Base form classes

Several form classes are provided with django-registration, covering common cases for gathering account information and implementing common constraints for user registration. These forms were designed with django-registration's built-in registration workflows in mind, but may also be useful in other situations.

class **`django_registration.forms.RegistrationForm`**

A form for registering an account. This is a subclass of Django's built-in `UserCreationForm`, and has the following fields, all of which are required:

username

The username to use for the new account.

email

The email address to use for the new account.

password1

The password to use for the new account.

password2

The password to use for the new account, repeated to catch typos.

Note

Validation of usernames

Django supplies a default regex-based validator for usernames in its base `AbstractBaseUser` implementation, allowing any word character along with the following set of additional characters: ., @, +, and -.

Because it's a subclass of Django's `UserCreationForm`, `RegistrationForm` will inherit the base validation defined by Django. It also applies some custom validators to the username: `ReservedNameValidator`, and `validate_confusables()`.

Note

Validation of email addresses

django-registration applies two additional validators – `HTML5EmailValidator` and `validate_confusables_email()` – to the email address.

The HTML5 validator uses the HTML5 email-validation rule (as implemented on HTML's *input type="email"*), which is more restrictive than the email RFCs. The purpose of this validator is twofold: to match the behavior of HTML5, and to simplify django-registration's other validators. The full RFC grammar for email addresses is enormously complex despite most of its features rarely if ever being used legitimately, so disallowing those features allows other validators to interact with a much simpler format, ensuring performance, reliability and safety.

Note

Custom user models

If you are using a custom user model, you **must** subclass this form and tell it to use your custom user model instead of Django's default user model. If you do not, django-registration will deliberately crash with an error message reminding you to do this. See the custom user compatibility guide for details.

---

class **`django_registration.forms.RegistrationFormCaseInsensitive`**

A subclass of `RegistrationForm` which enforces case-insensitive uniqueness of usernames, by applying `CaseInsensitiveUnique` to the username field.

Note

Unicode case handling

This form will normalize the username value to form NFKC, matching Django's default approach to Unicode normalization. it will then case fold the value, and use a case-insensitive (*iexact*) lookup to determine if a user with the same username already exists; the results of this query may depend on the quality of your database's Unicode implementation, and on configuration details. The results may also be surprising to developers who are primarily used to English/ASCII text, as Unicode's case rules can be quite complex.

---

class **`django_registration.forms.RegistrationFormTermsOfService`**

A subclass of `RegistrationForm` which adds one additional, required field:

tos

A checkbox indicating agreement to the site's terms of service/user agreement.

class **`django_registration.forms.RegistrationFormUniqueEmail`**

A subclass of `RegistrationForm` which enforces uniqueness of email addresses in addition to uniqueness of usernames, by applying `CaseInsensitiveUnique` to the email field.