Creating Multi-Tenanted ToolTwist Sites

What does “multi-tenanted” mean?

A multi-tenanted site is one where multiple users come to the site, but see different versions of the website, usually as a result of accessing the site with different URLs.

Examples of multi-tenanted sites include:

- Sites that support multiple languages, providing a different variant for each location.

- Hosted online stores, where multiple online stores are running on the same web server.

In each case, what appears to be multiple websites, may in fact just be a single website that changes it’s content, site map, and appearance depending upon which URL is being used.

Site Maps

A common restriction of multi-tenanted sites is that the sites may have different color schemes, logos, background images, etc, but have the same site map (i.e. the same pages and the same links between pages). While this simplifies the task of creating multiple sites, it is restrictive because it assumes that “one size fits all”.

In ToolTwist we don’t have this restriction. Different sites, even those with different functionality, are usually targeted at different markets, so the content of each site, and hence the site maps, may need to be different for each site.

In ToolTwist we handle this using *Navpoints*. Navpoints provide a hierarchy of web pages, and are used to define the links between pages. These form the site map for the site. On multi-tenanted sites, multiple hierarchies of navpoints can be defined, where each site effectively has it’s own site map.

In ToolTwist, the actual web pages themselves are designed independently from the navpoints that are used to include them in a site. Each navpoint refers to a specific page layout, however multiple navpoints, or even multiple sites, may use the same page layout. This removes the need to clone pages dozens of hundreds of times.

Multi-language sites
ToolTwist supports multi-language site building in two ways. The first is standard internationalization and localization provided by Java, commonly known as i18n. This allows

formatting of dates, times, and other region specific information. It also allows labels displayed on pages to be substituted with translations into various languages. (See http://en.wikipedia.org/wiki/Internationalization_and_localization).

This approach goes a long way towards allowing a website to work across regions, and may work well across a spread of western and European countries, but rarely results in an optimal solution for all locations:

1. Whereas one languages might use a very long word or phrase for a particular meaning, another might use a very short word or phrase in the same context. Designing a web page that is generic enough to be able to handle these variable lengths will often result in a page that is either very sparse, or else has poor alignment.

2. Right-to-left or top-to-bottom text do not transpose well to generic page designs. The relative positions of headings, text, menus, and images do not remain the same.

3. In different regions people often have different expectations for the navigation of the site. This will affect the ways menus and the pages are linked.

4. In some languages, and in particular the Chinese language, a single character has the meaning of an entire word in other languages. As a result these languages can contain a far greater amount of information on a single page. A typical example is a newspaper - a Chinese newspaper can contain the same information as a western newspaper whilst having far fewer pages. As a result, Chinese readers expect a much denser layout than readers of Western newspapers. In the following example, the front page of the Chinese paper has eight stories, whereas the front page of the English paper has four.

Websites - the layout of web pages, and the sitemap of the site - need to be compatible with the expectations of readers. A generic layout may be *intelligible* across regions, but is unlikely to be *desirable* across all regions. In order to provide a website that is appealing in each region, multiple websites have to be created whenever straight text translations aren't satisfactory. ToolTwist allows this to be done in several ways.

Alternate page hierarchies

The navpoints mentioned in the previous section can be used to provide multiple site maps and sets of page layouts. The same 'widgets' can be used on all these pages, with the same standard and custom functionality, but can be placed on different pages in different locations. When a user comes to the site, they will automatically be directed to the relevant navpoints for their language or region, based upon the URL they use or other criteria.

Customized widgets

From version 8.2 of ToolTwist, custom widgets can have multiple 'themes'. In effect, the templates used to generate the production code can be switched in and out, resulting in the same widget having different appearance, or subtle changes in functionality, when it is use to generate the code for different pages.

Whether localizations are being made to creating alternate navpoint hierarchies, or by changing the templates used by widgets, it is common that the changes can be done without involving programmers, allowing staff experienced in the subtleties of each region to be involved in the internationalization process. This streamlines the process and results in a more suitable websites being created.

Separation of Roles

In most multi-tenanted applications it's necessary to keep the individual sites clearly separated from each other, both at run time and also while the site is being designed or maintained. The people looking after each site will often not even know what other users are building sites within the same environment.

For example, it's critically important that the shopping cart on one eCommerce site does not get intermixed with that of another online store. Similarly, customers, products, and other information in each site must be walled off from the other sites.

To do this, the context under which the user access the site must be determined throughout the operation of each site, from the time the request comes from the browser, through all

subsequent operations.

Runtime Considerations

1. As a request comes in to the server, the context of the user must be determined, usually from the domain name in the URL. The *context* indicates which site is being accessed. In ToolTwist this is done using a *Interceptor* plugin. This context can be stored in the ToolTwist *WbdSession* object, from where it is accessible from all parts of the application.

2. All software within the application needs to be written to respect the user context. Typically this means adding filters to database accesses to ensure that the application only selects database records relevant to the specific site. Similarly all writes to the database should include the context / site identifier so subsequent selects can be filtered correctly. In ToolTwist, these database functionality can be handled by adding a single parameter to the select criteria. In the case of ToolTwist's XDS services, the identifier can be inserted automatically.

3. Many applications require back-end interfaces, for example to payment gateways, or an order provisioning system. A plugin architecture can be implemented within ToolTwist using XDS services to allowing routing queries and updates to the correct backend, in a way that reduces clutter and complexity in the application code. In most cases new interfaces can then be added in the future without "butchering the existing code".

ToolTwist Designer

To keep separation whilst designing sites, it is possible to restrict the navpoints that can be viewed and edited in the ToolTwist Designer. This makes it impossible for the maintainer of one site to alter or corrupt the content of another site.

A plugin approach is used to restrict what can be viewed. In some environments there may be shared resources (images, pages, etc) across sites. In others the separation should be absolute, as entirely different organizations own the different sites, but are sharing the development environment.

These plugins can be built on a case by case basis, and attached to external sign on or security systems if required.

ToolTwist Controller

The ToolTwist Controller is the deployment tool used to generate ToolTwist websites and deploy them to production servers. Within the Controller configuration it is possible to specify which servers should be used to support each sub-site. At one extreme you can have a single web

server supports all your sites. At the other extreme you can have a different web server for each site. In most cases this is entirely configuration driven, and can be reconfigured at any time without code changes.

The Controller also deploys across replicated, load balanced and redundant servers, allowing you scale as required to meet user load. Version control of the generation process supports controlled deployment from development, through testing and staging environments to production, and allows quick recovery or rollback in the event of production errors.

A typical scenario is for the sites for multiple languages will be installed on a group of servers located within the same region as the users who will be accessing the site. For example, using the Amazon AWS servers in Australia, Singapore, United States, South America, Africa and Europe. Each server location supports all of the countries in it’s region, load balanced across multiple servers, to provide scalability and fault tolerance. ToolTwist multi-tenanting is then used to provide separation of the individual sites, even though they are running on the same physical servers.

-oOo-