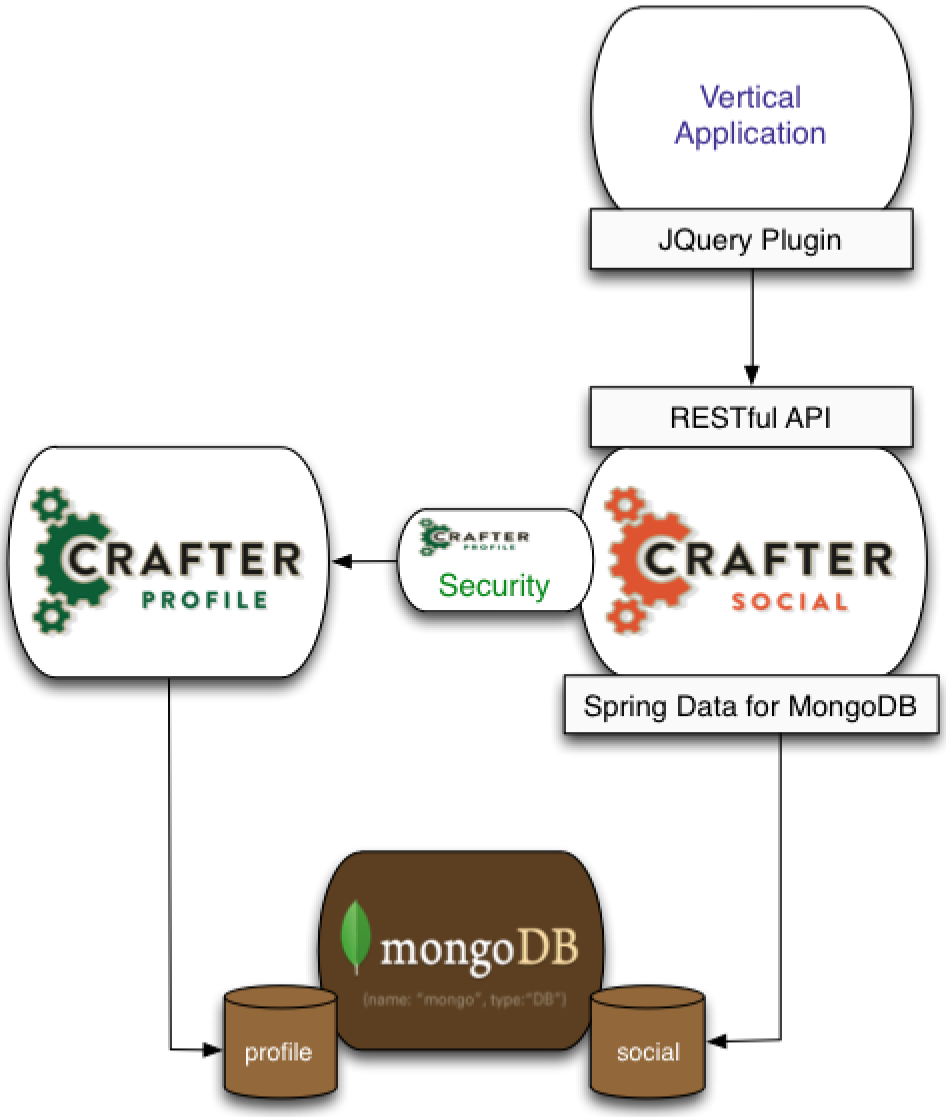
Crafter Social Design

Contents

Overview

Crafter Social is a headless RESTful application. It uses MongoDB for persisting the user-generated content and Crafter Profile for authentication. This document details the architecture and features of the application.



Glossary

UGC	User-generated content. A term to refer to content that users created, for example ratings, reviews, threaded comments.

Technologies

Headless RESTful Application

Crafter Social is a headless RESTful application and so, the vertical applications using Crafter Social can be of any technology type. Crafter Social comes bundled with a JQuery plugin that can be used to speak RESTfully to Crafter Social. As an example, a products site could integrate with the JQuery plugin to speak RESTfully to Crafter Social, but it is not limited to using this. The exposed RESTful API supports CRUD operations on the user generated content (UGC), moderation, permissions and GridFS attachment capabilities. The complete RESTful API can be found documented here.

RESTful API

Crafter Social API	RESTful API guide for Crafter Social

MongoDB

Crafter Social server uses MongoDB as its persistence layer. MongoDB provides document-oriented storage that supports the content that is relevant to Crafter Social. MongoDB also features replication, auto-sharding and GridFS, to support high-availability and scalability.   Crafter Social speaks to MongoDB using Spring Data for MongoDB.

Crafter Profile

Crafter Profile is used for the management of profile, tenant and roles and authentication. More details of Crafter Profile can be found at the link provided below.

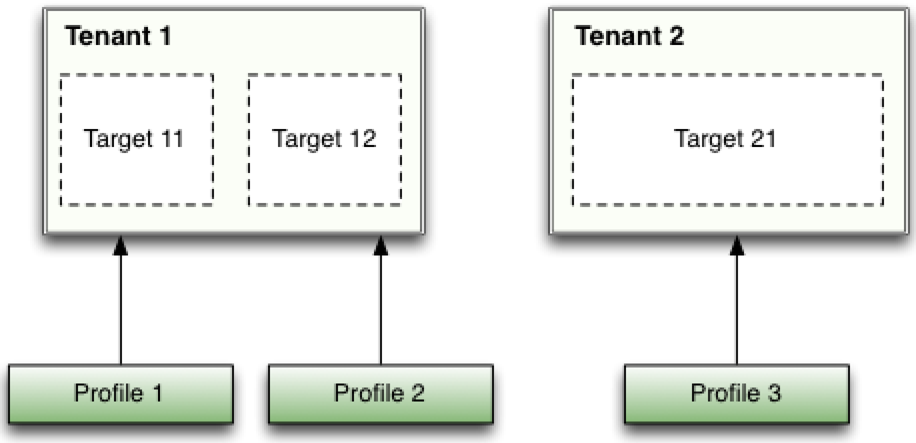
Crafter Profile	Crafter Profile wiki

Design

This section describes some of the design features of Crafter Social.

Multi-tenancy

Crafter Social server is a multi-tenant application. A tenant can be thought of a directory or silo where content is specific & relevant to only that tenant. Furthermore, content will only be accessible by profiles that are members of that tenant. A tenant could be a website, part of a website or a client you are supporting with your Crafter Social infrastructure.



Every request to Crafter Social is tenant specific. Through Crafter Profile Security, Crafter Social secures content for a particular tenant by authorizing only requests from the configured domain(s) list for that tenant. In addition, the user making the request must also be a member of the specified tenant. A tenant's domain(s) are configured in Crafter Profile and can be managed through the Crafter Profile Admin Console.

Security

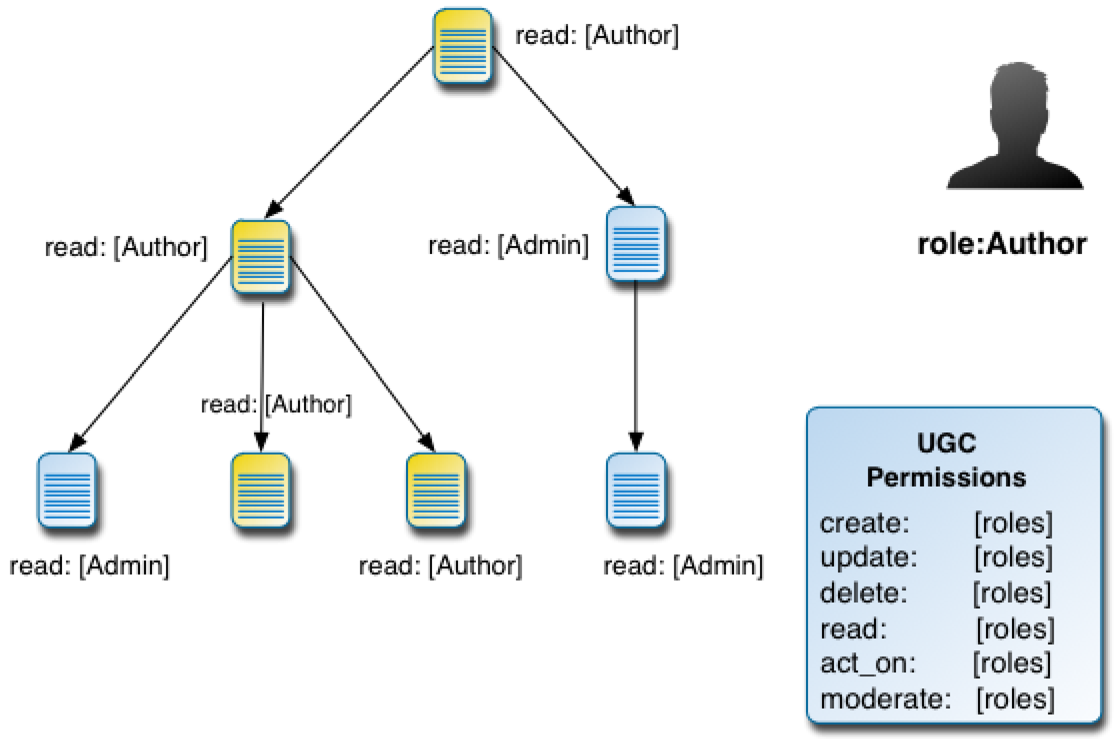
Authorization

Crafter Social supports fine-grained permissioning of the user-generated content.

At the time of creation, every user is assigned a role or roles, such as Admin, Author or Moderator. After the user logs in, their roles are used to determine what content they can read, what content they can edit, where they can create content and so on.

Each UGC knows what roles can perform each action on it. So for example it knows what roles can delete or read it, or what roles can create child UGCs that hang off of it.

When a UGC is created, specific roles can be set then. If no roles are set, it inherits the parents roles, or if it is a root, the default roles are given to it. So if a user logs in with an Author role and navigates to a particular forum topic that it has access to, it will only see the threaded comments that an Author can see.



Salted encrypted cookie

To improve performance, Crafter Social uses a salted-encrypted cookie to store details of the user's profile. This details are stored as a secured cookie and used to authorize and authenticate the user. The profile details are refreshed periodically, triggered by an expiration date in the cookie.

Scalability

Scalability is achieved through a number of avenues.

Headless RESTful Applications

Both Crafter Social & Crafter Profile are headless RESTful applications and so each can be scaled out separately. Furthermore, each application has their own MongoDB database and so can each have their own instance of MongoDB.

Replication

MongoDB can be configured to replicate its instances. Replication allows for performance improvements whereby all writes are pushed to a primary member, which would then be asynchronously distributed to secondary members. All reads would be distributed between all of the members. Replication also enables the high-availability of the applications through failover instances and the geographical distribution of the content.

Auto-sharding

With MongoDB, you can add a new machine, select a shard key and configure MongoDB to auto shard. Depending on how you are using Crafter Social your shard key may change. For the UGC collection, if you have a lot of targets and the content is pretty evenly distributed across the targets, the target ID could allow for an even distribution of content.

GridFS

For UGC systems, attachments, like videos and images, can be uploaded with the content. Attachments can vary in size and can easily be big, for example a video that is uploaded as part of a blog. MongoDB's document size limit is 16MB. GridFS is a convention in MongoDB for storing files of arbitrary size. In Crafter Social, to allow for the scalability of attachments, we make use of Spring Data's GridFS implementation to store the attachments.

JQuery Plugin

Crafter Social bundles with a JQuery plugin. The JQuery plugins can, if desired, be used to speak to Crafter Social server RESTfully.

This example shows how the JQuery plugin can be invoked.

```
<script type="text/javascript">
  jQuery(document).ready(function(){
   var configData = {};
   var configManager = {
        getConfig: function (token) {
            var url = 'get_config.json';
            var data = {};
            return $.ajax({
            url: url,
            data: data,
            dataType : 'json',
            contentTypeString:"application/json;charset=UTF-8",
            cache: false,
            type: 'GET',
            success: function(aData, textStatus, jqXHR){
             configData = aData;
            },
            error: function(jqXHR, textStatus, errorThrown) {
             $.error('Could not load tenant: ' + textStatus + ' - ' + errorThrown);
            }
        });
        }

    };
   configManager.getConfig("history").done(function(data){
     jQuery('#ugc_div').ugc_blog_console({
     clientId : 'CrafterCMS',
     restUrl : '/crafter-social/api/2',
     resourceUrl : '/crafter-social/resources',
     target : data.blogListForm.target,
     tenant : data.blogListForm.tenant,
     actions: data.blogListForm.actions,
     ticket  : data.blogListForm.ticket
     }).ugc_blog_console('loadUGCBlogConsole');
   });

  });
</script>
```