Developer's Guide

Reservation System App

(`http://reservaionsystem.appspot.com`)

(Sahil Goel)

Models

I have used 3 models. (Resource, Reservations, Availabilities)

1. **Resource**

   Name(String):
   Name of the resource

   Availability(List of *TimeSlot*)
   To store the availabilities time slots. Example [00:00-01:00],[05:00-06:00] as it might have been reserved for [01:00-05:00]

   originalStartTime(DateTime)
   Availability can change due to reservations. originalStartTime is used to store the original available start time of the resource, so that when editing the resource, this information can be used in displaying in the form as default values.

   originalEndTime(DateTime)
   To store original end time for resource. Reason same as for original Start Time

   Tags(String)
   Stores all the tags in csv format. Example Electronics, Education

   Reservations(List of *Reservations*)
   Stores all the reservations for the current resource object

   Owner(String)
   Stores the resource owner's email address

   lastReservation(DateTimeProperty)
   Stores the time, when the last reservation was made for that object

Uid(String)
Unique id which uniquely identifies a resource. This is randomly generated when a resource is created.

2. Reservations

resourceName(String)
It stores, the name of the resource for which this reservation is made

resourceUid(String)
It stores the uid of the resource for which this reservation is made

reservedBy(String)
It stores the email address of the person who made this reservation

startTime(DateTime)
It stores the start time of the reservation

duration(String)
It stores the duration for current reservation

uid(String)
It stores the unique id for current reservation.

3. TimeSlot

startTime(DateTime)
Stores the start time of availability slot

endTime(DateTime)
Stores the end time of the availability slot

Some Design Decisions:

1. resource_key(): Whenever Resource table is queried in the datastore, (ancestor = resource_key()) is passed as an argument. resource_key() returns a key formed from string (currentDay+_+currentMonth+_+currentyear). I have chosen to work it this way, since we are dealing with resources on day to day basis. So whenever we display resources, we can retrieve the resources which are available today.

2. reservation_key(): Whenever Reservations table is queried in datastore, (ancestor=reservations_key(user.email())) is passed as an argument. reservations_key(user.email()) returns a key formed by concatenating currentDate+_+user.email(). I have chosen to work it this way, since we are only retrieving the reservations made by some specific user. This helps to get the reservations faster.

3. **Edit a resource:** Whenever an owner of resource edits a resource, all the previous reservations for that resource are deleted since those might not be possible now.

Design(View):

Main Page(Landing Page): *(class MainPage)*

If there is no current active user, page redirects to login page.

Main page has 5 sections:

1. **My Reservations:** This section shows the reservations, which the current active user has made. It also has a button to delete the current reservation. It links to Delete Reservation page.

2. **All Resources in System:** This section shows all the resources in the system

3. **Owned Resources:** This section shows the resources, which the current user owns i.e. he added those resources

4. **Add a resource** button: It links to a Add Resource Page.

5. Search by name: It allows user to search for a resource by name. If suppose current resources in system are (Phone, Laptop, Room, Book) and user searches for 'p' then Phone and Laptop will be displayed. Also this search is case insensitive. It links to **Search Resource** Page

Wherever a resource name is shown, it links to **Resource Main** page, which displays information about that resource. It also has a link to RSS, to dump RSS Feed. UID for that resource is passed as URL parameter.

Add Resource Page:*(class AddResource)*

This page displays a form, which has 5 inputs

1. **Resource Name**: User can enter any resource name.

2. **Start Time**: User can enter any start time for the availability of that resource in (HH:MM) format. It can take values between (00:00 and 23:59)

3. **End Time**: User can enter any end time for the availability of that resource in (HH:MM) format. It can also take values between (00:00 and 23:59)

4. **Tags**: User can enter any number of tags (comma separated), which describe the resource. Example for a resource 'Room', tags can be 'Hotel, Taj' etc. Each tag has a link to Tag Resource Page.

5. **Submit**: When a user clicks on submit, a request is made and all the form values are submitted.

Form validation on Add Resource Page:

Following validations are done on the add resource page:

1. Resource name cannot be empty. If it is empty, the error is displayed to user.

2. Start Time should be in format HH:MM. If it is not in this format, error is displayed to user. Also if it is not in limits, UI doesn't accept input and shows the error to user.

3. End Time also goes through all of the validations as Start Time. Apart from that, it also goes through one more validation i.e. it checks if End Time is after Start Time. If it not after Start Time, error is displayed to user.

In all cases, if any error is shown to user, the values of all the form input fields are not lost, so that is more user friendly.

Resource Main Page:*(class ResourceMain)*

This page displays information about the resource queried.

1. **Current and Upcoming Reservations:**
   It displays the current and upcoming reservations for the current resource. For each reservation, the following attributes are displayed:
   (a). Owner of that reservation
   (b). Start time of reservation
   (c). Duration of reservation

2. Total number of reservations for that resource including past reservations

3. Link to Add a Reservation

4. **Edit Resource(Only if the current user is the owner of that resource):** It links to Edit Resource page.

In first section, wherever the owner id is displayed (owner of the reservation), it has a link to **User Profile Page**.

Search Resource Page:(class *SearchResource*)

This page displays all the resources, which matched the user's search query

For each resource following information is displayed:

1. **Resource Name:** Name of the resource. It has links to Generate Rss Page and Resource Main Page. UID for that resource is passed as URL parameter.
2. **Availabilities:** It takes into account all the reservations for that resource
3. **Tags:** Tags, which describes the resource. Each tag links to Tag Resource Page

Generate Rss Page:(class *GenerateRss*)

This page receives uid for a resource as a URL parameter and dumps the RSS feed for that resource. RSS Feed has the following format

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<Resource>
<Name> Resource Name </Name>
<Owner>Resource Owner</Owner>
<Tags>
<Tag>Tag 1</Tag>
<Tag>Tag 2</Tag>
</Tags>
<Reservations>
<Reservation>
<ReservedBy>Owner 1</ReservedBy>
<StartTime>Start Time 1</StartTime>
<Duration> Duration 1</Duration>
<Reservation>
</Reservations>
</Resource>
</rss>
```

Tag Resource Page:(class *TagResource*)

This page receives the tag as the URL parameter. It then displays all the resources, which has that tag. For each resource, following information is displayed:

1. Resource Name: Name of the resource. It has links to Resource Main page and Generate RSS Page.
2. Availabilities: It takes into account all the reservations for that resource
3. Tags: Tags, which describes the resource. Each tag links to Tag Resource Page.

Edit Resource Page:(class *EditResource*)

This page receives the uid of the resource as URL parameter and then displays a form with already existing values for that resource. The form has following fields:

1. Resource Name
2. Start Time
3. End Time
4. Tags

It is similar to Add Resource Page form and similar validation is done here to ensure the correctness of input

Delete Reservation Page:(class *DeleteReservation*)

This page accepts the reservation uid as input and it displays the following information about the reservation:

1. **Resource Name for Reservation**
2. **Start Time of Reservation**
3. **Duration of Reservation**

After the reservation section, it has a form, which confirms from user if he wants to delete this reservation or not. If user selects no, page is redirected to main page. Otherwise the reservation is deleted and resource availability is modified to reflect the deletion of this reservation.

Add Reservation Page:(class *AddReservation*)

This page accepts resource uid as input. It then allows user to create a reservation for that resource. It displays a form to user to add a reservation. The form has following fields:

1. **Start Time**: Start Time of the reservation in HH:MM format.
2. **Duration**: Duration for which, user wants to make a reservation

Validation on start time is done(similar to Add Resource page). **One extra validation** is done to ensure that resource is available during the specified time period. If it is not available, error is displayed to user.

If everything is validated, a new reservation is created and put into datastore. Also resource reservations, availabilities and lastReservation are modified accordingly to reflect changes. It then redirects to Main Page. Also whenever a reservation is added, a mail is sent to reservation owner confirming his/her reservation.

User Profile Page:(class *UserProfile*)

This page accepts the user email as input and displays 2 sections.

1. **Reservations made by that user**
   For each reservation following information is displayed:
   (a). **Resource Name** (Link to Generate RSS and Resource Main Page)
   (b). **Start Time of Reservation**
   (c). **Duration**

2. **Resources owned by that user**

   For each resource following information is displayed:
   (a). **Resource Name** (Link to Generate RSS and Resource Main Page)
   (b). **Availabilities**: Available time slots for the resource
   (c). **Tags:** Tags, which describe the resource. It also has link to Tag Resource Page.

Assumptions

1. I am assuming when a user adds a resource and give start time and end time, it is for today i.e. the resource will be available today from start time to end time

2. Since the requirements mentioned that we don't need to take care of cases where if a user tries to reserve a resource, which finishes after the day finishes, those cases are not handled explicitly. They may result in an error if user tries to do so.

3. According to 5th requirement, owner can edit his/her resource. If owner edits a resource, which already had reservations, it will delete those reservations.

4. In RSS dump, I have included the all the reservations including the past ones, since it makes sense to dump all the information for later use.

5. I named another branch as Requirement-11 instead of Experiment.

6. Extra credits 1 -> On the resource main page, I am showing the number of times this resource has been reserved including current and upcoming reservations.

7. Extra credits 2 -> On main page, I have included a section to search for a resource by a key. If search key is "R" then all the resources whose names contain "R" will be displayed.

8. Extra credit 5 implemented. Please check spam folder as the mail goes to spam folder.