Puzzlehunt Server Documentation

Release v4.1.0

Dillon Lareau

Aug 17, 2021

Contents

If you’re here as a user, start with the “How to Create a Hunt” and “How to Run a Hunt” sections. If you are here as a developer, start with the “Setup” and “Basics” sections.

Chapter 1

How to Create a Hunt

Table of Contents

This is a guide describing how to set up a new puzzlehunt on the Puzzle Hunt CMU server. This guide assumes that the server itself is already set up and that you already have an account on the server with both "Staff" and "Superuser" permissions.

1.1 Prepare the Hunt Content

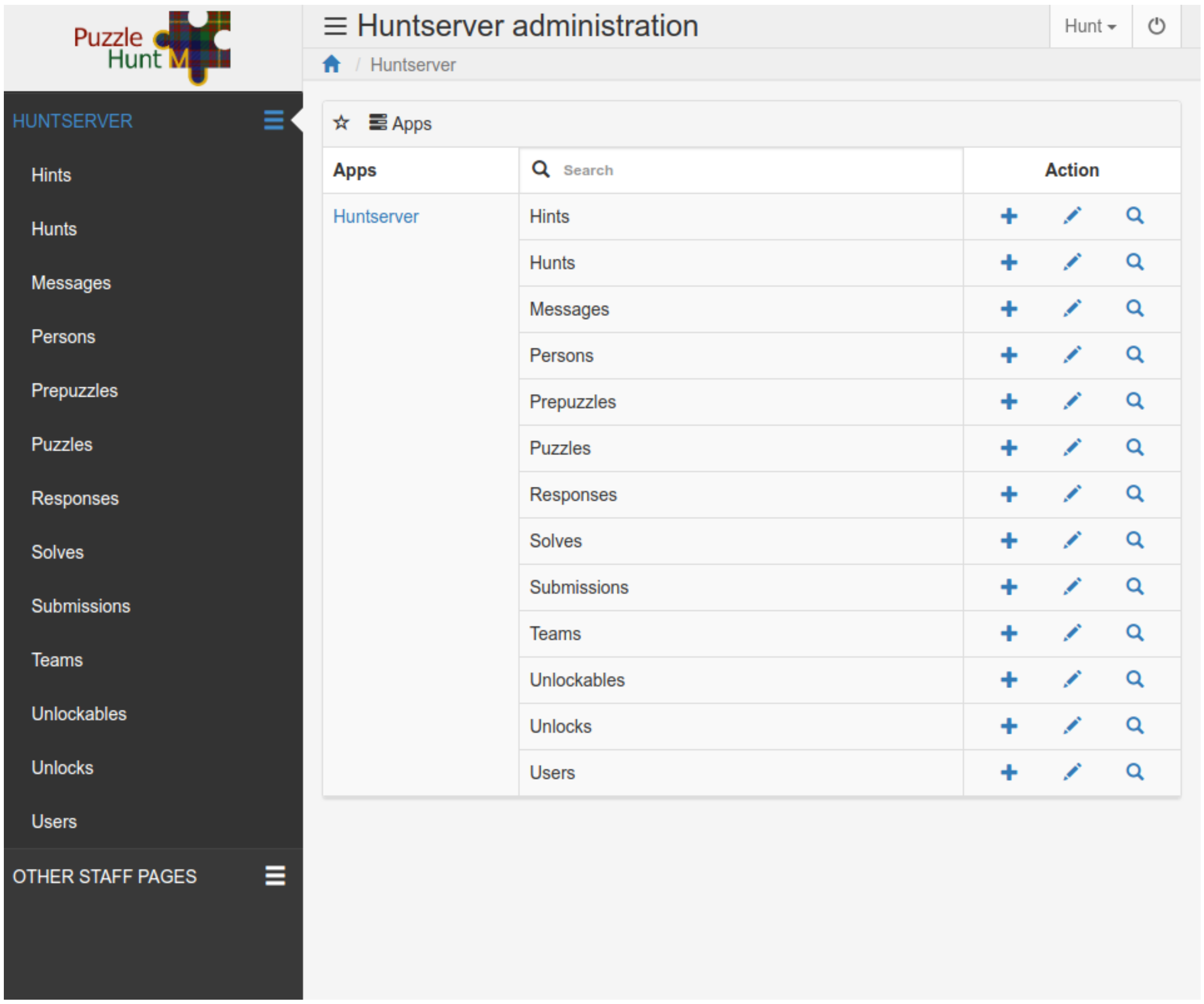
Before anything is done on the server you should decide on some basic details about the hunt. You can always come back and edit these details later if they change. Some things to think about:

- Hunt Name
- Hunt Date
- Hunt duration
- Hunt starting location
- Max team size
- Number of puzzles in the hunt
- Unlocking structure for the puzzles in the hunt

Those first 5 details are especially important because they will be visible on the front page as soon as you establish this hunt as the current hunt.

1.2 Create the Hunt Object

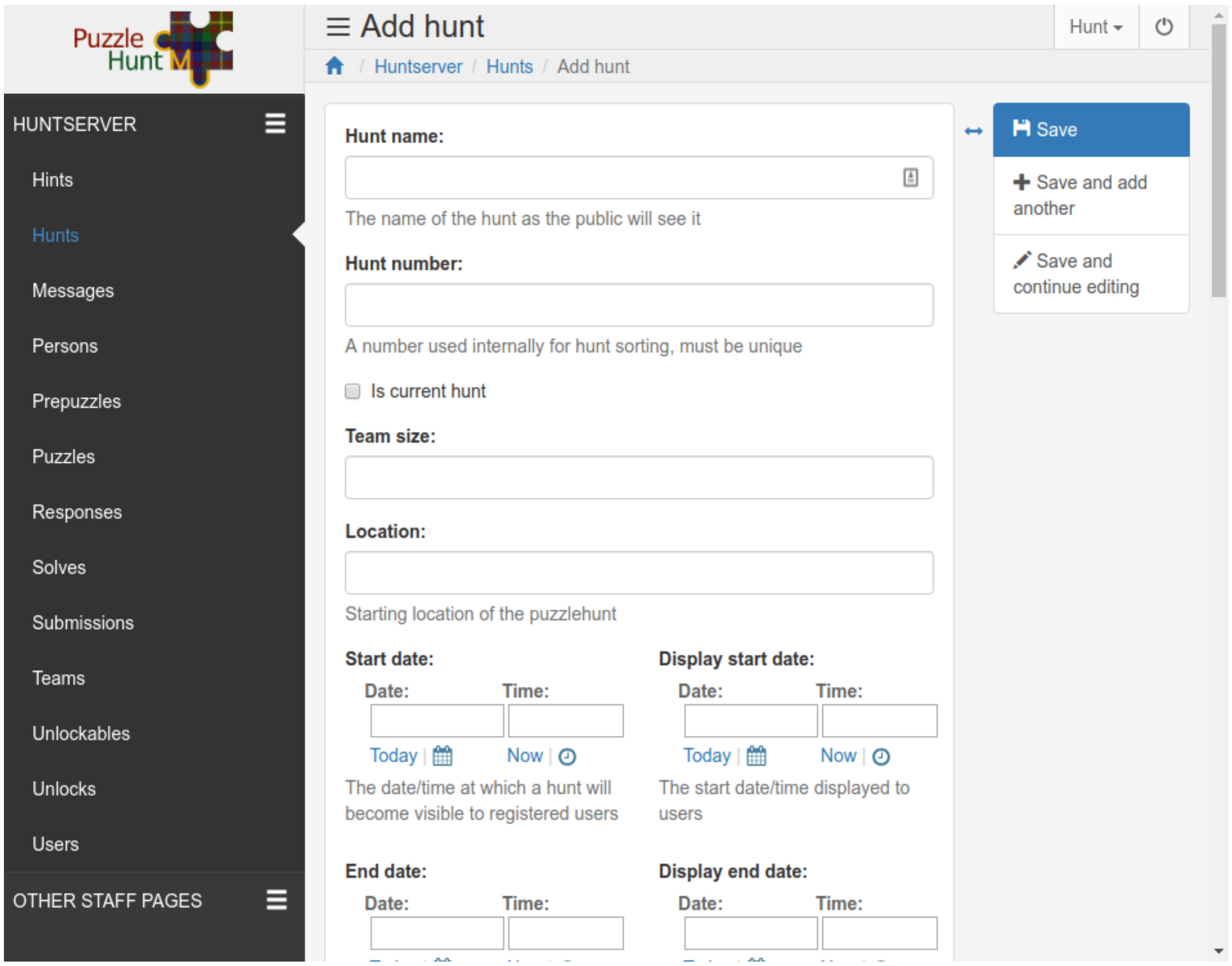
You aren't going to get very far without a hunt object to attach all of this data to, so sign in with your staff account and navigate over to `{server URL}/staff`. You should be greeted with a page like the one below:



Apps	Search	Action
Huntserver	Hints	+ ✎ 🔍
	Hunts	+ ✎ 🔍
	Messages	+ ✎ 🔍
	Persons	+ ✎ 🔍
	Prepuzzles	+ ✎ 🔍
	Puzzles	+ ✎ 🔍
	Responses	+ ✎ 🔍
	Solves	+ ✎ 🔍
	Submissions	+ ✎ 🔍
	Teams	+ ✎ 🔍
	Unlockables	+ ✎ 🔍
	Unlocks	+ ✎ 🔍
	Users	+ ✎ 🔍

Click on the "Hunts" label either in the center or on the left-hand sidebar.

Once on the hunts page, click the blue "+" button in the upper right-hand corner to create a new hunt object. The page should now look like the below image:



Hunt name:

The name of the hunt as the public will see it

Hunt number:

A number used internally for hunt sorting, must be unique

Is current hunt

Team size:

Location:

Starting location of the puzzlehunt

Start date: Date: Time: Today | Now |

The date/time at which a hunt will become visible to registered users

Display start date: Date: Time: Today | Now |

The start date/time displayed to users

End date: Date: Time:

Display end date: Date: Time:

Start by filling out everything in the "Basic Info" section.

Caution: Checking the "Is current hunt" box will make this hunt the hunt visible on the front page of the website. Only do so if all of the public facing details (everything in the "Basic Info" section) are correct.

Important: There are two start and end dates.

"Start Date" and "End date" are for internal use and will control things like when the puzzles become available to the players, when teams start gaining points for point based hunts, and when the hunt will automatically stop accepting answers.

"Display start date" and "Display end date" are the dates/times displayed on the front page of the website and control nothing.

In general, set the display dates for when people should arrive and leave and set the actual dates for when teams should be actively solving puzzles.

Next, fill in the two fields in the "Hunt Behavior" section, the help texts should be pretty self explanatory.

The next section covers the "Resources/Template" section. If you don't want to deal with making the hunt template right now just type anything you want in the template field and then skip to "Hint Unlock Plans" below.

1.2.1 Editing the Hunt Template

This is where we give the hunt its look and feel. Before this point, navigating to the hunt page would just give you a blank page.

Basic Information

Everything typed into the "Template" form on the hunt editing page will be run through Django's templating engine and rendered as HTML on the hunt main page.

You can find documentation about Django's template language here: https://docs.djangoproject.com/en/2.2/ref/templates/language/. I'd recommend reading the "Variables", "Filters", and "Tags" sections.

Speaking of variables, the following variables will be passed to the renderer for use in the template:

hunt The current hunt object

team The team object for the team of the user currently visiting the page

puzzles A list of puzzle objects that the team currently has unlocked, sorted by puzzle number

solved A list of puzzle objects that the team currently has solved, unsorted

Tip: You can view the fields that are available to access on each of the team, hunt and puzzle objects in the *models documentation*.

Since version 3.0, in order to reduce repository clutter, it is now against policy to commit files specific to a certain hunt to the repository. This means that you are no longer allowed to load resource files directly onto the server.

To still allow the use of new static files in each hunt, there is now a field on each hunt's admin page for a resource URL. This URL should point to a publicly accessible zip file, which contains all static media needed for the main hunt page. The resources can be downloaded by clicking the "Resources" button next to the appropriate hunt on the Hunt Management page. After the resources have been downloaded, they will be accessible through the use of a special template tag.

The `{% hunt static %}` template tag will insert the URL to the current hunt's resource directory. For example, putting the text `{% hunt static %}myimage.png` in the template would insert the URL to the file `myimage.png`.

Inheriting the Base Template

It is recommended to start your template out with the following code:

```
{% extends "hunt_base.html" %}

{% block content %}
  Your content here
{% endblock content %}
```

The above code inherits the hunt_base.html template, which in turns inherits the base.html template. You don't need to know the contents of those two files, just that they provide the basic functionality like the site header and they define the following blocks that you can override for additional custom behavior:

{% block title %} This block controls what title is in the web browser tab. The default value for this block is "Puzzlehunt!"

{% block base_includes %} This block controls what content will be sourced/included before the standard Bootstrap and Jquery imports. This allows you to override unwanted bootstrap styles. The default value for this block only imports hunt_base.css.

{% block includes %} This block controls what content will be sourced/included after the standard Bootstrap and Jquery imports. This is for content that you want to use to extend those libraries, or content that relies on those libraries.

{% block footer %} This block controls what content will be inserted at the bottom of the page. The default value is links to our social media and bridge page.

You can read more about Django template inheritance and blocks here: https://docs.djangoproject.com/en/2.2/ref/templates/language/#template-inheritance

Warning: While you may use completely custom HTML, it is STRONGLY RECOMMENDED that you follow the instructions below on how to inherit the base template to get nice features like the header bar, common style sheets, Google analytics, and graceful degradation when the hunt becomes public.

Starter Example

While you may now technically have all of the information you need, that doesn't mean you know what to do with it. Below is a simple example based one of our first hunts to use this server. It will show the puzzles, display the answer for any solved puzzles, and demonstrates how to break a hunt into two rounds.

```
{% extends "hunt_base.html" %}
{% block title %}Puzzles!{% endblock title %}

{% block base_includes %}
<link rel="stylesheet" type="text/css" href="{{ STATIC_URL }}huntserver/hunt_base.css
↪">
<style>
.puzzle-name {
  white-space: nowrap;
  overflow: hidden;
  width: 320px;
}
</style>
{% endblock base_includes %}

{% block content %}
<div class="container" >
  <div class="row" >
    <div class="content col-md-6 col-md-offset-3" id='puzzle-frame'>
      <h1 class="title">Puzzlehunt: The Musical</h1>
      <div id="puzzles">
        <table>
          <thead>
            <tr>
              <th style='width: 320px'>Puzzle Name</th>
              <th style='width: 180px'>Solution?</th>
            </tr>
          </thead>
          <tbody>
            {% for puzzle in puzzles %}
              {% if puzzle.puzzle_number == 8 %}
```

(continues on next page)

(continued from previous page)

```
                </tbody>
                </table>
                <h3 class="title">- Intermission -</h3>
                <table>
                  <tbody>
                  <col width="320px">
                  <col width="180px">
              {% endif %}
              <tr id='puzzle{{ puzzle.puzzle_number }}' class='puzzle'>
                <td>
                  <p class="puzzle-name">
                    <a href='/puzzle/{{ puzzle.puzzle_id }}/'>
                      {{puzzle.puzzle_name}}
                    </a>
                  </p>
                </td>
                <td>
                  {% if puzzle in solved %}
                    {{ puzzle.answer|upper }}
                  {% endif %}
                </td>
              </tr>
            {% endfor %}
          </tbody>
        </table>
      </div>
      <p> Feeling stuck? <a href="/chat/">Chat</a> with us</p>
    </div>
  </div>
</div>
{% endblock content %}
```

Template Wrap Up

That should be enough to get you started with template writing. Don't forget to download resources each time you update them and save often when editing the template as it won't save if you close or leave the page for any reason.

Tip: You can use ctrl-s/cmd-s to save the page and continue working

1.2.2 Hint Unlock Plans

The final section of the Hunt creation page is for determining if and when hints will automatically become available to teams. If you do not want to use automatic hints (or hints at all) in the current hunt, simply ignore this section. Manual hints can still be awarded from the "Hints" page under the "Other Staff Pages" sidebar header.

If you do want to automatically award hints during the hunt, there are three possible unlock mechanisms for hints:

Exact Time Unlock: All teams will gain a single hint some amount of time into the hunt. Use the unlock parameter field to indicate how many minutes into the hunt this hint should be given out.

Interval Based Unlock: All teams will gain a hint every X minutes for the entire duration of the hunt. Use the unlock parameter field to indicate the number of minutes between hints. The first hint will be given out X minutes after the start of the hunt.

Solves Based Unlock: Each team will individually be given a hint when they reach a certain number of puzzle solves. Use the unlock parameter field to indicate how many solves a team needs to unlock this hint.

You may add as many hint unlock plans as you want, using the "Add another Hint unlock plan" link at the bottom to add additional rows to the table. All hint plans will trigger independently of each other.

Caution: "Exact Time Unlock" and "Interval Based Unlock" hints are both calculated against the "Start Date" field of the hunt, making it even more important that the start date is actually when teams will start solving puzzles and not just when teams arrive for check in.

Danger: Changing a hint unlock plan after the hunt has started can have unexpected results. Please take extra care to make sure that the hint plans are correct before the hunt starts.

1.2.3 Hunt object creation wrap up

After you've filled in everything make sure "Is current hunt" box is appropriately checked or unchecked and hit the blue "Save" button in the upper right.

1.3 Create Puzzle Objects

Great, now we have a hunt template and we can view our hunt, but that's not good without any puzzles, so let's add some.

Start by going to the "Puzzles" section using the side navbar and clicking the blue "+" button in the upper right-hand corner to be brought to the puzzle creation page.

1.3.1 The Basics

Start by choosing which hunt the puzzle will belong to and giving the puzzle a name and an answer.

Tip: Answers are not case sensitive

Next, the puzzle must be given both a number and an ID. The number is for ordering within the hunt, and controls the order of puzzle objects passed into the hunt template. The ID used as a unique identifier across all puzzles is used in the URL for the puzzle.

Note: The current trend for ID's is to have the same 3 digit prefix for all puzzles in a hunt and to use the puzzle's number as the last 2 digits. This allows easy visual grouping of puzzles by hunt, and an ordering over all puzzles.

1.3.2 Attributes

Next are three True/False puzzle properties, all of which default to False:

"Is a metapuzzle" Controls which puzzles are marked as metapuzzles for the purpose of scoring on the progress page.

"Doesn't Count" Controls whether or not the puzzle is discounted from scoring on the progress page.

"Is HTML puzzle" Controls whether the puzzle is more than just a PDF. If this box is checked, the puzzle page will not display a PDF, and instead display a link to the HTML content from the "Resource link" discussed below.

1.3.3 Content

Puzzle content is controlled by the following three links:

"Link" The link to a publicly accessible PDF of the puzzle (if the puzzle is not an HTML puzzle).

"Resource link" The link to a publicly accessible ZIP file of the puzzle contents if the puzzle is an HTML puzzle. The ZIP file must contain a file named "index.html". All links from the index file to other files in the ZIP file should be relative links, as the base URL of the final contents is not guaranteed.

"Solution link" The link to a publicly accessible PDF of the puzzle solution. If this field is filled in, the solutions for each puzzle will be available on the puzzle page after the hunt is over.

Tip: Linking an unzipped Dropbox folder for the resource link will also work. Dropbox will automatically generate a zip file of the folder upon download.

1.3.4 Unlocking the Puzzle

Next is the matter of how the puzzle is unlocked. As of version 4.0, there are now four options for puzzle unlocking:

Solves Based Unlock: The puzzle will be unlocked once a certain number of puzzles from a chosen subset are solved. Use the puzzle chooser to indicate which puzzles count towards unlocking this puzzle. Then enter the number of puzzles required to unlock this puzzle in the "Num required to unlock" field. Setting the number of required puzzles to zero means that this puzzle will automatically be unlocked when the hunt starts.

Points Based Unlock: The puzzle will be unlocked once a team has earned enough points. Use the "Points cost" field to specify how many points a team needs to unlock this puzzle and the "Points value" to specify how many points solving this puzzle gives a team. Points will also be given according to the rate specified by the "Points per minute" field in the hunt object. Setting the "Points cost" field to zero means that this puzzle will automatically be unlocked when the hunt starts.

Either (OR) Unlocking: Fill out both of the above field pairs and the puzzle will be unlocked when either unlocking method's criteria is met.

Both (AND) Unlocking: Fill out both of the above field pairs and the puzzle will be unlocked when both unlocking method's criteria are met.

1.3.5 Auto-Response Objects

At the moment, whenever a user submits a correct answer, the server will respond with "Correct!" and whenever the user submits a wrong answer the server will respond with "Wrong Answer". Often you will want additional customized responses that can do things like tell the user how they are wrong or to tell them to "Keep going!".

To create automatic responses, use the "Responses" section at the bottom of the puzzle creation form. The "Regex" field is a python-style regex checked against the answer and the "Text" field is the text that will be returned to the team. The regexes are not applied in any specific order, so answers that match more than one regex will result in undefined behavior.

Tip: Response text can contain links using markdown style format: [foo](https://link.to.foo)

1.3.6 Puzzle Wrapup

After filling out everything on the puzzle creation page, hit "Save and add another" and continue to add puzzles until you have added all of the puzzles for the hunt. This will take a while; my recommendations are to be patient and have the unlocking graph on hand.

1.4 Create Prepuzzle Objects

As of version 3.3, the server now supports prepuzzles. A prepuzzle is a simpler puzzle that exists outside of the normal set of puzzles for a hunt. Prepuzzles are different in a number of ways:

- Prepuzzles do not require users to sign in
- Once published, prepuzzles are accessible before the hunt is open
- Prepuzzle submissions only support auto-response and do not show up on the queue page
- Prepuzzles can be, but do not need to be tied to any specific hunt.

Like other above objects, to create a prepuzzle object, navigate to the prepuzzle section of the admin pages and click the blue "+" icon in the upper right.

Below is a quick summary of the fields, most of them are similar to other fields above:

Puzzle name: The name the puzzle is given and shown to users

Released: Controls whether or not non-staff members can see the puzzle

Hunt: Select which hunt this prepuzzle is associated with, leave blank to not associate it with any hunt.

Answer: The answer to the puzzle, not case sensitive.

Template: See the "Prepuzzle Templating" section below

Resource link: Allows the optional inclusion of static files for the prepuzzle, must be a link to a publicly accessible ZIP file. See the "Prepuzzle Templating" section for details on how to reference the files.

Response string: The string that the server sends back to the prepuzzle page when the puzzle is solved. In the simple example, this string is just displayed to the user, but more complex templates could do anything they desire with this string.

Puzzle URL: This isn't really a field but rather an easy way to copy out the prepuzzle URL because it isn't currently accessible from anywhere on the site.

1.4.1 Prepuzzle Templating

As with the hunt "Template" field, everything typed into the "Template" form on the prepuzzle editing page will be run through Django's templating engine and rendered as HTML.

Again, more information about Django's templating language is available here: https://docs.djangoproject.com/en/2.2/ref/templates/language/.

Unlike the hunt template, the only variable that is passed to this template is a variable named "puzzle" containing the current prepuzzle object.

Just like the hunt template, it is recommended to use the below code to extend a basic template, in this case the template name is `prepuzzle.html`.

```
{% extends "prepuzzle.html" %}

{% block content %}
  Your content here
{% endblock content %}
```

The following blocks are available to override in the prepuzzle template:

{% block title %} This block controls what title is in the web browser tab. The default value for this block is the puzzle name.

{% block base_includes %} This block controls what content will be sourced/included before the standard Bootstrap and Jquery imports. This block contains the navbar formatting and the javascript helper functions discussed below, so it is not recommended to override this block without making a call to `{{ block.super }}` inside to include the existing contents.

{% block includes %} This block controls what content will be sourced/included after the standard Bootstrap and Jquery imports. This is for content that you want to use to extend those libraries, or content that relies on those libraries.

The prepuzzle template has some other special functionality added:

{% prepuzzle_static %} The `{% prepuzzle_static %}` tag allows access to the files from the prepuzzle's resource URL. It works just like the "hunt_static" tag.

check_answer(callback, answer) The prepuzzle base template supplies a function called `check_answer` that will deal with all of the server communication needed for answer checking. The function takes a callback function and the user's answer. The answer is then submitted to the server, and the the response from the server is then passed to the given callback function. The server response is a dictionary in the following form: `{is_correct: True, response: "response string"}`, where `is_correct` is a boolean indicating whether the answer matches the prepuzzle's answer and `response` is just a string that is either empty if the response was not correct, or the prepuzzle's given response string if the answer was correct.

{% include "prepuzzle_answerbox.html" %} If you use this include statement it will insert a no-hassle answer submission box that includes a spot for users to enter their answer, a submission button and will display the prepuzzle's response text if the answer was correct.

Warning: Just like the hunt template, you may use completely custom HTML if you want, but it is STRONGLY RECOMMENDED that you follow the instructions below on how to inherit the base template to get nice features like the header bar, common style sheets, Google analytics, and javascript helper funcions.

1.5 Hunt Creation Wrapup

If you've been following along, you should now have created everything needed to run a puzzlehunt. Head over to section 2: *How to Run a Hunt* for specific information on how to use the other parts of the staff site.

Chapter 2

How to Run a Hunt

So you want to run a puzzlehunt…

Here are all of the things you should need to know to run an already created puzzlehunt:

2.1 Staff Pages

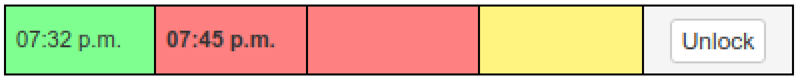
Below are descriptions of all of the custom staff pages, their features and how to interact with them. All of these pages are accessible under the “Other Staff Pages” header of the sidebar. The first 4 pages are critical to keep up during the hunt, the latter 4 pages are more situational and will likely be more useful before or after the hunt.

2.1.1 Progress Page

The progress page shows all the teams, and their progress in the hunt.

Tip: The table is pretty large, it is recommended to click the three lines next to “Django administration” at the top to collapse the side navbar for more room.

The main focus of the page is the large status table. The table lists puzzles across the top, and teams down the side. Each cell in the table is the team’s status on the corresponding puzzle, represented by one of 4 possible states:



From left to right:

Green Box with a Time The team solved the puzzle at the specified time.

Yellow/Orange/Red Box with a Time The team has unlocked the puzzle but has not yet solved the puzzle. The time is the time of the team’s last submitted guess. The box will start yellow when they first unlocked the puzzle and will slowly change to red over the course of 4 hours. This can be helpful to see how long a team has been stuck on a puzzle.

Yellow/Orange/Red Box without a Time The team has unlocked the puzzle but has not yet solved the puzzle and has not yet submitted a guess. The box follows the same yellow to red color scheme as above.

White box with an "Unlock" Button The team has not yet unlocked the puzzle. The button can be clicked to manually unlock the puzzle for the team.

On the left side of the table there are 3 columns next to the team's name. They indicate the number of metapuzzles the team has solved, the number of normal puzzles the team has solved and the last time that the team has solved a puzzle. Above the table there is a checkbox alongside 3 dropdowns that allows you to sort the table by these three columns instead of the default A-Z team name sort.

2.1.2 Queue Page

The queue page shows team's puzzle answer guesses as they come in. The main table on the page has one row for each submission. As the table header says, the table shows the team, the puzzle, the submission, the submission time, and the response. The row will be color coded for each submission: red for wrong and green for correct.

The response column also has a "Fix" link next to each response. Clicking it will bring up a form where you can edit the response to the submission. The edited response will automatically be pushed to the team's puzzle page. This can be used to nudge a team in the correct direction if they are close or have possibly just misspelled something.

Above the table there are two dropdown selectors and a "Filter" button. These can be used to filter the shown submissions on the queue page by Team or Puzzle.

2.1.3 Chat Page

The chat page allows staff to chat with teams during the hunt. Every team in the hunt has a button on the lefthand side that will bring up the chat box for that team. The currently selected team will be shown in blue, and teams with unread messages will be shown in red.

Attention: Due to the technical limitations of the server, only messages that have arrived while the chat page is open will cause the teams' name to turn red, so try not to refresh the page too often after the hunt starts. It's also not a bad idea to click around the teams every so often to make sure something hasn't slipped through the cracks.

It is possible to check the checkbox at the bottom to "Make the message an announcement". This will send the message to all teams in the hunt.

2.1.4 Hints Page

The hints page allows you to see hints that teams have requested and respond to them. Hints will appear one on top of another just like submissions on the queue page. In addition to the puzzle and team filters like the ones on the queue page, the hint page also has a "Filter by Status" dropdown that lets you view only the answered or unanswered hints.

Each hint that comes in will start with a space for you to type a response and hit submit. After hitting submit the response is sent to the team, but responses can be further edited by clicking the "Edit Response" link at the bottom, at which point the new response will be pushed to the team.

Finally, there is a button at the top left of the page titled "Show/Hide hint counts". Clicking this button will bring up a list of all of the teams and the number of hints they currently have available to them. Clicking the plus and minus buttons next to the number of available hints will give or take away available hints from the team.

Note: There is a very small chance that the team will naturally gain a hint in the same time period that you click to give them a hint. The counter will tick up by two in that case, you probably didn't double click.

2.1.5 Management Page

The first of the situational pages, the management page allows you to manage the resources and the overall state of the hunt.

The top portion of the page is list of hunts, one hunt per row. Each row has 3 buttons:

Set current This sets the selected hunt as the current hunt for all of the staff pages, the site front page and everywhere else.

Download Puzzles This downloads all PDFs and resources for all puzzles in the hunt.

Download Resources This downloads all resources for the hunt template page.

Each row can also be expanded to reveal an individual download button for each puzzle.

Underneath the "Hunt Downloads" section is a "Prepuzzle downloads" section which allows the downloading of resources for any chosen prepuzzle.

Finally, there is a single button at the bottom of the page titled "Reset all progress". This button resets all team interaction that has happened so far with a hunt, all submissions, responses, unlocks, solves, hints, and chat messages are deleted. This is normally only used once between playtesting and the start of the hunt.

2.1.6 Info Page

The info page lists information about teams that are signed up for a hunt and the people on them. Along the left is a list of all of the teams separated into 3 categories: "Needs a room", "Has a room", and "Off Campus" with each team having a text box next to their name with their current location. You can bulk edit team locations, for example assigning rooms to "Need a room" teams, and then click the "Update Locations" button at the bottom to save all the edits for a certain section.

Note: Teams are listed in signup order with the first team to sign up at the top so rooms can be assigned easily in sign up order.

Along the right side of the page is a statistic of how many people are registered for the hunt for things like ordering food, followed by all listed dietary restrictions of the registrants. Clicking any dietary restriction will take you to the corresponding user so you can either contact the user for more details or edit the restriction if the user has abused the field.

2.1.7 Email Page

The email page allows hunt staff to send an email out to all people registered for the hunt. To send an email, simply enter a subject, a body and hit send email.

If you want more customization or formatting than is available from the two simple textboxes, you can click the button at the bottom of the page to show the emails of all registered users to allow copy and pasting into your preferred email client.

2.1.8 Charts Page

Finally there is the charts page. There are no actions to take on the charts page, just a bunch of interesting charts. Most charts are pretty self-explanatory, and offer very helpful mouse-hover information.

The last item on the charts page isn't a chart at all, it is a table showing the first team to solve a puzzle and when that first solve happened.

2.2 Preparing for the Hunt

2.2.1 Download Puzzles

Before anybody can start playing your hunt, you have to download the puzzles. Sign into the staff part of the website located at `{server URL}/staff` and head over to the "Management" page located under the "Other Staff Pages" sidebar header. From there, click the "Puzzles" button next to the current hunt, which will download all of the puzzles. It takes a few minutes, be patient.

That should just work. If it doesn't, check your links and PDF accessibility and try again. (If you really think it is a bug, feel free to submit an issue on the github project)

If any individual puzzle fails to download or you just want to re-download a single puzzle for some reason, remember that you can un-collapse the hunt and click the download buttons for individual puzzles.

2.2.2 Playtesting

You probably want people to test your hunt before the actual event. This is easy using the puzzlehunt server. Just have the team of playtesters sign up like normal. Then navigate to the "Teams" page on the sidebar, find the team, check the "Playtester" checkbox on their edit page, fill in the playtest start and end dates and save the team. They will then have access to the puzzlehunt as if it was open to them during the given dates.

Attention: Playtest start and end dates are a new required part of having a team playtest as of version 4.0 due to the number of time based features now available.

All interactions with the playtest team should be done as they normally would be through the various staff pages described above. Things like the queue, the progress page, chat, puzzle unlocking, and hints should all work. The only feature currently not working for playtest teams is time released hints. If you want playtest teams to get hints, you will have to award them manually from the "hints" page.

Attention: Again, in a bigger orange box: Time released hints currently do not trigger for playtest teams, you must manually award hints from the "hints" page.

2.3 Running the Hunt

2.3.1 Pre-Hunt Checklist

Okay, the hunt is almost ready to happen, you've downloaded all the puzzles, you've had people playtest the hunt, and now you're ready to turn it over to the public. Below is a short checklist of items to consider before the hunt starts.

Before the hunt:

- [ ] *Make sure the hunt start time is accurate*
- [ ] Reset all progress from the management page
- [ ] Ensure all puzzles have working PDFs and images
- [ ] Ensure teams have been assigned rooms on the info page

2.3.2 During the Hunt

Hopefully your opening information session went well, the puzzles released flawlessly and people are now solving puzzles. Time to sit back and watch/make the magic happen. It is recommended to have the progress, queue, chat and hints pages open.

With version 4.0, puzzles should now automatically release at the set hunt start time, removing the need for the "release initial puzzles" button.

2.3.3 Hunt End

The hunt is nearing completion, hopefully everything went well and enough teams have completed the hunt for it to end. If you think the hunt hasn't run long enough, be sure to update the hunt end time before you reach it.

Once the hunt end time is reached, all puzzles will be available for the public and all hunt interfaces will update to indicate that the hunt is over.

2.4 Random Info and Common Issues

Teams can view their room assignments from the "team info" page: Let teams know that if they forget or lose their room assignments (or you just don't feel like telling them) that they can view their room assignments by clicking "View Registration" link on the front page.

What if I find a typo or other issue with a puzzle?: Simply fix the puzzle, make sure the new version is uploaded to Dropbox and click the download button for that puzzle from the management page.

What if I accidentally unlock a puzzle for a team I shouldn't have?: You can go to the "Unlocks" tab under the "Huntserver" section of the side navbar and delete the unlock object for that team/puzzle combo. The team will lose access to the puzzle.

Chapter 3

Setup

Instructions on how to setup a machine to run this project.

3.1 Basic Setup Instructions

This project now uses docker-compose as it's main form of setup. You can use the following steps to get a sample server up and going

1. Install [docker/docker-compose.](https://docs.docker.com/compose/install/)
2. Clone this repository.
3. Make a copy of `sample.env` named `.env` (yes, it starts with a dot).
4. Edit the new `.env` file, filling in new values for the first block of uncommented lines. Other lines can be safely ignored as they only provide additional functionality.
5. Run `docker-compose up` (possibly using `sudo` if needed)
6. You should now have the server running on a newly created VM, accessible via (http://localhost). The repository you cloned has been linked into the VM by docker, so any changes made to the repository on the host system should show up automatically. (A `docker-compose restart` may be needed for some changes to take effect)

3.1.1 Setup details

The basic instructions above bring up the following docker containers:

- **db** The postgres database with the settings specified in the .env file. Data is retained across container restarts in `docker/volumes/redis_data`.
- **redis** A redis server for caching and task management. Data is stored in `docker/volumes/redis_data`.
- **app** The Django application running using gunicorn on port 8000.
- **huey** A Huey consumer for scheduled tasks.
- **web** An apache server to proxy web requests to the "app" container and serve the static files. By default, this container serves web requests using plain HTTP over port 80. See the "Extra Setup Instructions" for details on setting up SSL.

Note: There are also 2 volumes shared by a number of the containers that hold static files and media files and will persist across docker restarts.

3.2 Extra Setup Instructions

In addition to the basic instructions above, there are a few additional setup options available. These additional options are provided via "override files" that override various parts of the docker compose logic. You can enable which override files are being used by setting the `COMPOSE_FILE` variable in the `.env` file. By default only the `local_override.yml` file is enabled.

3.2.1 local_override

By default, the "web" docker container only "exposes" port 80. The local override file takes things one step further and maps the host port 80 to the web container port 80. This is done via an override because docker compose doesn't support unmapping ports and the proxy_override settings need to map the reverse proxy to host port 80.

3.2.2 shib_override

Enabling this override sets up shibboleth authentication on the apache server. To use pre-existing shibboleth certificates, place sp-cert.pem and sp-key.pem in `docker/volumes/shib-certs`. This override file also uses LetsEncrypt to get a certificate for the site using the DOMAIN and CONTACT_EMAIL settings from the `.env` file. SSL certs are stored in `docker/volumes/ssl-certs`. Right now this is the only override that provides SSL capabilities. In the future there will likely be an SSL_override file that breaks out the LetsEncrypt functionality.

3.2.3 proxy_override

Enabling this override file sets up a reverse proxy using Traefik. This functionality is in development and mostly untested. It currently only works with shib_override. It also requires an already created docker network named `proxy-net`

Chapter 4

Basics

Despite it's size, this project only has one main app named huntserver which does nearly everything. This page is meant to outline basic low level operational aspects and design choices of the server. This information is really only helpful for people looking to help develop or modify the application. If you're just using the application, you can skip all this. (models and views too)

4.1 Design

The design of this project is somewhat divided into two parts, the staff experience and the hunt participant experience.

Staff is anyone that has the staff attribute set in the admin page. These users have access to the /staff/ area of the site; however, in order to access all functions and access the /admin/ area of the site, the user must also be a superuser as designated by Django.

4.2 Dynamic Content

Dynamic content is created by using a combination of the model-view controller and the default Django templating engine. Both are extensively documented on Django's website. Both models and views used in this project are documented by later pages.

4.3 Static Content

Puzzles should not be checked into the Github repository. They should exist on some accessible online file source (we have used Dropboxin the past) and will be downloaded and converted when the admin choses to do so. Once downloaded, the puzzle files live in `{PROJECT FOLDER}/media/puzzles/` and are named using the "puzzle id" field of the puzzle which is enforced to be unique to each puzzle.

To protect users from being able to just go to `/media/puzzles/{Puzzle_id}.pdf` and get puzzles, the server comes included with a protected routing path utilizing X-Sendfile. The /protected/ URL will only allow a user to access puzzle files if they have unlocked the puzzle. To avoid hard-coding that path, you can use the variable "settings.PROTECTED_URL" after importing the project settings.

It is a bit simplistic, but anything in the puzzles directory is permission guarded by the set of hexadecimal characters before the '-' or '.' of the filename. If the requesting user has access to the puzzle object with the corresponding puzzle_id, then they will have access to that file. You can use this to protect files other than just the puzzle PDFs and PNGs.

You should protect your /media/puzzles URL by only allowing access to /media/puzzles/ from internal sources. The Apache configuration for this project includes protection like this already.

4.4 Database

As noted in setup, the default database for this project is a Postgres database. After setup, the database should never need to be modified by hand, additions or deletions should be done from the online admin GUI or if absolutely necessary, from the Django interactive shell. Modifications to the table structure should only be done by modifying models.py and using the automatically created migration files.

Chapter 5

Models

class huntserver.models.**Hint**(**args*, ***kwargs*)

A class to represent a hint to a puzzle

Parameters

- **id** (*AutoField*) – Id
- **puzzle_id** (ForeignKey to *Puzzle*) – The puzzle that this hint is related to
- **team_id** (ForeignKey to *Team*) – The team that requested the hint
- **request** (*TextField*) – The text of the request for the hint
- **request_time** (*DateTimeField*) – Hint request time
- **response** (*TextField*) – The text of the response to the hint request
- **response_time** (*DateTimeField*) – Hint response time
- **last_modified_time** (*DateTimeField*) – Last time of modification

exception DoesNotExist

exception MultipleObjectsReturned

class huntserver.models.**HintUnlockPlan**(**args*, ***kwargs*)

A class to represent when Teams are given hints

Parameters

- **id** (*AutoField*) – Id
- **hunt_id** (ForeignKey to *Hunt*) – The hunt that this hint unlock plan refers to
- **unlock_type** (*CharField*) – The type of hint unlock plan
- **unlock_parameter** (*IntegerField*) – Parameter (Time / Interval / Solves)
- **num_triggered** (*IntegerField*) – Number of times this Unlock Plan has given a hint

exception DoesNotExist

exception MultipleObjectsReturned

reset_plan()

Resets the HintUnlockPlan

class `huntserver.models.`**`Hunt`**`(*args, **kwargs)`

Base class for a hunt. Contains basic details about a puzzlehunt.

Parameters

- **`id`** (*`AutoField`*) – Id
- **`hunt_name`** (*`CharField`*) – The name of the hunt as the public will see it
- **`hunt_number`** (*`IntegerField`*) – A number used internally for hunt sorting, must be unique
- **`team_size`** (*`IntegerField`*) – Team size
- **`start_date`** (*`DateTimeField`*) – The date/time at which a hunt will become visible to registered users
- **`end_date`** (*`DateTimeField`*) – The date/time at which a hunt will be archived and available to the public
- **`display_start_date`** (*`DateTimeField`*) – The start date/time displayed to users
- **`display_end_date`** (*`DateTimeField`*) – The end date/time displayed to users
- **`location`** (*`CharField`*) – Starting location of the puzzlehunt
- **`resource_file`** (*`FileField`*) – Hunt resources, MUST BE A ZIP FILE.
- **`is_current_hunt`** (*`BooleanField`*) – Is current hunt
- **`extra_data`** (*`CharField`*) – A misc. field for any extra data to be stored with the hunt.
- **`template`** (*`TextField`*) – The template string to be rendered to HTML on the hunt page
- **`hint_lockout`** (*`IntegerField`*) – The number of minutes before a hint can be used on a newly unlocked puzzle
- **`points_per_minute`** (*`IntegerField`*) – The number of points granted per minute during the hunt

exception DoesNotExist

exception MultipleObjectsReturned

`clean``(*args, **kwargs)`

Overrides the standard clean method to ensure that only one hunt is the current hunt

`dummy_team`

The dummy team for the hunt

`in_reg_lockdown`

A boolean indicating whether or not registration has locked for this hunt

`is_day_of_hunt`

A boolean indicating whether or not today is the day of the hunt

`is_locked`

A boolean indicating whether or not the hunt is locked

`is_open`

A boolean indicating whether or not the hunt is open to registered participants

`is_public`

A boolean indicating whether or not the hunt is open to the public

`real_teams`

A queryset of all non-dummy teams in the hunt

save(**args*, ***kwargs*)
Overrides the standard save method to ensure that only one hunt is the current hunt

season
Gets a season string from the hunt dates

team_from_user(*user*)
Takes a user and a hunt and returns either the user's team for that hunt or None

class huntserver.models.**HuntAssetFile**(**args*, ***kwargs*)
A class to represent an asset file for a puzzlehunt

Parameters

- **id** (*AutoField*) – Id
- **file** (*FileField*) – File

exception DoesNotExist

exception MultipleObjectsReturned

class huntserver.models.**Message**(**args*, ***kwargs*)
A class that represents a message sent using the chat functionality

Parameters

- **id** (*AutoField*) – Id
- **team_id** (ForeignKey to *Team*) – The team that this message is being sent to/from
- **is_response** (*BooleanField*) – A boolean representing whether or not the message is from the staff
- **text** (*CharField*) – Message text
- **time** (*DateTimeField*) – Message send time

exception DoesNotExist

exception MultipleObjectsReturned

class huntserver.models.**OverwriteStorage**(*location=None*, *base_url=None*, *file_permissions_mode=None*, *directory_permissions_mode=None*)
A custom storage class that just overwrites existing files rather than erroring

get_available_name(*name*, *max_length=None*)
Return a filename that's free on the target storage system and available for new content to be written to.

class huntserver.models.**Person**(**args*, ***kwargs*)
A class to associate more personal information with the default django auth user class

Parameters

- **id** (*AutoField*) – Id
- **user_id** (OneToOneField to User) – The corresponding user to this person
- **phone** (*CharField*) – Person's phone number, no particular formatting
- **allergies** (*CharField*) – Allergy information for the person
- **comments** (*CharField*) – Comments or other notes about the person
- **is_shib_acct** (*BooleanField*) – A boolean to indicate if the person uses shibboleth authentication for login

- **`teams`** (*`ManyToManyField`*) – Teams that the person is on

`exception DoesNotExist`

`exception MultipleObjectsReturned`

`class` `huntserver.models.`**`Prepuzzle`**`(`**args*, ***kwargs*`)`

A class representing a pre-puzzle within a hunt

Parameters

- **`id`** (*`AutoField`*) – Id
- **`puzzle_name`** (*`CharField`*) – The name of the puzzle as it will be seen by hunt participants
- **`released`** (*`BooleanField`*) – Released
- **`hunt_id`** (OneToOneField to *`Hunt`*) – The hunt that this puzzle is a part of, leave blank for no associated hunt.
- **`answer`** (*`CharField`*) – The answer to the puzzle, not case sensitive
- **`template`** (*`TextField`*) – The template string to be rendered to HTML on the hunt page
- **`resource_file`** (*`FileField`*) – Prepuzzle resources, MUST BE A ZIP FILE.
- **`response_string`** (*`TextField`*) – Data returned to the webpage for use upon solving.

`exception DoesNotExist`

`exception MultipleObjectsReturned`

`save``(`**args*, ***kwargs*`)`

Save the current instance. Override this in a subclass if you want to control the saving process.

The ‘force_insert’ and ‘force_update’ parameters can be used to insist that the “save” must be an SQL insert or update (or equivalent for non-SQL backends), respectively. Normally, they should not be set.

`class` `huntserver.models.`**`Puzzle`**`(`**args*, ***kwargs*`)`

A class representing a puzzle within a hunt

Parameters

- **`id`** (*`AutoField`*) – Id
- **`hunt_id`** (ForeignKey to *`Hunt`*) – The hunt that this puzzle is a part of
- **`puzzle_name`** (*`CharField`*) – The name of the puzzle as it will be seen by hunt participants
- **`puzzle_number`** (*`IntegerField`*) – The number of the puzzle within the hunt, for sorting purposes
- **`puzzle_id`** (*`CharField`*) – A 3-5 character hex string that uniquely identifies the puzzle
- **`answer`** (*`CharField`*) – The answer to the puzzle, not case sensitive
- **`is_meta`** (*`BooleanField`*) – Is this puzzle a meta-puzzle?
- **`puzzle_page_type`** (*`CharField`*) – The type of webpage for this puzzle.
- **`doesnt_count`** (*`BooleanField`*) – Should this puzzle not count towards scoring?
- **`puzzle_file`** (*`FileField`*) – Puzzle file. MUST BE A PDF
- **`resource_file`** (*`FileField`*) – Puzzle resources, MUST BE A ZIP FILE.
- **`solution_is_webpage`** (*`BooleanField`*) – Is this solution an html webpage?

- **`solution_file`** (*`FileField`*) – Puzzle solution. MUST BE A PDF.
- **`solution_resource_file`** (*`FileField`*) – Puzzle solution resources, MUST BE A ZIP FILE.
- **`extra_data`** (*`CharField`*) – A misc. field for any extra data to be stored with the puzzle.
- **`unlock_type`** (*`CharField`*) – The type of puzzle unlocking scheme
- **`num_required_to_unlock`** (*`IntegerField`*) – Number of prerequisite puzzles that need to be solved to unlock this puzzle
- **`points_cost`** (*`IntegerField`*) – The number of points needed to unlock this puzzle.
- **`points_value`** (*`IntegerField`*) – The number of points this puzzle grants upon solving.
- **`unlocks`** (*`ManyToManyField`*) – Puzzles that this puzzle is a possible prerequisite for

exception DoesNotExist

exception MultipleObjectsReturned

`save`(**args*, ***kwargs*)
Save the current instance. Override this in a subclass if you want to control the saving process.

The 'force_insert' and 'force_update' parameters can be used to insist that the "save" must be an SQL insert or update (or equivalent for non-SQL backends), respectively. Normally, they should not be set.

`serialize_for_ajax`()
Serializes the ID, puzzle_number and puzzle_name fields for ajax transmission

class `huntserver.models.`**`PuzzleOverwriteStorage`**(*location=None*, *base_url=None*, *file_permissions_mode=None*, *directory_permissions_mode=None*)
A custom storage class that just overwrites existing files rather than erroring

`get_available_name`(*name*, *max_length=None*)
Return a filename that's free on the target storage system and available for new content to be written to.

`url`(*name*)
Return an absolute URL where the file's contents can be accessed directly by a Web browser.

class `huntserver.models.`**`Response`**(**args*, ***kwargs*)
A class to represent an automated response regex

Parameters

- **`id`** (*`AutoField`*) – Id
- **`puzzle_id`** (ForeignKey to *`Puzzle`*) – The puzzle that this automated response is related to
- **`regex`** (*`CharField`*) – The python-style regex that will be checked against the user's response
- **`text`** (*`CharField`*) – The text to use in the submission response if the regex matched

exception DoesNotExist

exception MultipleObjectsReturned

class `huntserver.models.`**`Solve`**(**args*, ***kwargs*)
A class that links a team and a puzzle to indicate that the team has solved the puzzle

Parameters

- **id** (*AutoField*) – Id
- **puzzle_id** (ForeignKey to *Puzzle*) – The puzzle that this is a solve for
- **team_id** (ForeignKey to *Team*) – The team that this solve is from
- **submission_id** (ForeignKey to *Submission*) – The submission object that the team submitted to solve the puzzle

exception DoesNotExist

exception MultipleObjectsReturned

serialize_for_ajax()
Serializes the puzzle, team, time, and status fields for ajax transmission

class huntserver.models.**Submission**(**args*, ***kwargs*)
A class representing a submission to a given puzzle from a given team

Parameters

- **id** (*AutoField*) – Id
- **team_id** (ForeignKey to *Team*) – The team that made the submission
- **submission_time** (*DateTimeField*) – Submission time
- **submission_text** (*CharField*) – Submission text
- **response_text** (*CharField*) – Response to the given answer. Empty string indicates human response needed
- **puzzle_id** (ForeignKey to *Puzzle*) – The puzzle that this submission is in response to
- **modified_date** (*DateTimeField*) – Last date/time of response modification

exception DoesNotExist

exception MultipleObjectsReturned

convert_markdown_response
The response with all markdown links converted to HTML links

create_solve()
Creates a solve based on this submission

is_correct
A boolean indicating if the submission given is exactly correct

respond()
Takes the submission's text and uses various methods to craft and populate a response. If the response is correct a solve is created and the correct puzzles are unlocked

save(**args*, ***kwargs*)
Overrides the default save function to update the modified date on save

serialize_for_ajax()
Serializes the time, puzzle, team, and status fields for ajax transmission

update_response(*text*)
Updates the response with the given text

class huntserver.models.**Team**(**args*, ***kwargs*)
A class representing a team within a hunt

Parameters

- **id** (*AutoField*) – Id
- **team_name** (*CharField*) – The team name as it will be shown to hunt participants
- **hunt_id** (ForeignKey to *Hunt*) – The hunt that the team is a part of
- **location** (*CharField*) – The physical location that the team is solving at
- **join_code** (*CharField*) – The 5 character random alphanumeric password needed for a user to join a team
- **playtester** (*BooleanField*) – A boolean to indicate if the team is a playtest team and will get early access
- **playtest_start_date** (*DateTimeField*) – The date/time at which a hunt will become to the playtesters
- **playtest_end_date** (*DateTimeField*) – The date/time at which a hunt will no longer be available to playtesters
- **num_waiting_messages** (*IntegerField*) – The number of unseen messages a team has waiting
- **num_available_hints** (*IntegerField*) – The number of hints the team has available to use
- **num_unlock_points** (*IntegerField*) – The number of points the team has earned
- **solved** (*ManyToManyField*) – The puzzles the team has solved
- **unlocked** (*ManyToManyField*) – The puzzles the team has unlocked
- **unlockables** (*ManyToManyField*) – The unlockables the team has earned

exception DoesNotExist

exception MultipleObjectsReturned

hints_open_for_puzzle(*puzzle*)
Takes a puzzle and returns whether or not the team may use hints on the puzzle

is_normal_team
A boolean indicating whether or not the team is a normal (non-playtester) team

is_playtester_team
A boolean indicating whether or not the team is a playtesting team

playtest_happening
A boolean indicating whether or not the team's playtest slot is currently happening

playtest_over
A boolean indicating whether or not the team's playtest slot has passed

playtest_started
A boolean indicating whether or not the team is currently allowed to be playtesting

reset()
Resets/deletes all of the team's progress

short_name
Team name shortened to 30 characters for more consistent display

size
The number of people on the team

unlock_hints()
Gives teams the appropriate number of hints based on "Solves" HintUnlockPlans

unlock_puzzles()
Unlocks all puzzles a team is currently supposed to have unlocked

class huntserver.models.**Unlock**(**args*, ***kwargs*)
A class that links a team and a puzzle to indicate that the team has unlocked the puzzle

Parameters

- **id** (*AutoField*) – Id
- **puzzle_id** (ForeignKey to *Puzzle*) – The puzzle that this is an unlock for
- **team_id** (ForeignKey to *Team*) – The team that this unlocked puzzle is for
- **time** (*DateTimeField*) – The time this puzzle was unlocked for this team

exception DoesNotExist

exception MultipleObjectsReturned

serialize_for_ajax()
Serializes the puzzle, team, and status fields for ajax transmission

class huntserver.models.**Unlockable**(**args*, ***kwargs*)
A class that represents an object to be unlocked after solving a puzzle

Parameters

- **id** (*AutoField*) – Id
- **puzzle_id** (ForeignKey to *Puzzle*) – The puzzle that needs to be solved to unlock this object
- **content_type** (*CharField*) – The type of object that is to be unlocked, can be 'IMG', 'PDF', 'TXT', or 'WEB'
- **content** (*CharField*) – The link to the content, files must be externally hosted.

exception DoesNotExist

exception MultipleObjectsReturned

Chapter 6

Views

6.1 Hunt Views

`huntserver.hunt_views.`**`protected_static`**(*request*, *file_path*)
A view to serve protected static content. Does a permission check and if it passes, the file is served via X-Sendfile.

`huntserver.hunt_views.`**`hunt`**(*request*, *hunt_num*)
The main view to render hunt templates. Does various permission checks to determine the set of puzzles to display and then renders the string in the hunt's "template" field to HTML.

`huntserver.hunt_views.`**`current_hunt`**(*request*)
A simple view that calls `huntserver.hunt_views.hunt` with the current hunt's number.

`huntserver.hunt_views.`**`prepuzzle`**(*request*, *puzzle_id*)
A view to handle answer submissions via POST and render the prepuzzle's template.

`huntserver.hunt_views.`**`hunt_prepuzzle`**(*request*, *puzzle_id*)
A simple view that locates the correct prepuzzle for a hunt and redirects there if it exists.

`huntserver.hunt_views.`**`current_prepuzzle`**(*request*, *puzzle_id*)
A simple view that locates the correct prepuzzle for the current hunt and redirects to there.

`huntserver.hunt_views.`**`puzzle_view`**(*request*, *puzzle_id*)
A view to handle answer submissions via POST, handle response update requests via AJAX, and render the basic per-puzzle pages.

`huntserver.hunt_views.`**`puzzle_hint`**(*request*, *puzzle_id*)
A view to handle hint requests via POST, handle response update requests via AJAX, and render the basic puzzle-hint pages.

`huntserver.hunt_views.`**`chat`**(*request*)
A view to handle message submissions via POST, handle message update requests via AJAX, and render the hunt participant view of the chat.

`huntserver.hunt_views.`**`chat_status`**(*request*)
A view ajax requests for the status of waiting chat messages for a team.

`huntserver.hunt_views.`**`unlockables`**(*request*)
A view to render the unlockables page for hunt participants.

6.2 Info Views

`huntserver.info_views.`**`index`**(*request*)
: Main landing page view, mostly static with the exception of hunt info

`huntserver.info_views.`**`previous_hunts`**(*request*)
: A view to render the list of previous hunts, will show any hunt that is 'public'

`huntserver.info_views.`**`registration`**(*request*)
: The view that handles team registration. Mostly deals with creating the team object from the post request. The rendered page is nearly entirely static.

`huntserver.info_views.`**`user_profile`**(*request*)
: A view to handle user information update POST data and render the user information form.

6.3 Staff Views

`huntserver.staff_views.`**`queue`**(*request*, *page_num*)
: A view to handle queue response updates via POST, handle submission update requests via AJAX, and render the queue page. Submissions are pre-rendered for standard and AJAX requests.

`huntserver.staff_views.`**`progress`**(*request*)
: A view to handle puzzle unlocks via POST, handle unlock/solve update requests via AJAX, and render the progress page. Rendering the progress page is extremely data intensive and so the view involves a good amount of pre-fetching.

`huntserver.staff_views.`**`charts`**(*request*)
: A view to render the charts page. Mostly just collecting and organizing data

`huntserver.staff_views.`**`admin_chat`**(*request*)
: A view to handle chat update requests via AJAX and render the staff chat page. Chat messages are pre-rendered for both standard and AJAX requests.

`huntserver.staff_views.`**`hunt_management`**(*request*)
: A view to render the hunt management page

`huntserver.staff_views.`**`hunt_info`**(*request*)
: A view to render the hunt info page, which contains room and allergy information

`huntserver.staff_views.`**`control`**(*request*)
: A view to handle all of the different management actions from staff users via POST requests. This view is not responsible for rendering any normal pages.

`huntserver.staff_views.`**`staff_hints_text`**(*request*)
: A view to handle hint response updates via POST, handle hint request update requests via AJAX, and render the hint page. Hints are pre-rendered for standard and AJAX requests.

`huntserver.staff_views.`**`staff_hints_control`**(*request*)
: A view to handle the incrementing, decrementing, and updating the team hint counts on the hints staff page.

`huntserver.staff_views.`**`emails`**(*request*)
: A view to send emails out to hunt participants upon receiving a valid post request as well as rendering the staff email form page

`huntserver.staff_views.`**`lookup`**(*request*)
: A view to search for users/teams

6.4 Auth Views

`huntserver.auth_views.`**`login_selection`**`(`*request*`)`
A mostly static view to render the login selection. Next url parameter is preserved.

`huntserver.auth_views.`**`create_account`**`(`*request*`)`
A view to create user and person objects from valid user POST data, as well as render the account creation form.

`huntserver.auth_views.`**`account_logout`**`(`*request*`)`
A view to logout the user and *hopefully* also logout out the shibboleth system.

`huntserver.auth_views.`**`shib_login`**`(`*request*`)`
A view that takes the attributes that the shibboleth server passes back and either logs in or creates a new shibboleth user. The view then redirects the user back to where they were.

Changelog Here

Chapter 7

Contribute

Source Code: http://www.github.com/dlareau/puzzlehunt_server

Issue Tracker: http://www.github.com/dlareau/puzzlehunt_server/issues

If you are having issues, please let us know. Email dlareau@cmu.edu

Index