How To Develop A Simple CRUD Function In ActiveNetServlet

History

07/22/2015,Nan Liu: "Initial.Simple CRUD Development For ANServlet."

Preparation

ActiveNetServlet develop environment has been setup, see Reference Document. And if you start the servlet container, you can access http://localhost:8080/ActiveNetServlet/adminLogin.sdi .

Problem

Develop ‘Zoo Manager’, this is a single-table-crud function.

Create Database Table

- Edit `[source folder]/DBSchema/ActiveNetERD.vsd`
- Use AAAExportXml.vsd generate ActiveNetErd.xml
- Use AnUpsizer.exe upgrade db

Note:You can use Microsoft SQL Server Management Studio to create `dbo.ZOO`.

Using AdministrativeTable(Not Recommended)

Add AvailabelOption & Menu

- create `AvailabelOption.enmZoo`, cache it, and not allowed duplicated value.

```
public class AvailabelOption implements SortableItem {
    // ...
    private static final AvailableOption[] available_options[] = {
        // ...
        new AvailableOption(enmZoo, "Administration: Location - Zoos", "Zoo", SystemInfo.License.none.key()),
    }
}
```

- Add Menu into `ANServlet.Menu` class

```
public class Menu implements SortableItem, Serializable {
    // ...
    public static synchronized void constructNavBar(DBConnection dbc) {
        // ...
        Menu loc = new Menu("admin_locations", "Locations");
        // ...
        loc.add(new NavBarMenuItem("ZooAdmin", "Zoos", "", sdi_main_frame, "Manage Zoos",
                AvailableOption.enmZoo, false, NavBarMenuItem.DisplayColumn.secondary));
        // ...
    }
}
```

Create Business Class

Note: You must create Business-Class in package `ANservlet`

- create class `ANServlet.Zoo`

```
package ANServlet;
// ...
public class Zoo extends AdministrativeTable implements HTMLList, SortableItem, AdministrativeObject, Serializable {
    // ...
}
```

- static fields for database, html, authority:

```
public class Zoo extends AdministrativeTable implements HTMLList, SortableItem, AdministrativeObject, Serializable {
    // ...
    public static final String  sql_table_name      = "ZOO";
    public static final String  key_field_name      = "zoo_id";
    private static final String html_key            = "Zoo";
    private static final int    profile_authority   = AvailableOption.enmZoo;
    private static final String list_name           = html_key + "List";
    private static final String change_form_name    = "Change" + html_key;
    // ...
}
```

- fields for table columns:

```
public class Zoo extends AdministrativeTable implements HTMLList, SortableItem, AdministrativeObject, Serializable {
    // ...
    private int     zoo_id  = -1;   // primary-key
    private String  zoo_name;
    private String  zoo_address;
    private Date    zoo_created;
    private Date    zoo_updated;
    private String  zoo_creater;
    private String  zoo_updater;
    private String  zoo_type;
    // ...
}
```

- `Zoo.doUpdate` for save or update db-accessing.

```
public class Zoo extends AdministrativeTable implements HTMLList, SortableItem, AdministrativeObject, Serializable {
    // ...
    @Override
    public boolean dbUpdate(DBConnection dbc, Parameters p, SystemUser system_user) throws SDIException, SQLException {
        ANDBRecordUpdater dbru = new ANDBRecordUpdater(dbc, sql_table_name, key_field_name, zoo_id);
        dbru.addColumn("zoo_name", zoo_name);
        dbru.addColumn("zoo_address", zoo_address);
        dbru.addColumn("zoo_created", zoo_id == -1 ? new Date() : zoo_created);
        dbru.addColumn("zoo_updated", new Date());
        dbru.addColumn("zoo_creater", zoo_id == -1 ? system_user.name() : zoo_creater);
        dbru.addColumn("zoo_updater", system_user.name());
        dbru.addColumn("zoo_type", zoo_type);
        zoo_id = dbru.dbUpdate(system_user);
        return true;
    }
    // ...
}
```

- `Zoo.greaterThan` for sorting, usually called by `ActiveNetLib.Tools.SortMethods`
- `Zoo.admin` for showing list.

```
public class Zoo extends AdministrativeTable implements HTMLList, SortableItem, AdministrativeObject, Serializable {
    // ...
    public static void admin(DBConnectionManager dbcm, HTML html, Parameters p, HttpServletRequest req,
            HttpServletResponse res) throws ServletException, IOException {
        // read zoo-list from cache, and forward to list page
        adminList(OrgContext.getZoos(), list_name, profile_authority, html, p, req, res);
    }
    // ...
}
```

- `Zoo.adminChange` for edit-form, load item by id usually.

```
public class Zoo extends AdministrativeTable implements HTMLList, SortableItem, AdministrativeObject, Serializable {
    // ...
    public static void adminChange(DBConnectionManager dbcm, HTML html, Parameters p, HttpServletRequest req,
            HttpServletResponse res) throws ServletException, IOException {
        // load Zoo item and forward to input page
        adminChange(sql_table_name, find(p.getInt(key_field_name)), profile_authority, change_form_name, dbcm, html, p,
                req, res);
    }
```

```
    // ...
}
```

- `Zoo.adminProcessChange` for save or update.

```
public class Zoo extends AdministrativeTable implements HTMLList, SortableItem, AdministrativeObject, Serializable {
    // ...
    public static void adminProcessChange(DBConnectionManager dbcm, HTML html, Parameters p, HttpServletRequest req,
            HttpServletResponse res) throws ServletException, IOException {
        // (1) build a PO from Parameter
        Zoo c = null;
        try {
            c = new Zoo(p);
        } catch (NumberFormatException e) {
            p.put(HTML.param_error_msg, e.getMessage());
            html.respond(p, res, change_form_name);
            return;
        }
        // (2) save or update successfully, forward to list page
        if (adminProcessChange(sql_table_name, c, profile_authority, dbcm, html, p, req, res)) {
            admin(dbcm, html, p, req, res);
            return;
        }
        // (2) if failure, forward to inpu page
        if (p.get(HTML.param_error_msg) != null) {
            adminChange(sql_table_name, c, profile_authority, change_form_name, dbcm, html, p, req, res);
        }
    }
    // ...
}
```

- `Zoo.adminDelete` for delete.

```
public class Zoo extends AdministrativeTable implements HTMLList, SortableItem, AdministrativeObject, Serializable {
    // ...
    public static void adminDelete(DBConnectionManager dbcm, HTML html, Parameters p, HttpServletRequest req,
            HttpServletResponse res) throws ServletException, IOException {
        // (1) get param & find zoo item
        Zoo c = find(p.getInt("id"));
        if (c.id() <= 0) {
            HTML.sendHttpError(res, HttpServletResponse.SC_BAD_REQUEST);
            return;
        }
        // (2) delete item, forward to list page
        if (adminDelete(dbcm, sql_table_name, key_field_name, c, profile_authority, true, p, req, res)) {
            admin(dbcm, html, p, req, res);
            return;
        }
        // (3) if failure, forward to input page
        if (p.get(HTML.param_error_msg) != null) {
            adminChange(dbcm, html, p, req, res);
        }
    }
    // ...
}
```

- `Zoo.find` for business.

```
public class Zoo extends AdministrativeTable implements HTMLList, SortableItem, AdministrativeObject, Serializable {
    // ...
    public static Zoo find(int zoo_id) {
        for (Zoo c : OrgContext.getZoos()) {
            if (c.zoo_id == zoo_id) {
                return c;
            }
        }
        return new Zoo();
    }
    public static Zoo find(String zoo_name) {
```

```
        for (Zoo c : OrgContext.getZoos()) {
            if (c.zoo_name.equalsIgnoreCase(zoo_name)) {
                return c;
            }
        }
        return new Zoo();
    }
    // ...
}
```

Enable Caching

Note: This is not a required step! if not enable caching for previous step should be access db to find Zoo item.

- define a `List` field in `OrgContext` for Cache, and `Zoo.getFullCache`, `Zoo.setCache`, `Zoo.getFieldName`

```
public class OrgContext {
    // ...
    private List<Zoo> zoos = null;

    public static List<Zoo> getZoos() {
        OrgContext ctx = getOrgContext();
        return ctx.zoos;
    }

    public static void setZoos(List<Zoo> zoos) {
        OrgContext ctx = getOrgContext();
        ctx.zoos = zoos;
    }
    // ...
}

public class Zoo extends AdministrativeTable implements HTMLList, SortableItem, AdministrativeObject, Serializable {
    // ...
    public static Object getFullCache() {
        return OrgContext.getZoos();
    }
    public Object getFieldValue(Field f) {
        try {
            return f.get(this);
        } catch (Exception e) {
            AppLogger.error(e);
        }
        return null;
    }
    public void setCache(List<Zoo> cache) {
        OrgContext.setZoos(cache);
    }
}
```

- append `Zoo` to `AdministrativeTable.cached_classed`

```
public abstract class AdministrativeTable {
    // ...
    private static final String[] cached_classes = {
        // ...
        "Zoo",
    };
    // ...
}
```

Create html pages

Note: Filename must be match with field defined in Zoo class, for example, `Zoo.list_name = "ZooList"` matched `ZooList.html`

- `ZooList.html` for list page.

```
^DocType^
<html>
```

```
<head>
^HTMLHead("Zoo List")^
</head>

<body ^bodyoptions^>

^PageHeadBeg^Zoo List^PageHeadEnd^
^OverallBeg^

<table class="compact" border=0 cellpadding=2 cellspacing=1 width="100%">
^ListAddNewBeg^<a href="~adminChange.sdi?oc=`oc`&zoo_id=-1~">^ListAddNewEnd^
    <table class="tableList">
        <tr>
            <th class="mwf_col">Name</th>
            <th class="owf_col">Type</th>
            <th class="owf_col">Address</th>
            <th class="owf_col">Created</th>
            <th class="owf_col">Updated</th>
            <th class="owf_col">Creater</th>
            <th class="owf_col">Updater</th>
        </tr>
        `rows`
        </tbody>
    </table>
</table>

^OverallEnd^
</body>
</html>
```

- `ChangeZoo.html` for edit form

```
^DocType^
<html>
<head>
<!-- ^HTMLHead("Change Zoo")^ -->
</head>
<body ^bodyoptions^ onLoad="tabFirstField()">
<script type="text/javascript" language="JavaScript">
<!-- (1) validate form -->
<!-- Begin
function zooformValidField(field) {
    with(field.form){
        if(field==zoo_name)return !empty(field,"the zoo name");

        return true;

    }
}
function checkform(form) {
    if (!allFieldsValid(form)) return false;
    return true;
}
// End -->
</script>
^PageBegin("change_zoo_form","Change Zoo")^
    ^ErrorMsg("`errormsg`")^
    <!-- (2) input form -->
    <form action="~adminProcessChange.sdi~" METHOD="POST" ENCTYPE="application/x-www-form-urlencoded"
            name="zooform" onSubmit="return checkform(this)">
        <input type="hidden" name="oc" value=`oc`>
        <input type="hidden" name="zoo_id" value="`zoo_id`">
        <input type="hidden" name="zoo_created" value="`zoo_created`">
        <input type="hidden" name="zoo_updated" value="`zoo_updated`">
        <input type="hidden" name="zoo_creater" value="`zoo_creater`">
        <input type="hidden" name="zoo_updater" value="`zoo_updater`">
        <input type="hidden" name="hidemod" value="`hidemod`">

        ^GroupBegin("general_information","General Information")^
        <table class="tableForm">
```

```
            <tr>
                <th align=right>Zoo Name</td>
                <td><input type="text" name="zoo_name" value="`zoo_name`" size=25 maxlength=50></td>
            </tr>
            <tr>
                <th align=right>Address</td>
                <td><input type="text" name="zoo_address" value="`zoo_address`" size=25 maxlength=500></td>
           </tr>
            <tr>
                <th align=right>Type</td>
                <td>
                    <select name="zoo_type">
                        <option value="">zoo-type</option>
                        <option value="SAFARI">safari zoo</option>
                        <option value="MARINE">marine zoo</option>
                    </select>
                    <script type="text/javascript">
                        $('select[name="zoo_type"]').val(``zoo_type`');
                    </script>
                </td>
            </tr>
        </table>
        ^GroupEnd^

        <input type="hidden" name="submit-buttons" value="`submit-buttons`">
        <input type="hidden" name="d-f" value="`d-f`">
        <tr>
            <td class="outGroupButtons">`submit-buttons`</td>
        </tr>
    </form>
^PageEnd^
</body>
</html>
```

- `ZooListRow.html` for list table rows.

```
<tr>
    <td class="mwf_col"><a href="~adminChange.sdi?oc=`oc`&zoo_id=`zoo_id`~">`zoo_name`</a></td>
    <td class="owf_col">`zoo_address`</td>
    <td class="owf_col">`zoo_type`</td>
    <td class="owf_col">`zoo_created`</td>
    <td class="owf_col">`zoo_updated`</td>
    <td class="owf_col">`zoo_creater`</td>
    <td class="owf_col">`zoo_updater`</td>
</tr>
```

Using HTMLObject & BusinessObject(Recommended)

Add AvailabelOption & Menu

Almost same as Using AdministractiveTable, and difference is:

```
new AvailableOption(enmZooHtml, "Administration: Location - Zoos", HTMLObject.package_name+".ZooHTML", SystemInfo.License.none.key())
```

Create ZooBO

- create class `ANBusinessObjects.ZooBO`, this is a BusinessObject for logic & db-access.

```
public class ZooBO extends BusinessObject implements SortableItem, Serializable {
    // (1)static field for db, authoriy, Note: not for html view name
    // (2)field for db-table
    // (3)implement write same as Zoo.dbUpdate
    // (4)implement greaterThan same as Zoo.greaterThan
}
```

- `find` & `findList` for search zoo, following example is a typical method for accessing database

```
public static ZooFormBO find(DBConnectionManager dbcm, int zoo_id) {
```

```
    DBConnection dbc = null;
    try {
        // (1) get connection
        dbc = dbcm.getConnection();

        // (2) build sql script
        String sql = "select * from " + dbc.table(sql_table_name) + " where zoo_id = " + dbc.put(zoo_id);

        // (3) execute sql
        DBRecord dbr = dbc.getRecordSet(sql);

        // (4) process result
        if (dbr.nextRecord()) {
            return new ZooFormBO(dbr);
        }
    } catch (Exception e) {
        // (5) must logging error
        AppLogger.error(e);
    } finally {
        // (6) release connection manually
        dbcm.freeConnection(dbc, true);
    }

    return new ZooFormBO();
}
```

- `validate` for checking input

Note: business-rule checking maybe complex, extracting a method is a good choose

```
@Override
protected String validate(DBConnection dbc) {
    // (1) requried, regex, length ....
    if (StringUtils.isBlank(zoo_name)) {
        return "Please input zoo name.";
    }
    if (StringUtils.isBlank(zoo_type)) {
        return "Please select zoo type.";
    }

    // (2) business rule checking
    DBRecord dbr = null;
    try {
        dbr = dbc.getRecordSet(
                "select * from " + dbc.table(sql_table_name) + " where zoo_name = '" + zoo_name + "'");
        if (dbr != null && dbr.nextRecord()) {
            ZooFormBO zoo = new ZooFormBO(dbr);
            if (zoo_id == -1) {
                if (StringUtils.equals(zoo.zoo_name, zoo_name)) {
                    return "zoo name [" + zoo_name + "] is duplicated";
                }
            } else {
                if (StringUtils.equals(zoo.zoo_name, zoo_name) && zoo.zoo_id != zoo_id) {
                    return "zoo name [" + zoo_name + "] is duplicated";
                }
            }
        }
    } catch (SQLException e) {
        AppLogger.error(e);
        return e.getLocalizedMessage();
    }

    // (3) others...

    return null;
}
```

- `putColumns` for html shown

```
@Override
```

```
public void putColumns(Parameters p, boolean its_a_form) {
    p.put(key_field_name, zoo_id, its_a_form);
    p.put("zoo_name", zoo_name, its_a_form);
    p.put("zoo_address", zoo_address, its_a_form);
    p.put("zoo_created", zoo_created, its_a_form);
    p.put("zoo_updated", zoo_updated, its_a_form);
    p.put("zoo_creater", zoo_creater, its_a_form);
    p.put("zoo_updater", zoo_updater, its_a_form);
    p.put("zoo_type", zoo_type, its_a_form);
}
```

Create ZooHTML

- create class `ANHTMLObjects.ZooHTML`

```
public class ZooFormHTML extends HTMLObject {
    // static field for html
    // implement admin, adminChange, adminProcessChange etc.
}
```

- `renderColumns` for render column data

```
@Override
public void renderColumns(BusinessObject item, Parameters p, boolean its_a_form) {
    item.putColumns(p, its_a_form);
}
```

Create html pages

- create `ZooList.html`, `ChangeZoo.html` and `ZooListRow.html`, same as Using AdministrativeTable.
- create `ChooseZoo.html` ,search-form page

```
^DocType^
<html>
<head>
^HTMLHead("Choose Zoo")^
</head>

<body ^bodyoptions^ onLoad="tabFirstField()">

<script type="text/javascript" language="JavaScript">
<!-- (1) form validate -->
<!-- Begin
function checkform(form) {
    if (!emptyField(form.zoo_type))return true;
    if (!emptyField(form.zoo_name))return true;
    alert ("Please supply some filter criteria");
    return false;
}
// End -->
</script>

^PageBegin("choose_street","Choose Street")^
^ErrorMsg("`errormsg`")^
^SearchAddNew("<a href="~adminChange.sdi?oc=`oc`&street_id=-1~">")^

<!-- (2) search form -->
<form action="~adminList.sdi~" METHOD="POST" ENCTYPE="application/x-www-form-urlencoded"
    name="zooformform" onSubmit="return checkform(this)">
<input type="hidden" name="oc" value=`oc`>
<input type="hidden" name="hidemod" value="`hidemod`">
^GroupBegin("search_criteria","Search Criteria")^
    <table cellspacing=0 cellpadding=2 border=0>
        <tr><th align=right>Zoo Name</td>
            <td><input type="text" name="zoo_name" value="`zoo_name`" size=10 maxlength=15></td></tr>
           <tr><th align=right>Zoo Type</td>
            <td> <select name="zoo_type">
                <option value="">zoo-type</option>
```

```
                <option value="SAFARI">safari zoo</option>
                <option value="MARINE">marine zoo</option>
            </select>
            <!-- (3) default value settings -->
            <script type="text/javascript">
                $('select[name="zoo_type"]').val('`zoo_type`');
            </script></td></tr>
    </table>
^GroupEnd^
<tr>
    <td class="groupButtons">
        <input type="image" name="search" alt="Search"
                    src="^akamai^getButton.sdi?user_site_id=`user_site_id`&button=search"border=0>
          ^CancelBackButtonFromBreadcrumb("cancel")^
    </td>
</tr>
</form>
^PageEnd^
</body>
</html>
```