SKILLBUILDER DOCUMENTATION

version

Jeffrey Kwei Afutu

June 22, 2023

Contents

Welcome to SkillBuilder’s documentation! **1**

SkillBuilder 1

About me 1

All about project 1

myapp package 1

myapp.forms module 2

myapp.models module 4

myapp.tokens module 7

course package 7

course.admin module 9

course.forms module 9

course.models module 11

quiz package 19

quiz.views module 19

quiz.admin module 20

quiz.models module 20

event package 24

event.models module 24

employerAdmin package 25

employerAdmin.admin module 25

employerAdmin.employer_admin module 27

Indices and tables **28**

Index **29**

Python Module Index **35**

Welcome to SkillBuilder’s documentation!

SkillBuilder

About me

All about project

SKILLBUILDER PROJECT (Employee Training App) This is an employee training app to enable employers to perform CRUD on announcement, resources, quiz, events etc Employees would get resources, announcement, quiz etc created by their employer

Customer Requirements Your users should be able to: Signup & log in with an email and password with account verification. There should be a reset password feature to recover lost passwords password. Employees Can download resources Employees can view announcement Real time chat app Admin performs CRUD on Course, Announcement, Resource, Room and Quiz

myapp package

myapp.views module

myapp.views.authentication_page module

`myapp.views.authentication_page.`**`activate_user`**`(request, uidb64, token)`

View function to activate user account.

Parameters:
- **request** – The HTTP request object.
- **uidb64** (*str*) – The encoded user ID.
- **token** (*str*) – The activation token.

Returns: A rendered HTML template for successful account activation or activation failure.

`myapp.views.authentication_page.`**`employeeSignupPage`**`(request)`

View function to handle employee signup.

Parameters: **request** – The HTTP request object.

Returns: A rendered HTML template for the employee signup page.

`myapp.views.authentication_page.`**`employerSignupPage`**`(request)`

View function to handle employer signup.

Parameters: **request** – The HTTP request object.

Returns: A rendered HTML template for the employer signup page.

`myapp.views.authentication_page.`**`loginPage`**`(request)`

View function to handle user login.

Parameters: **request** – The HTTP request object.

Returns: A rendered HTML template for the login page.

`myapp.views.authentication_page.`**`logoutUser`**`(request)`

View function to handle user logout.

Parameters: **request** – The HTTP request object.

Returns: A redirection to the login page.

`myapp.views.authentication_page.`**`send_activation_email`**`(user, request)`

Function to send activation email to the user.

Parameters:
- **user** – The User object.
- **request** – The HTTP request object.

Returns: None.

myapp.views.front_page module

`myapp.views.front_page.`**`about`** `(request)`

View function to render the about page.

Parameters: **request** – The HTTP request object.

Returns: A rendered HTML template for the about page.

`myapp.views.front_page.`**`contact`** `(request)`

View function to handle the contact form submission.

Parameters: **request** – The HTTP request object.

Returns: A redirection to the contact page after submitting the form, or a rendered HTML template for the contact page if it's a GET request.

`myapp.views.front_page.`**`home`** `(request)`

View function to render the home page.

Parameters: **request** – The HTTP request object.

Returns: A rendered HTML template for the home page.

myapp.views.user_management_page module

`myapp.views.user_management.`**`updateUser`** `(request, pk)`

View function to update a user's information.

Parameters:
- **request** – The HTTP request object.
- **pk** – The primary key of the user to be updated.

Returns: A redirection to the update user page after successfully updating the user, or a rendered HTML template for the update user page if it's a GET request.

myapp.views.users_page module

`myapp.views.users_page.`**`employeeHome`** `(request, pk)`

View function for the employee home page.

Parameters:
- **request** – The HTTP request object.
- **pk** – The primary key of the user.

Returns: A rendered HTML template for the employee home page, which includes the enrollment form and a list of available courses. If a course is selected for enrollment, the user is enrolled in the course and appropriate success messages are displayed.

myapp.forms module

class `myapp.forms.`**`EmployeeSignUpForm`** `(*args, **kwargs)`

Bases: **`UserCreationForm`**

Form for employee sign up.

Inherits from UserCreationForm and adds additional fields such as first name, last name, email, and employer selection.

class **`Meta`**

Bases: **`object`**

fields = *('username', 'first_name', 'last_name', 'email', 'password1', 'password2', 'is_employee', 'employer_select')*

model
alias of **User**

base_fields = *{'email': <django.forms.fields.CharField object>, 'employer_select': <django.forms.models.ModelChoiceField object>, 'first_name': <django.forms.fields.CharField object>, 'is_employee': <django.forms.fields.BooleanField object>, 'last_name': <django.forms.fields.CharField object>, 'password1': <django.forms.fields.CharField object>, 'password2': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

declared_fields = *{'email': <django.forms.fields.CharField object>, 'employer_select': <django.forms.models.ModelChoiceField object>, 'first_name': <django.forms.fields.CharField object>, 'is_employee': <django.forms.fields.BooleanField object>, 'last_name': <django.forms.fields.CharField object>, 'password1': <django.forms.fields.CharField object>, 'password2': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

property **media**
Return all media required to render the widgets on this form.

save (commit=True**)**
Saves the form data and associates the selected employer with the user instance.

Parameters:	**commit** (*bool, optional*) – Determines whether to save the user instance immediately to the database. Defaults to True.
Returns:	The saved user instance.
Return type:	User
Raises:	**None** –

class myapp.forms.**EmployerSignUpForm** (*args, **kwargs)
Bases: **UserCreationForm**
Form for employer sign up.
Inherits from UserCreationForm and adds additional fields such as first name, last name, and email.

class **Meta**
Bases: **object**

fields = *('username', 'first_name', 'last_name', 'email', 'password1', 'password2', 'is_employer')*

model
alias of **User**

base_fields = *{'email': <django.forms.fields.CharField object>, 'first_name': <django.forms.fields.CharField object>, 'is_employer': <django.forms.fields.BooleanField object>, 'last_name': <django.forms.fields.CharField object>, 'password1': <django.forms.fields.CharField object>, 'password2': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

declared_fields = *{'email': <django.forms.fields.CharField object>, 'first_name': <django.forms.fields.CharField object>, 'is_employer': <django.forms.fields.BooleanField object>, 'last_name': <django.forms.fields.CharField object>, 'password1': <django.forms.fields.CharField object>, 'password2': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

property **media**
Return all media required to render the widgets on this form.

class myapp.forms.**LoginForm** (data=None, files=None, auto_id='id_%s', prefix=None, initial=None, error_class=<class 'django.forms.utils.ErrorList'>, label_suffix=None, empty_permitted=False, field_order=None, use_required_attribute=None, renderer=None)
Bases: **Form**
Form for user login.

Provides fields for username and password entry.

base_fields = *{'password': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

declared_fields = *{'password': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

property **media**
Return all media required to render the widgets on this form.

class myapp.forms.**UserForm**(*args, **kwargs)
Bases: **ModelForm**
Form for updating user information.
Provides fields for updating the user's first name, last name, username, bio, and avatar.

class **Meta**
Bases: **object**

fields = *['first_name', 'last_name', 'username', 'bio', 'avatar']*

model
alias of **User**

base_fields = *{'avatar': <django.forms.fields.ImageField object>, 'bio': <django.forms.fields.CharField object>, 'first_name': <django.forms.fields.CharField object>, 'last_name': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

declared_fields = *{'avatar': <django.forms.fields.ImageField object>, 'bio': <django.forms.fields.CharField object>, 'first_name': <django.forms.fields.CharField object>, 'last_name': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

property **media**
Return all media required to render the widgets on this form.

myapp.models module

class myapp.models.**User**(*args, **kwargs)
Bases: **AbstractUser**
Custom User model that includes both employees and employers.

Fields:

is_employer (bool): Indicates whether the user is an employer. is_employee (bool): Indicates whether the user is an employee. is_email_verified (bool): Indicates whether the user's email is verified. my_employer (str): The employer associated with the user. first_name (str): The first name of the user. last_name (str): The last name of the user. bio (str): A text field for the user's bio. avatar (ImageField): An image field for the user's avatar.

None ()

exception **DoesNotExist**
Bases: **ObjectDoesNotExist**

exception **MultipleObjectsReturned**
Bases: **MultipleObjectsReturned**

avatar
Just like the FileDescriptor, but for ImageFields. The only difference is assigning the width/height to the width_field/height_field, if appropriate.

bio

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`course_set`

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`date_joined`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`email`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`first_name`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`get_next_by_date_joined` (*, `field=<django.db.models.fields.DateTimeField: date_joined>`, `is_next=True`, `**kwargs`)

`get_previous_by_date_joined` (*, `field=<django.db.models.fields.DateTimeField: date_joined>`, `is_next=False`, `**kwargs`)

`groups`

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`id`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`is_active`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`is_email_verified`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`is_employee`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`is_employer`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_staff

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

is_superuser

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

last_login

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

last_name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

logentry_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

message_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

my_employer

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

participants_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

password

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

result_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`user_permissions`
Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.
In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`username`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

myapp.tokens module

class `myapp.tokens.`**`AccountActivationTokenGenerator`**
Bases: **`PasswordResetTokenGenerator`**
Custom token generator for account activation.

`_make_hash_value` **(**`self`**,** `user`**,** `timestamp`**)**
Override method to generate a unique hash value based on user and timestamp.

course package

course.views module

course.views.announcement_page module

`course.views.announcement_page.`**`announcementPage`** (`request`, `pk`)
Render the announcement page for a specific course.
This view function retrieves the necessary data to display the announcement page for a particular course. It fetches the participant's enrolled courses, retrieves the course based on the provided primary key (pk), and retrieves the announcements associated with the course. The announcements are ordered by their 'updated' field in descending order.
Parameters: - request: The HTTP request object generated by the user's interaction with the web application. - pk: The primary key of the course for which the announcement page is being rendered.
Returns: - A rendered HTML template displaying the announcement page with the course details and associated announcements.

course.views.chat_room module

`course.views.chat_room.`**`chatRoom`** (`request`, `pk2`, `pk`)
Render the employee chat room page for a specific room.
This view function retrieves the necessary data to display the chat room page for a specific room in a course. It fetches the courses in which the employee is a participant, retrieves the specific course and room based on the provided primary keys (pk2 and pk), and fetches all messages associated with the room.
If the HTTP request method is POST, a new message is created and associated with the room. The page is then redirected back to the chat room with the updated message.
Parameters: - request: The HTTP request object generated by the user's interaction with the web application. - pk2: The primary key of the course to which the chat room belongs. - pk: The primary key of the chat room being accessed.

Returns: - If the request method is POST, the function redirects to the chat room page with the updated message. - Otherwise, it renders the chat room template with the necessary context data.

course.views.course_page module

`course.views.course_page.`**`all_events`**`(request, pk)`

Return JSON response with all events associated with a course.

This view function retrieves all events associated with a specific course and returns a JSON response containing information about each event. The response includes the event title, id, start date and time, and end date and time.

Parameters: - request: The HTTP request object generated by the user's interaction with the web application. - pk: The primary key of the course for which events are being fetched.

Returns: - A JSON response containing information about all events associated with the course.

`course.views.course_page.`**`coursePage`**`(request, pk)`

Render the course page for employees.

This view function retrieves the necessary data to display the course page for a specific course. It fetches the course resources, announcements, and rooms based on the provided primary key (pk). The function also retrieves the list of courses in which the employee is a participant.

Parameters: - request: The HTTP request object generated by the user's interaction with the web application. - pk: The primary key of the course being accessed.

Returns: - The function renders the course page template with the necessary context data.

course.views.profile_page module

`course.views.profile_page.`**`profile`**`(request, pk)`

Render the profile page for a user.

This view function retrieves the user profile data based on the provided primary key (pk) and renders the profile page template with the user profile as context.

Parameters: - request: The HTTP request object generated by the user's interaction with the web application. - pk: The primary key of the user whose profile is being accessed.

Returns: - The function renders the profile page template with the user profile as context.

course.views.quiz_page module

`course.views.quiz_page.`**`quizPage`**`(request, pk)`

Render the quiz page for a course.

This view function retrieves the necessary data to display the quiz page for a specific course. It fetches the courses in which the employee is a participant and retrieves the specific course based on the provided primary key (pk).

Parameters: - request: The HTTP request object generated by the user's interaction with the web application. - pk: The primary key of the course for which the quiz page is being accessed.

Returns: - The function renders the quiz page template with the necessary context data.

course.views.resource_page module

`course.views.resource_page.`**`downloadFile`**`(request, pk)`

Handle the file download action for a resource.

This view function handles the file download action for a specific resource. It increments the download count for both the user and the resource, and redirects to the resource page.

Parameters: - request: The HTTP request object generated by the user's interaction with the web application. - pk: The primary key of the resource being downloaded.

Returns: - The function redirects to the resource page after the download action is processed.

`course.views.resource_page.`**`previewPdf`**`(request, pk)`

Preview a PDF resource.

This view function handles the preview action for a PDF resource. It streams the PDF content in chunks and returns a StreamingHttpResponse with the appropriate content type and disposition.

Parameters: - request: The HTTP request object generated by the user's interaction with the web application. - pk: The primary key of the PDF resource being previewed.

Returns: - A StreamingHttpResponse object with the PDF content.

`course.views.resource_page.`**`resourcePage`**`(request, pk)`

Render the resource page for a course.

This view function retrieves the necessary data to display the resource page for a specific course. It fetches the participants of the course, performs a search query if specified, and retrieves the resources of different file types associated with the course.

Parameters: - request: The HTTP request object generated by the user’s interaction with the web application. - pk: The primary key of the course for which the resource page is being accessed.

Returns: - The function renders the resource page template with the necessary context data.

course.views.room_page module

`course.views.room_page.`**`roomPage`**`(request, pk)`

Render the room page for a course.

This view function retrieves the necessary data to display the room page for a specific course. It fetches the participants of the course, retrieves the rooms associated with the course, and prepares the context data for rendering the room page template.

Parameters: - request: The HTTP request object generated by the user’s interaction with the web application. - pk: The primary key of the course for which the room page is being accessed.

Returns: - The function renders the room page template with the necessary context data.

course.admin module

class `course.admin.`**`CourseAdmin`**`(model, admin_site)`

Bases: **`ModelAdmin`**

Admin class for Course model.

This admin class customizes the Course model’s representation in the Django admin interface. It defines the behavior and appearance of the Course model’s admin page.

`inlines` = *[<class 'course.admin.ParticipantsInline'>]*

property **`media`**

class `course.admin.`**`ParticipantsInline`**`(parent_model, admin_site)`

Bases: **`TabularInline`**

Inline admin class for Participants.

This inline admin class defines the tabular layout for the Participants model in the Django admin interface. It allows managing the participants of a course directly from the course’s admin page.

property **`media`**

`model`

alias of **`Participants`**

course.forms module

class `course.forms.`**`AnnouncementForm`**`(*args, **kwargs)`

Bases: **`ModelForm`**

Form class for creating or updating an Announcement.

This form is used for creating or updating an Announcement object. It inherits from Django’s ModelForm class and defines the Announcement model as the model to be used for the form. It includes fields such as ‘title’, ‘content’, and ‘course’. The ‘course’ field is customized to show only courses created by the employer (user).

class **`Meta`**

Bases: **`object`**

`fields` = *['title', 'content', 'course']*

`model`

alias of **`Announcement`**

base_fields = *{'content': <django.forms.fields.CharField object>, 'course': <django.forms.models.ModelChoiceField object>, 'title': <django.forms.fields.CharField object>}*

declared_fields = *{}*

property **media**
Return all media required to render the widgets on this form.

class `course.forms.`**CourseForm** (`data=None`, `files=None`, `auto_id='id_%s'`, `prefix=None`, `initial=None`, `error_class=<class 'django.forms.utils.ErrorList'>`, `label_suffix=None`, `empty_permitted=False`, `instance=None`, `use_required_attribute=None`, `renderer=None`)
Bases: **ModelForm**
Form class for creating or updating a Course.
This form is used for creating or updating a Course object. It inherits from Django’s ModelForm class and defines the Course model as the model to be used for the form. It includes all fields of the Course model except ‘course_participants’ and ‘instructor’.

class **Meta**
Bases: **object**

exclude = *['course_participants', 'instructor']*

fields = *'__all__'*

model
alias of **Course**

base_fields = *{'description': <django.forms.fields.CharField object>, 'name': <django.forms.fields.CharField object>}*

declared_fields = *{}*

property **media**
Return all media required to render the widgets on this form.

class `course.forms.`**ResourceForm** (`*args`, `**kwargs`)
Bases: **ModelForm**
Form class for creating or updating a Resource.
This form is used for creating or updating a Resource object. It inherits from Django’s ModelForm class and defines the Resource model as the model to be used for the form. It includes fields such as ‘name’, ‘course’, ‘description’, ‘youtubeLink’, and ‘file’. The ‘course’ field is customized to show only courses created by the employer (user).

class **Meta**
Bases: **object**

fields = *['name', 'course', 'description', 'youtubeLink', 'file']*

model
alias of **Resource**

base_fields = *{'course': <django.forms.models.ModelChoiceField object>, 'description': <django.forms.fields.CharField object>, 'file': <django.forms.fields.FileField object>, 'name': <django.forms.fields.CharField object>, 'youtubeLink': <django.forms.fields.CharField object>}*

declared_fields = *{}*

property **media**
Return all media required to render the widgets on this form.

class `course.forms.`**RoomForm** (`*args`, `**kwargs`)
Bases: **ModelForm**

Form class for creating or updating a Room.

This form is used for creating or updating a Room object. It inherits from Django’s ModelForm class and defines the Room model as the model to be used for the form. It includes fields such as ‘room_topic’, ‘course’, and ‘room_description’. The ‘course’ field is customized to show only courses created by the employer (user).

class **`Meta`**

Bases: **`object`**

`fields` = *['room_topic', 'course', 'room_description']*

`model`

alias of **`Room`**

`base_fields` = *{'course': <django.forms.models.ModelChoiceField object>, 'room_description': <django.forms.fields.CharField object>, 'room_topic': <django.forms.fields.CharField object>}*

`declared_fields` = *{}*

property **`media`**

Return all media required to render the widgets on this form.

course.models module

class `course.models.`**`Announcement`**(`*args`, `**kwargs`)

Bases: **`Model`**

Model representing an announcement associated with a course.

An announcement has a title, content, course (ForeignKey to Course), creation date, and last update date.

exception **`DoesNotExist`**

Bases: **`ObjectDoesNotExist`**

exception **`MultipleObjectsReturned`**

Bases: **`MultipleObjectsReturned`**

`content`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`course`

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

`course_id`

`created`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`date`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`get_next_by_created` **(**`*`, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=True`, `**kwargs`**)**

`get_next_by_date` (*, `field=<django.db.models.fields.DateTimeField: date>`, `is_next=True`, `**kwargs`)

`get_next_by_updated` (*, `field=<django.db.models.fields.DateTimeField: updated>`, `is_next=True`, `**kwargs`)

`get_previous_by_created` (*, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=False`, `**kwargs`)

`get_previous_by_date` (*, `field=<django.db.models.fields.DateTimeField: date>`, `is_next=False`, `**kwargs`)

`get_previous_by_updated` (*, `field=<django.db.models.fields.DateTimeField: updated>`, `is_next=False`, `**kwargs`)

`id`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`objects` = *<django.db.models.manager.Manager object>*

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

Override the save method to create an event for the announcement.

After saving the announcement, an event is created with the announcement's title, associated course, start time, and end time.

`title`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`updated`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class `course.models.`**`Course`** (`*args`, `**kwargs`)

Bases: **`Model`**

Model representing a course created by an employer.

A course has a name, description, instructor (ForeignKey to User), creation date, and last update date.

exception **`DoesNotExist`**

Bases: **`ObjectDoesNotExist`**

exception **`MultipleObjectsReturned`**

Bases: **`MultipleObjectsReturned`**

`announcement_set`

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`created`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`description`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

event_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

get_next_by_created **(***, `field=<django.db.models.fields.DateField: created>`, `is_next=True`, `**kwargs`**)**

get_next_by_updated **(***, `field=<django.db.models.fields.DateField: updated>`, `is_next=True`, `**kwargs`**)**

get_previous_by_created **(***, `field=<django.db.models.fields.DateField: created>`, `is_next=False`, `**kwargs`**)**

get_previous_by_updated **(***, `field=<django.db.models.fields.DateField: updated>`, `is_next=False`, `**kwargs`**)**

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

instructor

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

instructor_id

name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = *<django.db.models.manager.Manager object>*

participants_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

quiz_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`resource_set`
Accessor to the related objects manager on the reverse side of a many-to-one relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`room_set`
Accessor to the related objects manager on the reverse side of a many-to-one relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`updated`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class `course.models.`**`Message`** (`*args`, `**kwargs`)
Bases: **`Model`**
Model representing a message in a chat room.
A message has a user (ForeignKey to User), room (ForeignKey to Room), message body, creation date, and last update date.

exception **`DoesNotExist`**
Bases: **`ObjectDoesNotExist`**

exception **`MultipleObjectsReturned`**
Bases: **`MultipleObjectsReturned`**

`body`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`created`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`get_next_by_created` **(**`*`, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=True`, `**kwargs`**)**

`get_next_by_updated` **(**`*`, `field=<django.db.models.fields.DateTimeField: updated>`, `is_next=True`, `**kwargs`**)**

`get_previous_by_created` **(**`*`, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=False`, `**kwargs`**)**

get_previous_by_updated **(***, `field=<django.db.models.fields.DateTimeField: updated>`, `is_next=False`, `**kwargs`**)**

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = *<django.db.models.manager.Manager object>*

room

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

room_id

updated

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

user

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id

class `course.models.`**Participants**(`*args`, `**kwargs`)

Bases: **Model**

Model representing a participant in a course.

A participant has a user (ForeignKey to User), course (ForeignKey to Course), room (ForeignKey to Room), user download count, and user email sent count.

exception **DoesNotExist**

Bases: **ObjectDoesNotExist**

exception **MultipleObjectsReturned**

Bases: **MultipleObjectsReturned**

course

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

course_id

download_count ()

Increment the user download count by 1 and save the participant.

`email_count` **()**
Increment the user email sent count by 1 and save the participant.

`has_course` **()**
Check if the participant has an associated course.

Returns:	True if the participant has a course, False otherwise.
Return type:	bool

`id`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`objects` = *<django.db.models.manager.Manager object>*

`room`
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

`room_id`

`user`
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

`user_download_count`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`user_email_sent`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`user_id`

class `course.models.`**`Resource`** (`*args`, `**kwargs`)
Bases: **`Model`**
Model representing a resource associated with a course.
A resource has a name, course (ForeignKey to Course), description, file type, YouTube link, file upload, creation date, last update date, user download count, and user email sent count.

exception **`DoesNotExist`**
Bases: **`ObjectDoesNotExist`**

`FILE_TYPES` = *(('pdf', 'PDF'), ('image', 'Image'), ('audio', 'Audio'), ('video', 'Video'), ('link', 'link'))*

exception **`MultipleObjectsReturned`**
Bases: **`MultipleObjectsReturned`**

`course`

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

`course_id`

`created`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`description`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`download_count` **()**
Increment the user download count by 1 and save the resource.

`email_count` **()**
Increment the user email sent count by 1 and save the resource.

`file`
The descriptor for the file attribute on the model instance. Return a FieldFile when accessed so you can write code like:

```
>>> from myapp.models import MyModel
>>> instance = MyModel.objects.get(pk=1)
>>> instance.file.size
```

Assign a file object on assignment so you can do:

```
>>> with open('/path/to/hello.world') as f:
...     instance.file = File(f)
```

`file_type`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`get_file_type_display` **(**`*`, `field=<django.db.models.fields.CharField: file_type>`**)**

`get_next_by_created` **(**`*`, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=True`, `**kwargs`**)**

`get_next_by_updated` **(**`*`, `field=<django.db.models.fields.DateTimeField: updated>`, `is_next=True`, `**kwargs`**)**

`get_previous_by_created` **(**`*`, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=False`, `**kwargs`**)**

`get_previous_by_updated` **(**`*`, `field=<django.db.models.fields.DateTimeField: updated>`, `is_next=False`, `**kwargs`**)**

`id`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`name`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = *<django.db.models.manager.Manager object>*

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

Override the save method to create an event for the resource.

After saving the resource, an event is created with the resource’s name, associated course, start time, and end time.

updated

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

user_download_count

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

user_email_sent

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

youtubeLink

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class `course.models.`**Room** (`*args`, `**kwargs`)

Bases: **Model**

Model representing a chat room associated with a course.

A room has a room topic, course (ForeignKey to Course), room description, creation date, and last update date.

exception **DoesNotExist**

Bases: **ObjectDoesNotExist**

exception **MultipleObjectsReturned**

Bases: **MultipleObjectsReturned**

course

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

course_id

created

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_next_by_created **(**`*`, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=True`, `**kwargs`**)**

get_next_by_updated **(**`*`, `field=<django.db.models.fields.DateTimeField: updated>`, `is_next=True`, `**kwargs`**)**

get_previous_by_created **(**`*`, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=False`, `**kwargs`**)**

get_previous_by_updated (*, `field=<django.db.models.fields.DateTimeField: updated>`, `is_next=False`, `**kwargs`)

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

message_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

objects = *<django.db.models.manager.Manager object>*

participants_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

room_description
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

room_topic
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

updated
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`course.models.`**delete_associated_event** (`sender`, `instance`, `**kwargs`)
Signal receiver to delete the associated event when an announcement is deleted.
This signal receiver is triggered after deleting an announcement. It looks for an event with the same name as the deleted announcement and deletes it if found.

quiz package

quiz.views module

`quiz.views.`**quiz_data_view** (`request`, `pk`, `pk2`)
View function to retrieve quiz data (questions and answers) for AJAX requests.

Parameters:
- **request** – The HTTP request object.
- **pk** (*str*) – The primary key of the quiz.
- **pk2** (*str*) – The primary key of the course.

Returns: A JsonResponse containing the quiz data.

`quiz.views.`**`quiz_list_view`**`(request, pk2)`
View function to display all quizzes for a specific course.

Parameters:
- **request** – The HTTP request object.
- **pk2** (*str*) – The primary key of the course.

Returns: A rendered HTML template displaying the quizzes for the course.

`quiz.views.`**`quiz_view`**`(request, pk, pk2)`
View function to display a specific quiz.

Parameters:
- **request** – The HTTP request object.
- **pk** (*str*) – The primary key of the quiz.
- **pk2** (*str*) – The primary key of the course.

Returns: A rendered HTML template displaying the quiz.

`quiz.views.`**`save_quiz_view`**`(request, pk, pk2)`
View function to save the quiz responses submitted by the user.

Parameters:
- **request** – The HTTP request object.
- **pk** (*str*) – The primary key of the quiz.
- **pk2** (*str*) – The primary key of the course.

Returns: A JsonResponse indicating the quiz result.

quiz.admin module

class `quiz.admin.`**`AnswerInline`**`(parent_model, admin_site)`
Bases: **`TabularInline`**
Admin inline class for displaying answers in tabular format within a question.

property **`media`**

`model`
alias of **`Answer`**

class `quiz.admin.`**`QuestionAdmin`**`(model, admin_site)`
Bases: **`ModelAdmin`**
Admin model class for configuring the display of Question model in the admin interface.

`inlines` = *[<class 'quiz.admin.AnswerInline'>]*

property **`media`**

quiz.models module

class `quiz.models.`**`Answer`**`(*args, **kwargs)`
Bases: **`Model`**
Model representing an answer to a question.

exception **`DoesNotExist`**
Bases: **`ObjectDoesNotExist`**

exception **`MultipleObjectsReturned`**
Bases: **`MultipleObjectsReturned`**

`correct`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`created`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`get_next_by_created` **(**`*`**,** `field=<django.db.models.fields.DateTimeField: created>`**,** `is_next=True`**,** `**kwargs`**)**

`get_previous_by_created` **(**`*`**,** `field=<django.db.models.fields.DateTimeField: created>`**,** `is_next=False`**,** `**kwargs`**)**

`id`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`objects` = *<django.db.models.manager.Manager object>*

`question`

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

`question_id`

`text`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class `quiz.models.`**`Question`** (`*args`, `**kwargs`)

Bases: **`Model`**

Model representing a question in a quiz.

exception **`DoesNotExist`**

Bases: **`ObjectDoesNotExist`**

exception **`MultipleObjectsReturned`**

Bases: **`MultipleObjectsReturned`**

`answer_set`

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`created`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`get_answers` **()**

Returns a list of answers associated with the question.

get_next_by_created **(***, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=True`, `**kwargs`**)**

get_previous_by_created **(***, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=False`, `**kwargs`**)**

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = *<django.db.models.manager.Manager object>*

quiz

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

quiz_id

text

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class `quiz.models.`**Quiz** (`*args`, `**kwargs`)

Bases: **Model**

Model representing a quiz.

DIFF_CHOICES = *(('easy', 'easy'), ('medium', 'medium'), ('hard', 'hard'))*

exception **DoesNotExist**

Bases: **ObjectDoesNotExist**

exception **MultipleObjectsReturned**

Bases: **MultipleObjectsReturned**

course

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

course_id

difficluty

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

end_date

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_difficluty_display **(***, `field=<django.db.models.fields.CharField: difficluty>`**)**

`get_questions` **()**

Returns a list of questions associated with the quiz. Randomizes the order of questions.

`id`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`name`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`number_of_questions`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`objects` = *<django.db.models.manager.Manager object>*

`question_set`

Accessor to the related objects manager on the reverse side of a many-to-one relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`required_score_to_pass`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`result_set`

Accessor to the related objects manager on the reverse side of a many-to-one relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.
Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

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

Overrides the save method to create an associated event for the quiz.

`start_date`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`time`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class `quiz.models.`**`Result`** (`*args`, `**kwargs`)

Bases: **`Model`**

Model representing the result of a user in a quiz.

exception **`DoesNotExist`**

Bases: **`ObjectDoesNotExist`**

exception **`MultipleObjectsReturned`**

Bases: **`MultipleObjectsReturned`**

`completion_time`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`created`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`get_next_by_created` **(**`*`, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=True`, `**kwargs`**)**

`get_previous_by_created` **(**`*`, `field=<django.db.models.fields.DateTimeField: created>`, `is_next=False`, `**kwargs`**)**

`id`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`objects` = *<django.db.models.manager.Manager object>*

`quiz`
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

`quiz_id`

`score`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`started`
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`user`
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.
In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

`user_id`

`quiz.models.`**`delete_associated_event`** (`sender`, `instance`, `**kwargs`)
Deletes the associated event when a quiz is deleted.

event package

event.models module

class `event.models.`**`Event`**(`*args`, `**kwargs`)

Bases: **`Model`**

Model representing an event.

Fields:

id (AutoField): Primary key for the event. name (CharField): Name of the event (max length: 255 characters). course (ForeignKey): Foreign key to the 'Course' model, representing the course associated with the event. start (DateTimeField): Start date and time of the event. end (DateTimeField): End date and time of the event.

exception **`DoesNotExist`**

Bases: **`ObjectDoesNotExist`**

exception **`MultipleObjectsReturned`**

Bases: **`MultipleObjectsReturned`**

`course`

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

`course_id`

`end`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`id`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`name`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

`objects` = *<django.db.models.manager.Manager object>*

`start`

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

employerAdmin package

employerAdmin.admin module

class `employerAdmin.admin.`**`AnnouncementAdmin`**(`model`, `admin_site`)

Bases: **`ModelAdmin`**

`get_queryset`(`request`)

Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

`list_display` = *('title', 'course', 'date', 'created', 'updated')*

property **`media`**

class `employerAdmin.admin.`**`AnswerAdmin`**(`model`, `admin_site`)

Bases: **`ModelAdmin`**

`get_queryset` **(**`request`**)**
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

`list_display` = *('text', 'correct', 'question', 'created')*

property **`media`**

class `employerAdmin.admin.`**`AnswerInline`** (`parent_model`, `admin_site`)
Bases: **`TabularInline`**
Inline model admin for Answer.
Provides an inline editing interface for Answer model within QuestionAdmin.

property **`media`**

`model`
alias of **`Answer`**

class `employerAdmin.admin.`**`CourseAdmin`** (`model`, `admin_site`)
Bases: **`ModelAdmin`**

`get_queryset` **(**`request`**)**
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

`inlines` = *[<class 'employerAdmin.admin.ParticipantsInline'>]*

`list_display` = *('name', 'instructor', 'created', 'updated')*

property **`media`**

class `employerAdmin.admin.`**`EventAdmin`** (`model`, `admin_site`)
Bases: **`ModelAdmin`**

`get_queryset` **(**`request`**)**
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

`list_display` = *('name', 'course', 'start', 'end')*

property **`media`**

class `employerAdmin.admin.`**`ParticipantsAdmin`** (`model`, `admin_site`)
Bases: **`ModelAdmin`**

`get_queryset` **(**`request`**)**
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

`list_display` = *('user', 'course')*

property **`media`**

class `employerAdmin.admin.`**`ParticipantsInline`** (`parent_model`, `admin_site`)
Bases: **`TabularInline`**
Inline model admin for Participants.
Provides an inline editing interface for Participants model within QuestionAdmin.

property **`media`**

`model`
alias of **`Participants`**

class `employerAdmin.admin.`**`QuestionAdmin`** (`model`, `admin_site`)
Bases: **`ModelAdmin`**

get_queryset (request)
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

inlines = *[<class 'employerAdmin.admin.AnswerInline'>]*

list_display = *('text', 'quiz', 'created')*

property **media**

class employerAdmin.admin.**QuizAdmin** (model, admin_site)
Bases: **ModelAdmin**

get_queryset (request)
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

list_display = *('name', 'course', 'number_of_questions', 'time', 'required_score_to_pass', 'difficluty')*

property **media**

class employerAdmin.admin.**ResourceAdmin** (model, admin_site)
Bases: **ModelAdmin**

get_queryset (request)
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

list_display = *('name', 'course', 'created', 'updated')*

property **media**

class employerAdmin.admin.**ResultAdmin** (model, admin_site)
Bases: **ModelAdmin**

get_queryset (request)
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

list_display = *('quiz', 'user', 'score', 'completion_time', 'created', 'started')*

property **media**

class employerAdmin.admin.**RoomAdmin** (model, admin_site)
Bases: **ModelAdmin**

get_queryset (request)
Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

list_display = *('room_topic', 'course', 'created', 'updated')*

property **media**

employerAdmin.employer_admin module

class employerAdmin.employer_admin.**MyAdminSite** (name='admin')
Bases: **AdminSite**
Custom admin site for SkillBuilder employers.

Site_header:	The header displayed on the admin site. :noindex:
Site_title:	The title displayed on the admin site. :noindex:
Login_template:	The template used for the admin login page. :noindex:

login_template = *'admin/login.html'*

site_header = *'SkillBuilder Employer Administration'*

site_title = *'SkillBuilder Employer Administration'*

Indices and tables

- **genindex**
- **modindex**
- **search**

Index

_

_make_hash_value() (myapp.tokens.AccountActivationTokenGenerator method)

A

about() (in module myapp.views.front_page)

AccountActivationTokenGenerator (class in myapp.tokens)

activate_user() (in module myapp.views.authentication_page)

all_events() (in module course.views.course_page)

Announcement (class in course.models)

Announcement.DoesNotExist

Announcement.MultipleObjectsReturned

announcement_set (course.models.Course attribute)

AnnouncementAdmin (class in employerAdmin.admin)

AnnouncementForm (class in course.forms)

AnnouncementForm.Meta (class in course.forms)

announcementPage() (in module course.views.announcement_page)

Answer (class in quiz.models)

Answer.DoesNotExist

Answer.MultipleObjectsReturned

answer_set (quiz.models.Question attribute)

AnswerAdmin (class in employerAdmin.admin)

AnswerInline (class in employerAdmin.admin)
- (class in quiz.admin)

avatar (myapp.models.User attribute)

B

base_fields (course.forms.AnnouncementForm attribute)
- (course.forms.CourseForm attribute)
- (course.forms.ResourceForm attribute)
- (course.forms.RoomForm attribute)
- (myapp.forms.EmployeeSignUpForm attribute)
- (myapp.forms.EmployerSignUpForm attribute)
- (myapp.forms.LoginForm attribute)
- (myapp.forms.UserForm attribute)

bio (myapp.models.User attribute)

body (course.models.Message attribute)

C

chatRoom() (in module course.views.chat_room)

completion_time (quiz.models.Result attribute)

contact() (in module myapp.views.front_page)

content (course.models.Announcement attribute)

correct (quiz.models.Answer attribute)

Course (class in course.models)

course (course.models.Announcement attribute)
- (course.models.Participants attribute)
- (course.models.Resource attribute)
- (course.models.Room attribute)
- (event.models.Event attribute)
- (quiz.models.Quiz attribute)

course.admin
- module

Course.DoesNotExist

course.forms
- module

course.models
- module

Course.MultipleObjectsReturned

course.views.announcement_page
- module

course.views.chat_room
- module

course.views.course_page
- module

course.views.profile_page
- module

course.views.quiz_page
- module

course.views.resource_page
- module

course.views.room_page
- module

course_id (course.models.Announcement attribute)
- (course.models.Participants attribute)
- (course.models.Resource attribute)
- (course.models.Room attribute)
- (event.models.Event attribute)
- (quiz.models.Quiz attribute)

course_set (myapp.models.User attribute)

CourseAdmin (class in course.admin)
- (class in employerAdmin.admin)

CourseForm (class in course.forms)

CourseForm.Meta (class in course.forms)

coursePage() (in module course.views.course_page)

created (course.models.Announcement attribute)

    (course.models.Course attribute)

    (course.models.Message attribute)

    (course.models.Resource attribute)

    (course.models.Room attribute)

    (quiz.models.Answer attribute)

    (quiz.models.Question attribute)

    (quiz.models.Result attribute)

D

date (course.models.Announcement attribute)

date_joined (myapp.models.User attribute)

declared_fields (course.forms.AnnouncementForm attribute)

    (course.forms.CourseForm attribute)

    (course.forms.ResourceForm attribute)

    (course.forms.RoomForm attribute)

    (myapp.forms.EmployeeSignUpForm attribute)

    (myapp.forms.EmployerSignUpForm attribute)

    (myapp.forms.LoginForm attribute)

    (myapp.forms.UserForm attribute)

delete_associated_event() (in module course.models)

    (in module quiz.models)

description (course.models.Course attribute)

    (course.models.Resource attribute)

DIFF_CHOICES (quiz.models.Quiz attribute)

difficulty (quiz.models.Quiz attribute)

download_count() (course.models.Participants method)

    (course.models.Resource method)

downloadFile() (in module course.views.resource_page)

E

email (myapp.models.User attribute)

email_count() (course.models.Participants method)

    (course.models.Resource method)

employeeHome() (in module myapp.views.users_page)

EmployeeSignUpForm (class in myapp.forms)

EmployeeSignUpForm.Meta (class in myapp.forms)

employeeSignupPage() (in module myapp.views.authentication_page)

employerAdmin.admin

    module

employerAdmin.employer_admin

    module

EmployerSignUpForm (class in myapp.forms)

EmployerSignUpForm.Meta (class in myapp.forms)

employerSignupPage() (in module myapp.views.authentication_page)

end (event.models.Event attribute)

end_date (quiz.models.Quiz attribute)

Event (class in event.models)

Event.DoesNotExist

event.models

    module

Event.MultipleObjectsReturned

event_set (course.models.Course attribute)

EventAdmin (class in employerAdmin.admin)

exclude (course.forms.CourseForm.Meta attribute)

F

fields (course.forms.AnnouncementForm.Meta attribute)

    (course.forms.CourseForm.Meta attribute)

    (course.forms.ResourceForm.Meta attribute)

    (course.forms.RoomForm.Meta attribute)

    (myapp.forms.EmployeeSignUpForm.Meta attribute)

    (myapp.forms.EmployerSignUpForm.Meta attribute)

    (myapp.forms.UserForm.Meta attribute)

file (course.models.Resource attribute)

file_type (course.models.Resource attribute)

FILE_TYPES (course.models.Resource attribute)

first_name (myapp.models.User attribute)

G

get_answers() (quiz.models.Question method)

get_difficulty_display() (quiz.models.Quiz method)

get_file_type_display() (course.models.Resource method)

get_next_by_created() (course.models.Announcement method)

    (course.models.Course method)
    (course.models.Message method)

(course.models.Resource method)
(course.models.Room method)
(quiz.models.Answer method)
(quiz.models.Question method)
(quiz.models.Result method)
get_next_by_date() (course.models.Announcement method)
get_next_by_date_joined() (myapp.models.User method)
get_next_by_updated() (course.models.Announcement method)
(course.models.Course method)
(course.models.Message method)
(course.models.Resource method)
(course.models.Room method)
get_previous_by_created() (course.models.Announcement method)
(course.models.Course method)
(course.models.Message method)
(course.models.Resource method)
(course.models.Room method)
(quiz.models.Answer method)
(quiz.models.Question method)
(quiz.models.Result method)
get_previous_by_date() (course.models.Announcement method)
get_previous_by_date_joined() (myapp.models.User method)
get_previous_by_updated() (course.models.Announcement method)
(course.models.Course method)
(course.models.Message method)
(course.models.Resource method)
(course.models.Room method)
get_queryset() (employerAdmin.admin.AnnouncementAdmin method)
(employerAdmin.admin.AnswerAdmin method)
(employerAdmin.admin.CourseAdmin method)
(employerAdmin.admin.EventAdmin method)
(employerAdmin.admin.ParticipantsAdmin method)
(employerAdmin.admin.QuestionAdmin method)
(employerAdmin.admin.QuizAdmin method)
(employerAdmin.admin.ResourceAdmin method)
(employerAdmin.admin.ResultAdmin method)
(employerAdmin.admin.RoomAdmin method)
get_questions() (quiz.models.Quiz method)
groups (myapp.models.User attribute)

H

has_course() (course.models.Participants method)
home() (in module myapp.views.front_page)

I

id (course.models.Announcement attribute)
(course.models.Course attribute)
(course.models.Message attribute)
(course.models.Participants attribute)
(course.models.Resource attribute)
(course.models.Room attribute)
(event.models.Event attribute)
(myapp.models.User attribute)
(quiz.models.Answer attribute)
(quiz.models.Question attribute)
(quiz.models.Quiz attribute)
(quiz.models.Result attribute)
inlines (course.admin.CourseAdmin attribute)
(employerAdmin.admin.CourseAdmin attribute)
(employerAdmin.admin.QuestionAdmin attribute)
(quiz.admin.QuestionAdmin attribute)
instructor (course.models.Course attribute)
instructor_id (course.models.Course attribute)
is_active (myapp.models.User attribute)
is_email_verified (myapp.models.User attribute)
is_employee (myapp.models.User attribute)
is_employer (myapp.models.User attribute)
is_staff (myapp.models.User attribute)
is_superuser (myapp.models.User attribute)

L

last_login (myapp.models.User attribute)
last_name (myapp.models.User attribute)
list_display (employerAdmin.admin.AnnouncementAdmin attribute)
(employerAdmin.admin.AnswerAdmin attribute)
(employerAdmin.admin.CourseAdmin attribute)
(employerAdmin.admin.EventAdmin attribute)
(employerAdmin.admin.ParticipantsAdmin attribute)

(employerAdmin.admin.QuestionAdmin attribute)

(employerAdmin.admin.QuizAdmin attribute)

(employerAdmin.admin.ResourceAdmin attribute)

(employerAdmin.admin.ResultAdmin attribute)

(employerAdmin.admin.RoomAdmin attribute)

logentry_set (myapp.models.User attribute)

login_template (employerAdmin.employer_admin.MyAdminSite attribute)

LoginForm (class in myapp.forms)

loginPage() (in module myapp.views.authentication_page)

logoutUser() (in module myapp.views.authentication_page)

M

media (course.admin.CourseAdmin property)

(course.admin.ParticipantsInline property)

(course.forms.AnnouncementForm property)

(course.forms.CourseForm property)

(course.forms.ResourceForm property)

(course.forms.RoomForm property)

(employerAdmin.admin.AnnouncementAdmin property)

(employerAdmin.admin.AnswerAdmin property)

(employerAdmin.admin.AnswerInline property)

(employerAdmin.admin.CourseAdmin property)

(employerAdmin.admin.EventAdmin property)

(employerAdmin.admin.ParticipantsAdmin property)

(employerAdmin.admin.ParticipantsInline property)

(employerAdmin.admin.QuestionAdmin property)

(employerAdmin.admin.QuizAdmin property)

(employerAdmin.admin.ResourceAdmin property)

(employerAdmin.admin.ResultAdmin property)

(employerAdmin.admin.RoomAdmin property)

(myapp.forms.EmployeeSignUpForm property)

(myapp.forms.EmployerSignUpForm property)

(myapp.forms.LoginForm property)

(myapp.forms.UserForm property)

(quiz.admin.AnswerInline property)

(quiz.admin.QuestionAdmin property)

Message (class in course.models)

Message.DoesNotExist

Message.MultipleObjectsReturned

message_set (course.models.Room attribute)

(myapp.models.User attribute)

model (course.admin.ParticipantsInline attribute)

(course.forms.AnnouncementForm.Meta attribute)

(course.forms.CourseForm.Meta attribute)

(course.forms.ResourceForm.Meta attribute)

(course.forms.RoomForm.Meta attribute)

(employerAdmin.admin.AnswerInline attribute)

(employerAdmin.admin.ParticipantsInline attribute)

(myapp.forms.EmployeeSignUpForm.Meta attribute)

(myapp.forms.EmployerSignUpForm.Meta attribute)

(myapp.forms.UserForm.Meta attribute)

(quiz.admin.AnswerInline attribute)

module

course.admin

course.forms

course.models

course.views.announcement_page

course.views.chat_room

course.views.course_page

course.views.profile_page

course.views.quiz_page

course.views.resource_page

course.views.room_page

employerAdmin.admin

employerAdmin.employer_admin

event.models

myapp.forms

myapp.models

myapp.tokens

myapp.views.authentication_page

myapp.views.front_page

myapp.views.user_management

myapp.views.users_page

quiz.admin

quiz.models

quiz.views

my_employer (myapp.models.User attribute)

MyAdminSite (class in employerAdmin.employer_admin)

myapp.forms
module
myapp.models
module
myapp.tokens
module
myapp.views.authentication_page
module
myapp.views.front_page
module
myapp.views.user_management
module
myapp.views.users_page
module

N

name (course.models.Course attribute)
(course.models.Resource attribute)
(event.models.Event attribute)
(quiz.models.Quiz attribute)
None() (myapp.models.User method)
number_of_questions (quiz.models.Quiz attribute)

O

objects (course.models.Announcement attribute)
(course.models.Course attribute)
(course.models.Message attribute)
(course.models.Participants attribute)
(course.models.Resource attribute)
(course.models.Room attribute)
(event.models.Event attribute)
(quiz.models.Answer attribute)
(quiz.models.Question attribute)
(quiz.models.Quiz attribute)
(quiz.models.Result attribute)

P

Participants (class in course.models)
Participants.DoesNotExist
Participants.MultipleObjectsReturned
participants_set (course.models.Course attribute)
(course.models.Room attribute)
(myapp.models.User attribute)
ParticipantsAdmin (class in employerAdmin.admin)
ParticipantsInline (class in course.admin)
(class in employerAdmin.admin)
password (myapp.models.User attribute)
previewPdf() (in module course.views.resource_page)
profile() (in module course.views.profile_page)

Q

Question (class in quiz.models)
question (quiz.models.Answer attribute)
Question.DoesNotExist
Question.MultipleObjectsReturned
question_id (quiz.models.Answer attribute)
question_set (quiz.models.Quiz attribute)
QuestionAdmin (class in employerAdmin.admin)
(class in quiz.admin)
Quiz (class in quiz.models)
quiz (quiz.models.Question attribute)
(quiz.models.Result attribute)
quiz.admin
module
Quiz.DoesNotExist
quiz.models
module
Quiz.MultipleObjectsReturned
quiz.views
module
quiz_data_view() (in module quiz.views)
quiz_id (quiz.models.Question attribute)
(quiz.models.Result attribute)
quiz_list_view() (in module quiz.views)
quiz_set (course.models.Course attribute)
quiz_view() (in module quiz.views)
QuizAdmin (class in employerAdmin.admin)
quizPage() (in module course.views.quiz_page)

R

required_score_to_pass (quiz.models.Quiz attribute)
Resource (class in course.models)
Resource.DoesNotExist
Resource.MultipleObjectsReturned
resource_set (course.models.Course attribute)
ResourceAdmin (class in employerAdmin.admin)
ResourceForm (class in course.forms)
ResourceForm.Meta (class in course.forms)

resourcePage() (in module course.views.resource_page)

Result (class in quiz.models)

Result.DoesNotExist

Result.MultipleObjectsReturned

result_set (myapp.models.User attribute)

(quiz.models.Quiz attribute)

ResultAdmin (class in employerAdmin.admin)

Room (class in course.models)

room (course.models.Message attribute)

(course.models.Participants attribute)

Room.DoesNotExist

Room.MultipleObjectsReturned

room_description (course.models.Room attribute)

room_id (course.models.Message attribute)

(course.models.Participants attribute)

room_set (course.models.Course attribute)

room_topic (course.models.Room attribute)

RoomAdmin (class in employerAdmin.admin)

RoomForm (class in course.forms)

RoomForm.Meta (class in course.forms)

roomPage() (in module course.views.room_page)

S

save() (course.models.Announcement method)

(course.models.Resource method)

(myapp.forms.EmployeeSignUpForm method)

(quiz.models.Quiz method)

save_quiz_view() (in module quiz.views)

score (quiz.models.Result attribute)

send_activation_email() (in module myapp.views.authentication_page)

site_header (employerAdmin.employer_admin.MyAdminSite attribute)

site_title (employerAdmin.employer_admin.MyAdminSite attribute)

start (event.models.Event attribute)

start_date (quiz.models.Quiz attribute)

started (quiz.models.Result attribute)

T

text (quiz.models.Answer attribute)

(quiz.models.Question attribute)

time (quiz.models.Quiz attribute)

title (course.models.Announcement attribute)

U

updated (course.models.Announcement attribute)

(course.models.Course attribute)

(course.models.Message attribute)

(course.models.Resource attribute)

(course.models.Room attribute)

updateUser() (in module myapp.views.user_management)

User (class in myapp.models)

user (course.models.Message attribute)

(course.models.Participants attribute)

(quiz.models.Result attribute)

User.DoesNotExist

User.MultipleObjectsReturned

user_download_count (course.models.Participants attribute)

(course.models.Resource attribute)

user_email_sent (course.models.Participants attribute)

(course.models.Resource attribute)

user_id (course.models.Message attribute)

(course.models.Participants attribute)

(quiz.models.Result attribute)

user_permissions (myapp.models.User attribute)

UserForm (class in myapp.forms)

UserForm.Meta (class in myapp.forms)

username (myapp.models.User attribute)

Y

youtubeLink (course.models.Resource attribute)

Python Module Index

c

course
course.admin
course.forms
course.models
course.views.announcement_page
course.views.chat_room
course.views.course_page
course.views.profile_page
course.views.quiz_page
course.views.resource_page
course.views.room_page

e

employerAdmin
employerAdmin.admin
employerAdmin.employer_admin
event
event.models

m

myapp
myapp.forms
myapp.models
myapp.tokens
myapp.views.authentication_page
myapp.views.front_page
myapp.views.user_management
myapp.views.users_page

q

quiz
quiz.admin
quiz.models
quiz.views