LocalSessionFactoryBean">
    <property name="mappingResources">
      <list>
        <value>CIAExamAnswer.hbm.xml</value>
        <value>UserRoles.hbm.xml</value>
        <value>InstructorCategory.hbm.xml</value>
      </list>
    </property>
  </bean>
</beans>
Spring Maps Data Source Dialect and Provides Transaction Management

<property name="hibernateProperties">
    <props>
        <prop key="hibernate.dialect">
            net.sf.hibernate.dialect.PostgreSQLDialect</prop>
        <prop key="hibernate.show_sql">true</prop>
        <prop key="hibernate.cglib.use_reflection_optimizer">true</prop>
    </props>
</property>

<property name="dataSource">
    <ref bean="myDataSource"/>
</property>
</bean>

<!-- Transaction manager for a single Hibernate SessionFactory -->
<bean id="myTransactionManager"
    class="org.springframework.orm.hibernate.HibernateTransactionManager">
    <property name="sessionFactory">
        <ref local="mySessionFactory"/>
    </property>
</bean>
Spring and Hibernate
Reduce Business Code

The sessionFactory property and the mySessionFactory bean are related in the Spring configuration file.

Spring creates described objects and factories that instantiate Hibernate DAO classes at run-time.

Spring simplifies the Hibernate configuration that otherwise would be stored in the hibernate.cfg.xml file.

The bottom line: Spring and Hibernate working together reduce your business code, especially when you operate with simple data records that reflect full table structure.
public class InstructorCategory extends BaseInstructorCategory {

    /**************************************************************************/
    public InstructorCategory () {
        super();
    }
    /**
     * Constructor for primary key
     */
    public InstructorCategory (com.its.cia.persistence.Users _user,
                                com.its.cia.persistenceTestCategory _category) {
        super (_user, _category);
    }

    /**************************************************************************/
}
Spring and Hibernate are perfect solutions when development can transit from SQL to objects and rely on Object-Relational Mapping (ORM) mechanisms and automatic SQL generation.

There are cases when developers want to keep full control on complex SQL statements, when creating and debugging SQL is an essential part of the development efforts.

The **Data Service & Semantic frameworks** by ITS, Inc. focus on these cases and alone with data handling provide mechanisms for application monitoring, diagnostics and semantic self-awareness.
Struts/Portal – DataServiceImpl

DataServiceImpl & Semantic Frameworks by ITS, Inc.

- DataAction extends StrutsAction
- PortletDataAction extends StrutsPortlet
- Takes care of connections, and data types
- No ORM, You master SQL and store statements in files in the WEB-INF/sql-folder

Persistence Layer

- Easy, no configuration files
- Self-Testing & Diagnostics Layer
- Service descriptions, rules & scenarios

DB
Data Intensive Web Application

- **Struts-config.xml**
- **Action-JSP map**

**Client(s) → Web Browser → Http Post/Get**

**Web Tier**
- Data Action or PortletDataAction
- Extract business data
- Receive the Forward value and map to a JSP

**Web Tier**
- JSPs (Forms)

**A set of Specific Actions**
- Use **DataService** to execute dynamic queries and provide object processing

**App Server**
- Create, Select, Update, Insert, Delete

**Database**
- Read SQL and replace variables with field values from a form

**Struts: DataAction extends StrutsAction**

**Portlets: PortletDataAction extends StrutsPortlet**

Along with **DataService** provide data access, diagnostics and common semantics

© ITS, Inc. | dean@JavaSchool.com
Design and Code Hints

Use common data services, avoid code duplications, and focus more on a business side of applications.

WEB-INF/sql/getUser.sql

Select * from users where loginName = ‘:loginName’

keys.put("loginName", form.getLoginName()); // common HashMap keys

List records = DataService.getData("getUser", keys, User.class);

User user = (User) records.get(0);

Don’t mess with SQL in Java code. Keep it in separate files in the SQL directory. Connections, Pooling, ResultSet Processing, and more reusable functions are provided with the DataService methods.

Don’t duplicate this code, use it, and focus on your specific business!
HTTPServlet

OMD

Form

Action

DataAction

data: HashMap
execute()
abstract perform()

SpecificAction1

specificData
perform()

SpecificAction2

specificData
perform()

Web Container and Application Server

ServletContextListener interface

ITSServletContextListener

c秉持Initialized() – init data
initialize() – to be overridden by a subclass

MyServletContextListener

initialize() – custom initialization

DataService

dataSource DataSource
dataSources Hashtable
getData(sqlName, map, bean.class) : List
setData(sqlName, map)
setDataSource(DS)

Color Legend: ___ Java library
___ Custom Library in the com.util.jar
___ Java classes we need to code
DataAction | Specific Service Action | DataService

Events/Operations

Client

Http

Post/Get

Extract data from the page and call a proper service

Request SQL execution with SQL statement name and map of parameters

Generic data processing:
Retrieve SQL from resources; Replace run-time variables; Execute SQL, process ResultSet (if any) and return a list of object-beans

Specific processing:
Work with objects and prepare data for JSP

Pass data saved in data beans, Session, or Request objects to JSP

Map the Forward value to a proper JSP (via Struts-config.xml) and display the next JSP page
public class LoginAction extends DataAction { // PortletDataAction in portlets

    public String perform(HttpServletRequest request) throws Exception {
        …..
        List beans = DataService.getData("getLogin", // name of the SQL file is “getLogin.sql”
                                keys, // HashMap of key-values collected by DataAction
                                LoginBean.class); // class that matches expected record structure
            
        if(beans.size() == 1) { // SUCCESS!
            LoginBean user = (LoginBean) beans.get(0);
            session.setAttribute("user", user);
            return "success";
        }
        …..
        return "failure";
    }
}
LoginBean Class Example

package beans;

/**
 * The LoginBean class matches the record selected by the getLogin.sql
 * A hint: provide variable names in alphabetical order
 */
public class LoginBean {
    private String loginName;
    private String password;

    public String getLoginName() { return loginName; }
    public String getPassword() { return password; }
    public void setLoginName(String name) { loginName = name; }
    public void setPassword(String psw) { password = psw; }
}

select username, password from LoginTable
where username = ':loginName' and password = ':password'
SQL Statements

SQL statements are stored as separate files in the SQL directory in the WEB-INF area.

Samples below demonstrate how SQL statements and their usage by DataService methods:

```java
List beans = DataService.getData("getLogin", keyMap, LoginBean.class);
Two lines below is stored in the “getLogin.sql” file

select username, password from LoginTable
where username = ':loginName' and password = ':password'

int nRecords = DataService.setData("insertUser", map);
The line below is stored in the “insertUser.sql” file

Insert into LoginTable values(':loginName', ':password')
```

Note that run-time variable names follow the “:" character.
Data Status

Diagnostics

NumberOfChanges:
NumberOfQueries:
FirstConnectionTime:
LastConnectionTime:
NumberOfErrors:
Errors:
MaxQueryTime:
LongestQuery:
Remote/Data Status

Local/Data Status

Legend: ___ Java library
___ Custom Library in the com.util.jar
___ Java classes we need to code

Web Container and Application Server

DbContextListener interface

ITSServletContextListener
contextInitialized() – init data
initialize() – to be overridden by a subclass

MyServletContextListener
initialize() – custom initialization

DataService
dataSource DataSource
dataSources Hashtable
getData(sqlName, map, bean.class) : List
setData(sqlName, map)
setDataSource(DS)

Works with

Uses

Reports run-time status of data sources
HTTPServlet

Form

Action

Uses

DataAction

keys: HashMap
execute()
abstract perform()

SpecificAction1

specificData
perform()

SpecificAction2

specificData
perform()

Legend:

___ Java library
___ Custom Library in the com.util.jar
___ Java classes we need to code

Self-Test Facilities
Web Application Unit Test

Test a selected action or a sequence of actions
Debug a specified method in a specified class
Check data with your query

Remote/test.do
Local/test.do
Local/Test All Actions
Automated System Test Scenario

<Scenario name="Test User Management Functions">

<Action name="/LoginAction">

    <!-- Replace default form data -->

    <initData name1="value1" />

    <!-- Set attributes for Request and Session objects -->

    <requestAttributes name1="value1" />

    <sessionAttributes name1="value1" />

    <!-- Set expectations -->

    <expectedResults location=... name=.... Value=.... />

</Action>

<Action name="/CreateUserAction">

............

</Action>

</Scenario>
DataService API

Include the library “com.its.util.jar” in the CLASSPATH and import com.its.util.DataService

// execute insert/delete/update SQL statements stored in the “sqlLocation”
@ param sqlStatementName for example “getLogin” stored as the “getLogin.sql”
@ param map of key-values to replace SQL <<keys>> with run-time values
@ return numberOfRowsAffected
public static int setData(String sqlStatementName, HashMap map)

// use other than “DataSource” connection pool
public static int setData(String sqlStatementName, HashMap map, String dsName)

// execute select statement and return a list of record-beans
@ param sqlStatementName for example “getLogin” stored as the “getLogin.sql”
@ param map of key-values to replace SQL <<keys>> with run-time values
@ param beanClass (e.g. LoginBean.class) supports records retrieved by the SQL statement
@ return list of objects of the beanClass
public static List getData(String sqlStatementName, HashMap map, Class beanClass)

// use other than “DataSource” connection pool
public static List getData(String sqlName, HashMap map, Class beanClass, String dsName)

@param dataSource your DataSource specified in JNDI context
@param dataSourceName of your DataSource specified as JNDI name
public static void setDataSource(DataSource dataSource, String dsName)
public static void setDataSource(DataSource dataSource)
MoreDataService API

// execute insert/delete/update SQL statements
@ param sqlStatement
@ return numberOfRowsEffected
public static int setData(String sqlStatement)

// use other than “DataSource” connection pool
public static int setData(String sqlStatement, String dsName)

// execute select statement and return a list of record-beans
@ param sqlStatement
@ param beanClass supports records retrieved by the SQL statement
@ return list of objects of the beanClass
public static List getData(String sqlStatement, Class beanClass)

// use other than “DataSource” connection pool
public static List getData(String sqlStatement, Class beanClass, String dsName)

// If application creates data from scratch – no SQL is needed
boolean createTable(String tableName, Class class)
Example:
DataService.createTable(“LoginTable”, LoginBean.class);

int insert(String tableName, Object[] objects)
Example:
LoginBean[] logins; // array of beans populated in an action
int nRows = DataService.insert(“LoginBean”, logins);
Use DataService Frameworks to Define Custom Actions in the **struts-config.xml**

App-name.war
- WEB-INF
  - struts-config.xml

```xml
<action path="/CheckDataConsistency"
   type="com.its.util.actions.CheckDataAction">
  <description>
    - Search for duplicate records
    
    If found, the "Remove Duplicates" link will be provided below.
  </description>
</action>
```
Validate and sync user data

Application functionality is highly sensitive to data quality. The application is extended with self-control and diagnostics functions that can detect and fix data errors. It is highly recommended to check and fix duplicate records first. The level of troubleshooting messages is LOW. Click here to change this.

**Check Data Consistency** - Search for duplicate records
If found, the "Remove Duplicates" link will be provided below

**Check Data**: Enter your QUERY to check data in the selected data source

Select from known data sources or Enter a connection URL in the text field below

Example: com.its.mail.Email.sendMail(to=jzhuk@javalchool.com, msg=Hello)
Problem
Even in the professional web applications the error screens often look not very professional and they differ from application to application.

Solution
Handling exceptions in web applications is partially standardized by Struts frameworks. Struts configuration file can include pointers to error handler actions. The WebExceptionHandler class extends Struts Error Handler to provide a standard look and feel to all error pages across web applications. It is recommended to include the section below in the struts configuration file to point to a standard handler that generates a standard error screen.
UseDataService Diagnostics Facilities

// Provide this code your Java class
} catch(Exception e) {
    Stats.addAppList(appName,
    "WebDisp Errors",
    e.getMessage());
}

This code will communicate the error to the DataService diagnostics facilities. See on the right side how such records look on the web page

<!– Provide these lines in your web.xml file -->
<global-exceptions>
    <exception type="java.lang.Exception" key="none"
        handler="com.its.actions.WebExceptionHandler" >
    </exception>
</global-exceptions>
Complementary Semantic Frameworks: Capture Service Descriptions

Register services and service scenarios (one at a time)

Service Signature, for example, com.its.util.XHandler.parse(String xml), or WSDL location

Service Descriptions (keep in mind that these descriptions will be used by search facilities).
## Capture Rules and Scenarios

### Rules

<table>
<thead>
<tr>
<th>RULEID</th>
<th>RULE</th>
<th>USERID</th>
<th>DATE</th>
<th>SUBJECT</th>
<th>IMPLEMENTED</th>
</tr>
</thead>
</table>

...Delete Checked Items...

Add more rules and reconciliation scenarios (one at a time) for DataSync

...Check and add a rule or scenario...

Member of a privileged group, you can [Change debug level to LOW (less messages)](#), [Check Data Status](#), [Sync data](#), [Capture rules](#) and [Register Services](#).
Business Intelligence Repository
Based on Data Models, Rules & Scenarios
To Enable Collaborative Decision Making,
Data Reconciliation and Smart Search

Conversational Interface

Integrity Rules
(Inference engine)

Optimization Solutions

Situational Rules and Scenarios

Business and Technology Mappings

Operations

Events

Procedures

Specific Facts and Rules

Generic Facts and Rules

Real-Time Input Data

Behavior & Services

SME

* Rules Collector
* Knowledge-Driven Architecture

Integrated Software & Knowledge Engineering
Commonsense Reasoning

© ITS, Inc. | dean@JavaSchool.com
Ontology tools can get more meaning from business descriptions, rules and scenarios, while improving decision support and automation of development, testing and business processes.
Web Application Frameworks

Summary/Repetition

1. **Struts**

- **Struts Frameworks**
  - Extends Servlet-Controller in a function-specific *Action*
  - Uses `struts-config.xml` file to define all function-actions
  - Collects the data from the web forms into specific *ActionForm* classes that keep data state between requests
  - Maps each *Action* to its *ActionForm* in the `struts-config.xml`
  - Introduces a powerful set of tag libraries
  - And more…

The Struts Action performs common operations. Each specific action extends Struts action and reuse common functionality. Usually Data Access Object (DAO) design pattern is implemented with plain Java classes, Java Beans, or EJBs.
Web Application Frameworks

Summary/Repetition

1. Struts
2. Spring
3. Hibernate
4. DataService
   - Complementary to Struts and Portlets Semantic
   - Frameworks by ITS, Inc.