Multi-Table Applications with ProgramManager.cfc

Posted At : November 2, 2010 10:30 AM | Posted By : Steve
Related Categories: com.sebtools

Today is Tuesday, so it must be time to continue figuring out how to take advantage of the **com.sebtools package** to manage CRUD and files in the easiest possible way.

So far, we have **defined our data set with XML** and **gotten that definition into Records.cfc** but we have only been using one table. To be clear, there is no reason we can't continue to do things the we we have been doing them and still handle multiple tables. It is only that we would have to have a separate component for each table.

Not only that, but the goal is to have an "HR" application. It seems to be that it would be convenient to have the model for the application all tied together.

We are going to use ProgramManager.cfc to do that. ProgramManager.cfc is meant to create the model for multi-table applications.

Let's add a table to manage employees. We'll use the full name of the employee as the label field for that table. The "FullName" field won't exist in the database - but will instead by a **DataMgr relation field** (for **concatenation**).

Here will be the code for our new "HR.cfc". We can go ahead and delete the old "Departments.cfc" as we won't be needing it.

```
<cfcomponent displayname="HR" extends="com.sebtools.ProgramManager" output="no">

<cffunction name="xml" access="public" output="yes">
<tables prefix="hr">
   <table entity="Department" />
   <table entity="Employee" labelField="FullName">
      <field name="FirstName" type="text" Length="80" label="First Name" required="true" />
      <field name="LastName" type="text" Length="80" label="Last Name" required="true" />
      <field name="FullName" type="relation">
         <relation
            type="concat"
            field="FirstName,LastName"
            delimiter=" "
         />
      </field>
   </table>
</tables>
</cffunction>

</cfcomponent>
```

We would pass Manager.cfc to the "init" method of the ProgramManager component, just like we would a Records.cfc component. The ProgramManager component will then effectively create Records.cfc components for each table defined in its "xml" method. Note, however, that it doesn't create any files to do this - it is all in memeory.

As a result, the API for Application.HR (at least many of the basic methods):

- Departments
  - getDepartment(DepartmentID): Returns a recordset of a single department
  - getDepartments(): Returns a recordset of multiple departments. Any arguments passed in act as equality filters (so, getDepartments(DepartmentName="Glue") would return departments with a "DepartmentName" of "Glue").
  - removeDepartment(DepartmentID): Deletes a department record.

    - saveDepartment(): Saves a department and returns the DepartmentID of the saved department.
- Employees
    - getEmployee(EmployeeID): Returns a recordset of a single employee
    - getEmployees(): Returns a recordset of multiple employees. Any arguments passed in act as equality filters (so, getEmployees(FullName="Joe Smith") would return employees with a "FullName" of "Joe Smith").
    - removeEmployee(EmployeeID): Deletes a employee record.
    - saveEmployee(): Saves an employee and returns the EmployeeID of the saved employee.

To get all employees with the last name of "Smith", for example, we would use the following code:

```
<cfset qEmployees = Application.HR.Employees.getEmployees(LastName="Smith")>
```

The big advantage of using ProgramManager.cfc here is that it provides a unified model and it automatically handles the Records.cfc components without our having to create the files (though we can if we want to). We could have several tables in our application and still have HR.cfc as our only model CFC (though, as we'll see, most real-world applications will eventually need to create CFCs for at least a few of the tables being managed).

ProgramManager.cfc is part of the **com.sebtools package** which is open source and free for any use.