Peter Joh
Chief Architect
Project Hierarchy
peter@projecthierarchy.org
June 15th, 2014
Doc V0.01

The Future of Hierarchy: The *Programmer's* CMS

Introduction

When a developer (like yourself) is deciding to build a new web-application, he basically has three general options he can go in terms of technology:

1. **Java Platform** - He can pick a platform like Java that is powerful, scalable and flexible, but typically more low-level, requiring more code and complexity than higher-level language platforms.
2. **Ruby on Rails** (or other high-level language / platform combinations) – He can pick a higher-level language that lets him build applications very quickly, but often don't scale as easily because its web framework provides a great deal of complex functionality that is too tightly integrated.
3. **Web CMS** – Or, he can pick an even higher-level way to build a web with a Web Content-Management System (or CMS) like Wordpress, Drupal and Joomla. These CMS offer the quickest route to building web applications with almost no programming, but are often very restrictive if the developer needs to modify the functionality beyond the defined behaviors provided by the CMS's plug-ins. Sometimes, it may require the building of one's own custom plug-ins. In addition, they can be very difficult to scale as users increase, and also are not well suited for medium / large complexity websites.

In the future releases of Hierarchy, we plan to create a web ecosystem that balances all these trade offs, providing the ease of maintainability and the quick development of the CMS with the power, flexibility and scalability of programming in the Java platform. In this document, we'll talk about this future product.

One note though, before you read this article, it is suggested you read the following before hand:

- "The Hierarchy Tutorial" – found in the **Hierarchy Developer-Guide** or in the **Main/Learn** section of the projecthierarchy.org website
- "N-Dimensional Architecture Overview" – found on the projecthierarchy.org website in the **Community/Articles** section.

We assume that you've already read these, and the ideas we are going to introduce build directly off ideas explained in these docs.

Using Hierarchy as a CMS:

As mentioned in the "N-Dimensional Architecture Overview" doc, by using matrices, Frictionless Persistence and NDA, developers can build web applications that are easy to maintain and enhance through Developer-Task Automation. To expand on this concept, we can take automation even further by providing actual user-interfaces to perform these changes, turning your system into a CMS. Now, non-technical users can maintain the site through these admin interfaces since most changes don’t require having to write any code. To see what we mean, if we briefly review the code example of the unconventionalthinking.com website introduced in the “N-Dimensional Architecture Overview” doc:

```
MATRIX Unconventional.Content USES (com.unconventional.matrix.j2ee::Web.Content)  {

    `Home` {
        PAGE.INFO: { "home", 0, -1, "Home",
          "We build good software. We develop software products to help improve our world. We truly enjoy developing
             software that enhances your quality of life.",
                              "","", false };
        NEWS {

            NEWS.STORY: {"Hierarchy beta release",
                "The Hierarchy beta is ready for you to try!",
                "June 6<sup>th</sup>, 2011",
                 "After years of development, the beta version of the Hierarchy Meta-Compiler for Java is ready for
                   developers to try out!."
            };

        }
    }

   `Products`  {
        PAGE.INFO: { "products", 1, -1, "Products",
          "We truly enjoy creating good software. Many applications are slow, crash or have poorly designed features.
             Bad applications make users sad. Building software that is powerful and easy to use, and reliable and
             responsive takes a lot of hard work, but we feel the greatest satisfaction in seeing our users happy. ",
          "","", false};

        NEWS {
            NEWS.STORY: {
                "Matrix", "Hierarchy for Matrix-Programming ", null,
                "We are excited to finally let users try what we've been working on for the past 4 years: " +
                "the Hierarchy Beta is ready for Java developers to try out!..."
            };
        }
    }

    .
    .
    .

}
```

This is the main matrix used by the system to generate the pages of the site (in this case the pages are the **Home** page and the **Products** page). The code that uses this matrix to do this generation of the website looks like this (this is an enhanced JSP and is taken again from the "N-Dimensional Architecture Overview" doc):

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<%@page import="MATRIX com.unconventional.web::Unconventional.Content"%>


<%
   //  This next method call looks at the request objects parameters and determines which page we're on. Then,
   //  it grabs the corresponding descriptor for the page from the Unconventional.Content matrix
   DESCRIPTOR<Unconventional.Content->ITEM> curr_PageItemDesc = determineCurrPageItemDescriptor(request);
%>

<table cellpadding=0 cellspacing=0 border=0 bordercolor=black width=532>
<tr>
    <td ><div style="width:500px">

        <div style="margin-top:17px">
            <img src="images/blurb_<%=currPageInfo_Desc:>Name_NoSpaces%>.gif"
                alt="<%=escapeHTML(currPageInfo_Desc:>BlurbText)%>"
                style="margin-top:3px" border="0">
        </div>

        <% if (curr_PageItemDesc.label == `Home`) { %>
            <div style="margin-top:47px; ">
                <img src="images/subsubtitle_news.gif" alt="News"><br>
                <div style="margin-top:25px">

                    <%
                     for (DESCRIPTOR<Unconventional.Content->ITEM->NEWS->NEWS.STORY> newsItemDesc :
                           curr_PageItemDesc->NEWS->NEWS.STORY{*}) {
                    %>
                       <%=newsItemDesc:>Title%><br>
                       <%=newsItemDesc:>DateText%><br>
                       <%=newsItemDesc:>Story_Blurb%><br>
                    <% } %>

                </div>
            </div>
        <% } else if (curr_PageItemDesc.label == `Products`) { %>
            <div style="margin-top:62px">
                <div style="margin-top:12px">

                    <%
                     for (DESCRIPTOR<com.unconventional.web::Unconventional.Content->ITEM->NEWS->NEWS.STORY>
                        newsItemDesc : curr_PageItemDesc->NEWS->NEWS.STORY{*}) {
                    %>
                        <%=newsItemDesc:>Title%><br>
                        <%=newsItemDesc:>Story_Blurb%><br>
                    <% } %>

                </div>

            </div>

        <% } else if … %>
            .
            .
            .
```

To see what the results of running this code would like, see the home page and the product page for the unconventionalthinking website at http://www.unconventionalthinking.com. These two code snippets were taken from the actual site. Now, in this site, we are creating an NDA architecture. What makes this code NDA is two main characteristics:

1. The content and the settings for the site are collected together.
2. The system architecture has been automated as most of the site is generated off of this settings matrices.

The result is a site that has automated the developer's maintenance tasks and is called **Developer-Task Automation**.

All these concepts were previously explained in the "N-Dimensional Architecture Overview" doc. But now, let's take these ideas further. One of the main concepts of NDA and Developer-Task Automation is that *most* of the changes a developer makes should *mostly* occur in the matrices plus *a little* extra coding. What if we required that *all* the changes needed to perform a development task occur in the matrices with *no* extra coding? When you've reached this level of automation, that means that now, non-technical people can be taught how the matrices of the site work and then perform changes themselves.
Developers are no longer required to administer the site except for major changes.

And, if Frictionless Persistence is turned on for these matrices (meaning changes can be made to them in real-time and then persisted to disk), this means maintenance task can actually be perfumed while the site is actually running. Now, any dynamic changes to the matrices will be saved to disk, and the running code will generate its content from these modified matrices.

On top of this, we can provide a GUI to allow the non-technical people to do these actual modifications. In future releases, Hierarchy will have an Excel-like application that can connect to any matrix allowing administrators to view and edit their contents. This Excel-like application would be able to connect to the matrices both statically (when working with the actual matrix code), and also dynamically (while the system is actually running). Using this application, administrators can now go into the site and make major changes to the system in real-time without having to recompile and restart the system. The site is now working as a CMS, similar to Drupal or even more closely, OpenCMS[1].

Moreover, this Excel-like application could understand permissions set on the matrix data, so that different administrator-roles would be limited to editing only certain parts of the matrices. And, on top of this, this application could also provide the quick and easy forms that Excel provides to build an admin interface specific to our site (If you're unfamiliar with Excel forms, these are pages that be created in Excel for editing data. These forms provide input boxes, radio buttons and other controls and are very similar to HTML forms, but possibly even easier to create). By adding in forms to Hierarchy's editor, an entire admin-interface for a decent-sized website could be built in a couple days instead of weeks. It could *perhaps* even be created by non-technical personnel (if this form-builder tool is created powerfully enough so that no coding is required for most tasks).

To sum up the features of our future Hierarchy-CMS: it would automate the maintenance tasks typically performed by developers, and with relatively little extra dev-work, provide a GUI to administer the site.

But, is a Hierarchy CMS scalable?

Unlike most CMS like Drupal, Joomla, OpenCMS… A Hierarchy CMS would be highly scalable. The reason is Hierarchy does not use complex components that are extremely specific in both their functionality and their performance. For instance, Drupal uses sophisticated plug-ins to provide behavior like banners and user comments. This are fully featured, but are black boxes with little flexibility in terms of performance and extendibility.

Hierarchy, on the other hand, uses a lot of *well-chosen* simpler elements and concepts to build a website: Matrices, Frictionless Persistence, NDA, and (in the future), an Excel-like matrix editor. Almost all these elements are lower-level, are lightweight, and whose performance characteristics are easy to understand. Hierarchy's matrices were designed with performance, scalability and extendibility in mind, and should be worked with like working on the level of a Java collection (not quite, they are obviously heavier than the collection objects, but are still quite lightweight).

In fact, Hierarchy CMS will probably have more sophisticated web-components such as banner objects and such, but we will always keep the mantra of trying to create simpler, *well thought-out* tools and features that balance ease of use with flexibility and scalability.

[1] OpenCMS (http://www.opencms.org) was a major inspiration for using matrices as the backend for a web CMS (OpenCMS uses XML files where is Hierarchy uses Matrices). *But*, one major difference is OpenCMS is designed around having one XML file per web page (you're not limited to this, but it is the main idea of how it works). This is great for simplicity, but can result in enormous redundancy of content (for example, displaying recent events for the company on multiple pages must be added to all the XML files for each page). It's really designed for small/medium static websites. Hierarchy is different in that using Hierarchy as a CMS is more similar to a repository-style CMS where all the content is generated more flexibly from this repository.