# Développement web côté serveur

# Framework python-django

**Travaux pratique N°7**

**( Projet : Essentiels )**

**L’essentiel sur les formulaires**

**Objectifs :**

* **Créer un formulaire en utilisant la classe forms.Form**
* **Transformer l’apprance d’un formulaire en utilisant la propriété widget**
* **Expploiter les données daisies dans un formulaire**
* **Utiliser le jeton de sécurité de Django contre les attques CSRF**

**Travail à faire :**

|  |
| --- |
| **1-Créer le projet myessential puis créer l’application myform\_form dans le projet.** |
| **2- Ajouter l’application myform\_form au projet :**  **a-Ouvrir le fichier settings.py**  **b-ajouter au pramètre INSTALLED\_APP du fichier settings.py la ligne ci-dessous :**  INSTALLED\_APPS = [  'myform\_form.apps.MyformFormConfig',  …………………………………………………………………………………………. |
| **3- Créer les fichiers forms.py et urls.py, et le dossier templates dans le dossier de l’application myform\_form. Dans le dossier templates, créer le sous dossier myform\_form Voir figure cidessous :** |
|  |
|  |
|  |
| **3-Création d’un formulaire** |
| **Dans le fichier forms.py que vous avez créé à l'étape 3, créer le formulaire ci-dessous :** |
| **from django import forms** **JOBS = (  ('Python', 'Python Developper'),  ('JavaScript', 'JavaScript Developper'),  ('Java', 'Java Developper') ) CITIES=(  ('Tanger', 'Capitale du nord'),  ('Fes', 'Capitale culturelle'),  ('Rabat', 'Capitale administrative'),  ('Casablanca', 'Capitale economique') ) LANGUAGES=(  ('Ar', 'Arabic'),  ('En', 'English'),  ('sp', 'Spanish'),  ('Fr', 'French') )  class SignupForm(forms.Form):  pseudo = forms.CharField(max\_length=8, required=False)  email = forms.EmailField()  password = forms.CharField(min\_length=6)  jobs = forms.ChoiceField(choices=JOBS)  city = forms.MultipleChoiceField(choices=CITIES)  language = forms.MultipleChoiceField(choices=LANGUAGES)  cgu\_accept = forms.BooleanField(initial=True)** |
| **Remarquer que la création d’un formulaire est identique à la création d’un modèle. La classe doit hériter de la classe Form du module forms.**  **Chaque champ du formulaire est un objet instance d’une classe définie dans le module prédéfini forms.** |
|  |
| **Utilisation du shell du Django pour examiner le formulaire** |
| **(venv) E:\mydjango\myessential> python manage.py shell** |
|  |
| Importer la classe qui représente le formulaire |
|  |
|  |
| Instanciation de la classe |
|  |
|  |
| Affichage de l’objet form : Examiner bien ce code |
|  |
|  |
| <div>  <label for="id\_pseudo">Pseudo:</label>  <input type="text" name="pseudo" maxlength="8" id="id\_pseudo">  </div>  <div>  <label for="id\_email">Email:</label>  <input type="email" name="email" maxlength="320" required id="id\_email">  </div>  <div>  <label for="id\_password">Password:</label>  <input type="text" name="password" minlength="6" required id="id\_password">  </div>  <div>  <label for="id\_jobs">Jobs:</label>  <select name="jobs" id="id\_jobs">  <option value="Python">Python Developper</option>  <option value="JavaScript">JavaScript Developper</option>  <option value="Java">Java Developper</option>  </select>  </div>  …………..  <div>  <label for="id\_cgu\_accept">Cgu accept:</label>  <input type="checkbox" name="cgu\_accept" required id="id\_cgu\_accept" checked>  </div> |
| Remarquer que l’objet form contient le code HTML qui correspond et qui est généré par Django lors de l’instanciation de la classe SignupForm.  Notez les **ids** attribués automatiquements aux champs, ils sont tous de la forme **id\_<nom du champ>.**  Notez que tous les champs sont définis par défaut avec l’attribut **Required.** Pour le champ **pseudo,** nous avons défini **required à False** de façon explicite. |
|  |
| **Il est possible d’obtenir des codes HTML différents en Exécutant l’une des commandes ci-dessous/ Examinez chaque résultat.** |
|  |
| Le code généré correspond à celui d'un tableau sauf qu'il manque les balises <table> </table>. N'oubliez donc pas de les ajouter dans le template. |
| <tr>  <th><label for="id\_pseudo">Pseudo:</label></th>  <td><input type="text" name="pseudo" maxlength="8" id="id\_pseudo"> </td>  </tr>  <tr>  <th><label for="id\_email">Email:</label></th>  <td> <input type="email" name="email" maxlength="320" required id="id\_email"> </td>  </tr>  <tr>  <th><label for="id\_password">Password:</label></th>  <td> <input type="text" name="password" minlength="6" required id="id\_password"> </td>  </tr>  <tr>  <th><label for="id\_jobs">Jobs:</label></th>  <td>  <select name="jobs" id="id\_jobs">  <option value="Python">Python Developper</option>  <option value="JavaScript">JavaScript Developper</option>  <option value="Java">Java Developper</option>  </select>  </td>  </tr>  <tr>  <th><label for="id\_cgu\_accept">Cgu accept:</label></th>  <td> <input type="checkbox" name="cgu\_accept" required id="id\_cgu\_accept" checked> </td>  </tr> |
|  |
| <p>  <label for="id\_pseudo">Pseudo:</label>  <input type="text" name="pseudo" maxlength="8" id="id\_pseudo">  </p>  <p>  <label for="id\_email">Email:</label>  <input type="email" name="email" maxlength="320" required id="id\_email">  </p>  <p>  <label for="id\_password">Password:</label>  <input type="text" name="password" minlength="6" required id="id\_password">  </p>  <p>  <label for="id\_jobs">Jobs:</label>  <select name="jobs" id="id\_jobs">  <option value="Python">Python Developper</option>  <option value="JavaScript">JavaScript Developper</option>  <option value="Java">Java Developper</option>  </select>  </p>  <p>  <label for="id\_cgu\_accept">Cgu accept:</label>  <input type="checkbox" name="cgu\_accept" required id="id\_cgu\_accept" checked>  </p> |
|  |
| <li>  <label for="id\_pseudo">Pseudo:</label>  <input type="text" name="pseudo" maxlength="8" id="id\_pseudo">  </li>  <li>  <label for="id\_email">Email:</label>  <input type="email" name="email" maxlength="320" required id="id\_email">  </li>  <li>  <label for="id\_password">Password:</label>  <input type="text" name="password" minlength="6" required id="id\_password">  </li>  <li>  <label for="id\_jobs">Jobs:</label>  <select name="jobs" id="id\_jobs">  <option value="Python">Python Developper</option>  <option value="JavaScript">JavaScript Developper</option>  <option value="Java">Java Developper</option>  </select>  </li>  <li>  <label for="id\_cgu\_accept">Cgu accept:</label>  <input type="checkbox" name="cgu\_accept" required id="id\_cgu\_accept" checked>  </li> |
|  |
| **Le formulaire contient-il des données valides** |
|  |
| Puisque les champs sont définis par défaut avec l’attribut **Required**, et comme on n’a pas fourni de valeurs, les données ne sont pas valides. Afficher et examiner le code HTML généré. |
|  |
| **form = SignupForm({"email":"moha@gmail.com", "password":"1234567", "jobs":"Python", "city":"Tanger",   "language":"Ar","cgu\_accept":True})** |
| **>>> form = SignupForm(**  **{**  **"email" : "test@gmail.com",**  **"password" : "1234567",**  **"job" : "Python",**  **“city” : ”Tanger”,**  **“language” : ”Ar”**  **"cgu\_accept" : True**  **}**  **)** |
|  |
| <div>  <label for="id\_pseudo">Pseudo:</label>  <input type="text" name="pseudo" maxlength="8" id="id\_pseudo">  </div>  <div>  <label for="id\_email">Email:</label>  <input type="email" name="email" value="test@gmail.com" maxlength="320" required id="id\_email">  </div>  <div>  <label for="id\_password">Password:</label>  <input type="text" name="password" value="1234567" minlength="6" required id="id\_password">  </div>  <div>  <label for="id\_jobs">Jobs:</label>  <ul class="errorlist">  <li>This field is required.</li>  </ul>  <select name="jobs" aria-invalid="true" id="id\_jobs">  <option value="Python">Python Developper</option>  <option value="JavaScript">JavaScript Developper</option>  <option value="Java">Java Developper</option>  </select>  </div>  <div>  <label for="id\_cgu\_accept">Cgu accept:</label>  <input type="checkbox" name="cgu\_accept" required id="id\_cgu\_accept" checked>  </div> |
| **Le formulaire contient t-il des données valides** |
|  |
|  |
| **>>>cd = form.cleaned\_data**  **>>> print(cd)**  **{'pseudo': '', 'email': 'moha@gmail.com', 'password': '1234567', 'jobs': 'Python', 'city': 'Tanger', 'language': 'Ar', 'cgu\_accept': True}** |
|  |
|  |
| **Dans le formulaire créé ci-dessous, le field password a une valeur incorrecte** |
| **>>>** **form = SignupForm({"pseudo":"toto", "email":"test@gmail.com", "password":"123", “city”: “Tanger”, “language”: “Ar”, "jobs":"Python", "cgu\_accept":True})** |
|  |
| >>> print(form) |
| <div>  <label for="id\_pseudo">Pseudo:</label>  <input type="text" name="pseudo" value="toto" maxlength="8" id="id\_pseudo">  </div>  <div>  <label for="id\_email">Email:</label>  <input type="email" name="email" **value="test@gmail.com"** maxlength="320" required id="id\_email">  </div>  <div>  <label for="id\_password">Password:</label>  <ul class="errorlist"><li>Ensure this value has at least 6 characters (it has 3).</li></ul>  <input type="text" name="password" value="123" minlength="6" required aria-invalid="true" id="id\_password">  </div>  <div>  <label for="id\_jobs">Jobs:</label>  <select name="jobs" id="id\_jobs">  <option value="Python" **selected**>Python Developper</option>  <option value="JavaScript">JavaScript Developper</option>  <option value="Java">Java Developper</option>  </select>  </div>  <div>  <label for="id\_cgu\_accept">Cgu accept:</label>  <input type="checkbox" name="cgu\_accept" required id="id\_cgu\_accept" **checked**>  </div> |
|  |

|  |  |
| --- | --- |
| **Affichage du formulaire dans un template** | |
|  | |
| **Template :**  **E:/mydjango/myessential/myform\_form/templates/myform\_form/ Signup10\_affichage.html** | |
| **<!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <title>form: affichage</title> </head> <body> <table border="1">  <tr>  <th>Affichage par defaut</th>  <th>\*\*\*\*\*\*\*\*\*\*</th>  <th>Affichage as\_p</th>  <th>\*\*\*\*\*\*\*\*\*\*</th>  <th>Affichage as\_ul</th>  <th>\*\*\*\*\*\*\*\*\*\*</th>  <th>Affichage as\_table</th>  </tr>  <tr>  <td>  <form action="" method="post">  {{form}}  <p>  <input type="submit" value="Submit"/>  </p>  </form>  </td>  <td>\*\*\*\*\*\*\*\*\*\*</td>  <td>  <form action="" method="post">  {{form.as\_p}}  <p>  <input type="submit" value="Submit"/>  </p>  </form>  </td>  <td>\*\*\*\*\*\*\*\*\*\*</td>  <td>  <form action="" method="post">  {{form.as\_ul}}  <p>  <input type="submit" value="Submit"/>  </p>  </form>  </td>  <td>\*\*\*\*\*\*\*\*\*\*</td>  <td>  <form action="" method="post">  <table>  {{form.as\_table}}  <p>  <input type="submit" value="Submit"/>  </p>  </table>**  **</form>  </td>   </tr> </table>  </body> </html>** | |
|  | |
| **E:/mydjango/myessential/myform\_form/views.py** | |
| **from django.shortcuts import render from .forms import SignupForm *# Create your views here.* def signup10\_affichage(request):  form = SignupForm()  return render(request, "myform\_form/signup10\_affichage.html", context={"form" : form})** | |
|  | |
| **E:/mydjango/myessential/myform\_form/urls.py**  **N’oubliez pas de connecter le fichier urls.py de l’application myform\_form avec l’url.py du projet myessential en ajoutant au fichier urls.py du projet la ligne :**  **path('myform\_form/', include('myform\_form.urls')),** | |
| **from django.shortcuts import render from django.urls import path from . import views urlpatterns = [  path("signup10\_affichage", views.signup10\_affichage, name="signup10\_affichage"),   ]** | |
|  | |
| **Exécution: http://127.0.0.1:8000/myform\_form/signup10\_affichage/** | |
|  | |
| **Widgets**  **Il est possible de modifier le widget affiché sur le formulaire en ayant recours à l’attribut widget de chaque champ dans la classe qui définit le formulaire.**  **Examiner l’exemple ci-dessous :** | |
| **Myform\_form/forms.py** | |
| JOBS = (  ('Python', 'Python Developper'),  ('JavaScript', 'JavaScript Developper'),  ('Java', 'Java Developper') )  CITIES=(  ('Tanger', 'Capitale du nord'),  ('Fes', 'Capitale culturelle'),  ('Rabat', 'Capitale administrative'),  ('Casablanca', 'Capitale economique') )  LANGUAGES=(  ('Ar', 'Arabic'),  ('En', 'English'),  ('sp', 'Spanish'),  ('Fr', 'French') )  **class SignupFormWidget(forms.Form):  pseudo = forms.CharField(max\_length=8, required=False)  email = forms.EmailField()  password = forms.CharField(min\_length=6, widget=forms.PasswordInput)  jobs = forms.ChoiceField(choices=JOBS, widget=forms.RadioSelect)  city = forms.MultipleChoiceField(choices=CITIES, widget=forms.CheckboxSelectMultiple)  language = forms.MultipleChoiceField(choices=LANGUAGES, widget=forms.SelectMultiple)  cgu\_accept = forms.BooleanField(initial=True)** | |
| **myform\_form/templates/myform\_form/signup20\_widget.html** | |
| **<!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <title>Utilisation de l'attribut widget</title> </head> <body> <form action="" method="POST">  {% csrf\_token %}  <table>  {{form.as\_table}}  </table> </form>  </body> </html>** | |
|  | |
| **Passer au shell, importer la classe SignupFormWidget, créer une instance de la classe, puis imprimer l'instance et examiner le code HTML généré. Vous devez obtenir le code ci-dessous.** | |
|  | |
|  | |
|  | |
|  | |
| **<div>**  **<label for="id\_pseudo">Pseudo:</label>**  **<input type="text" name="pseudo" maxlength="8" id="id\_pseudo">**  **</div>**  **<div>**  **<label for="id\_email">Email:</label>**  **<input type="email" name="email" maxlength="320" required id="id\_email">**  **</div>**  **<div>**  **<label for="id\_password">Password:</label>**  **<input type="password" name="password" minlength="6" required id="id\_password">**  **</div>**  **<div>**  **<fieldset>**  **<legend>Jobs:</legend>**  **<div id="id\_jobs">**  **<div>**  **<label for="id\_jobs\_0"><input type="radio" name="jobs" value="Python"   required id="id\_jobs\_0"> Python Developper</label>**  **</div>**  **<div>**  **<label for="id\_jobs\_1"><input type="radio" name="jobs" value="JavaScript"   required id="id\_jobs\_1"> JavaScript Developper</label>**  **</div>**  **<div>**  **<label for="id\_jobs\_2"><input type="radio" name="jobs" value="Java"   required id="id\_jobs\_2"> Java Developper</label>**  **</div>**  **</div>**  **</fieldset>**  **</div>**  **<div>**  **<fieldset>**  **<legend>City:</legend>**  **<div id="id\_city">**  **<div>**  **<label for="id\_city\_0"><input type="checkbox" name="city" value="Tanger"   id="id\_city\_0"> Capitale du nord</label>**  **</div>**  **<div>**  **<label for="id\_city\_1"><input type="checkbox" name="city" value="Fes"   id="id\_city\_1"> Capitale culturelle</label>**  **</div>**  **<div>**  **<label for="id\_city\_2"><input type="checkbox" name="city"   value="Rabat" id="id\_city\_2"> Capitale administrative</label>**  **</div>**  **<div>**  **<label for="id\_city\_3"><input type="checkbox" name="city"   value="Casablanca" id="id\_city\_3"> Capitale economique</label>**  **</div>**  **</div>**  **</fieldset>**  **</div>**  **<div>**  **<label for="id\_language">Language:</label>**  **<select name="language" required id="id\_language" multiple>**  **<option value="Ar">Arabic</option>**  **<option value="En">English</option>**  **<option value="sp">Spanish</option>**  **<option value="Fr">French</option>**  **</select>**  **</div>**  **<div>**  **<label for="id\_cgu\_accept">Cgu accept:</label>**  **<input type="checkbox" name="cgu\_accept" required id="id\_cgu\_accept" checked>**  **</div>** | |
|  | |
| **views.py : myform\_form/signup20\_widget** | |
| **from .forms import SignupForm, SignupFormWidget**  **def signup20\_widget(request):  form = SignupFormWidget()  return render(request, "myform\_form/signup20\_widget.html",   context={"form" : form})** | |
|  | |
| **urls.py : ajouter la ligne ci-dessous au fichier : myform\_form/urls.py** | |
|  | |
| **Exécution : http://127.0.0.1:8000/myform\_form/signup20\_widget/** | |
|  | |
|  | |

|  |
| --- |
| **Récupérer les données d’un formulaire** |
| **Classe formulaire : myform\_form/forms/SignupFormData** |
| **class SignupFormData(forms.Form):  pseudo = forms.CharField(max\_length=8, required=False)  email = forms.EmailField()  password = forms.CharField(min\_length=6, widget=forms.PasswordInput)  jobs = forms.ChoiceField(choices=JOBS, widget=forms.RadioSelect)  language = forms.MultipleChoiceField(choices=LANGUAGES,   widget=forms.CheckboxSelectMultiple)  cgu\_accept = forms.BooleanField(initial=True)** |
| **Template : myform\_form/templates/myform\_form/signup30\_data.html** |
| **<!DOCTYPE html> <html lang="fr"> <head>  <meta charset="UTF-8">  <title>Récuperer les données d'un formulaire</title> </head> <body> <form>  <table>  {{form.as\_table}}  </table>  <input type="submit" value="s'inscrire"> </form> </body> </html>** |
|  |
| **views.py : Ajouter la view signup30\_data ci-dessous** |
| from .forms import SignupForm, SignupFormWidget, SignupFormData |
| **def signup30\_data(request):  form = SignupFormData()  return render(request, "myform\_form/signup30\_data.html", context={"form":form})** |
|  |
| **urls.py : Ajouter la ligne ci-dessous** |
| **path("signup30\_data/", views.signup30\_data, name="signup30\_data")** |
|  |
| **Exécution: http://127.0.0.1:8000/myform\_form/signup30\_data** |
|  |
|  |
| **Remplissons le formulaire par des données** |
|  |
|  |
| Puis cliquons sur le bouton « S’inscrire » |
|  |
| **Remarquons que le formulaire devient vide et que les données saisies sont envoyées dans l’URL.**  **La balise Form n’a pas spécifié la valeur de l’attribut Method donc Method=GET est prise par défaut.**  **De même la balise Form n’a pas spécifié la valeur de l’attribut** action **donc action = « «  est pris par défaut c’est-à-dire que le formulaire est traitée par la page courante, autrement dit la view qui a permis d’afficher le formulaire au départ.** |
|  |
| Modifions notre template pour ajouter l’atribut method = « POST » à la balise <FORM>. |
| **Template : myform\_form/templates/myform\_form/signup30\_data.html** |
| **<body> <form method="POST">  <table>  {{form.as\_table}}  </table>  <input type="submit" value="s'inscrire"> </form> </body>** |
|  |
| **Exécution: http://127.0.0.1:8000/myform\_form/signup30\_data** |
|  |
|  |
| Remplissez le formulaire comme ci-dessous puis cliquer sur le bouton « S’inscrire » |
|  |
| **Réponse du système** |
|  |
| **En cliquant sur le bouton « S’inscrire », le système envoie un message d’erreur qui indique que la vérification CSRF a échoué.**  **CSRF signifie « Cross Site Request Forgery = Contrefaçon de requête intersites » qui est une attaque qui falsifie la requête formulée par le formulaire.**  **Django possède un mécanisme pour neutraliser ce type d’attaque en utilisant un jeton unique qui va être envoyé avec le formulaire et qui va permettre d’authentifier que le formulaire a bien été envoyé par la bonne personne et n’a pas été modifié en chemin.**  **Pour ne pas avoir cette erreur et pour plus de sécurité lors de l’envoie du formulaire il faut ajouter à chaque formulaire la balise {% csrf\_token %} comme indiqué dans l’exemple qui suit** |
|  |
| **Ajoutons la balise** **{% csrf\_token %} au formulaire** |
|  |
| **Template : signup30\_data.html** |
|  |
|  |
| **Exécution : http://127.0.0.1:8000/myform\_form/signup30\_data/** |
|  |
|  |
| **Affichons le code source de la page HTML affichée** |
|  |
|  |
| Remplissons le formulaire puis cliquons sur le bouton « s’inscrire » |
| Remplissons le formulaire puis cliquons sur le bouton « s’inscrire » |
|  |
| **Nous remarquons que le formulaire est correctement soumis et qu’il n’ya pas d’erreur. Les données n’apparaissent pas dans la barre d’URL.** |
|  |
|  |

|  |
| --- |
| **Maintenant, comment faire pour récupérer les données envoyées par le formulaire** |
|  |
| **views.py : Modifier la view signup30\_data comme indiqué ci-dessous :** |
| **def signup30\_data(request):  form = SignupFormData()**  ***#------------------------------  #Si le formulaire est affiché pour la 1ère fois request.method='GET'  #Si le formulaire a été rempli et envoyé request.method='POST'  # ------------------------------* if request.method == "POST":  print(request.POST)  *# ------------------------------  #Vérifier la sortie dans la fen^tre du serveur  # ------------------------------* return render(request, "myform\_form/signup30\_data.html", context={"form":form})** |
|  |
| **Exécution : http://127.0.0.1:8000/myform\_form/signup30\_data/** |
|  |
|  |
| Remplissons le formulaire puis cliquons sur le bouton **»s’inscrire**» |
|  |
|  |
| **Vérifions la fenêtre du serveur.**  **Nous devons trouver les données affiché sous forme du dictionnaire** |
| **<QueryDict: {'csrfmiddlewaretoken': ['envK1IpRueEPyVMfujHjm425S2eZkUbcDArpPHtr3uNj2mmzmCLmIyp1mrE**  **iovWQ'], 'pseudo': ['moha'], 'email': ['moha@gmail.com'], 'password': ['13456'], 'jobs': ['Python'], 'language': ['Ar', 'En'], 'cgu\_accept': ['on']}>** |
|  |
| **Remarquer que chaque champ du formulaire est une clé associée à sa valeur qui est une liste dans le dictionnaire** **request.POST** |
| Ces données ne sont pas forcémént dans le bon format, par exemple le champ **cgu\_accept** qui est de type boolean a la valeur **on. C’est cette valeur que nous retrouverons au niveau du HTML.** |

|  |
| --- |
| **Mise en forme des données du formulaire** |
| **Pour mettre les données en format et les convertir vers le type du champ du formulaire, Django fournit le moyen présenté dans la vue ci-dessous (Méthode is\_valid() et attribut cleaned\_data.**  **L’attribut cleaned\_data ne doit être appelé qu’après avoir exécuté is\_valid()** |
|  |
| **views.py : Modifier la view signup30\_data comme ci-dessous** |
| **def signup30\_data(request):  if request.method == "POST":  *#Créer le formulaire en le remplissant par les données envoyées* form = SignupFormData(request.POST)  *#Tester si les données sont valides* if form.is\_valid():  *# Si oui obtenir les données nétoyées* cd = form.cleaned\_data  print(cd)  *#Vérifier la sortie dans la fenêtre du serveur* else:  form = SignupFormData()   return render(request, "myform\_form/signup30\_data.html", context={"form":form})** |
|  |
| **Exécution : http://127.0.0.1:8000/myform\_form/signup30\_data/** |
|  |
|  |
| **Remplissons le formulaire avec des données valides comme ci-dessous** |
|  |
|  |
| **En soumettant le formulaire, celui-ci-garde les données puisque la view lui retourne le formulaire rempli de données à part le mot de passe. Voir figure ci-dessous.** |
|  |
|  |
| **En examinant la fenêtre du serveur, nous trouvons la sortie de l’instruction**  **if form.is\_valid():  *# Si oui obtenir les données nétoyées* cd = form.cleaned\_data  print(cd)** |
| **Résultat dans la fenêtre du serveur** |
| **{'pseudo': 'moha', 'email': 'moha@gmail.com', 'password': '1234567', 'jobs': 'Python', 'language': ['Ar', 'En'], 'cgu\_accept': True}** |
| **Nous constatons que la valeur « on « associé à cgu\_accept est converti en True.** |
|  |
| **Saisissons maintenant un password incorrect** |
|  |
|  |
| **En cliquant sur le bouton « s’inscrire », le formulaire est retourné avec les anciènnes valeur sauf que le password est effacé ce qui est toujours le cas avec un password qui est précédé d’un message d’erreur** |
|  |
|  |
| **Losque les données sont valides on peut retourner un message avec HttpResponse ou utiliser HttpResponseRedirect pour afficher un template ou rediriger vers une autre url.** |
|  |
| **Envoi d’un message avec HttpResponse** |
|  |
| **views.py : Modifier la view signup30\_data comme ci-dessous :** |
| **from django.http import HttpResponse** |
| **def signup30\_data(request):  if request.method == "POST":  *#Créer le formulaire en le remplissant par les données envoyées* form = SignupFormData(request.POST)  *#Tester si les données sont valides* if form.is\_valid():  *# Si oui obtenir les données nétoyées* cd = form.cleaned\_data  return HttpResponse("<h1>Nous avons bien reçu votre inscription</h1>")  else:  form = SignupFormData()   return render(request, "myform\_form/signup30\_data.html", context={"form":form})** |
|  |
| **Exécution : http://127.0.0.1:8000/myform\_form/signup30\_data/** |
|  |
|  |
| En saisissant des valeurs valides, puis en cliquant sur le bouton « s’inscrire » , nous obtenons : |
|  |
|  |
| **Redirection vers une autre URL avec HttpResponseRedirect** |
|  |
| **Créer le Template : myform\_form/signup30\_reussi.html** |
|  |
|  |
| Créer la view : myform\_form/views/**signup30\_reussi** |
| **def signup30\_reussi(request):  return render(request, 'myform\_form/signup30\_reussi.html')** |
|  |
| **Ajouter au fichier myform\_form/urls.py la ligne ci-dessous** |
| **path("signup30\_reussi/", views.signup30\_reussi, name="signup30\_reussi"),** |
|  |
| **views.py : modifier la view signup30\_data comme ci-dessous :** |
| **from django.http import HttpResponseRedirect** |
| **def signup30\_data(request):  if request.method == "POST":  *#Créer le formulaire en le remplissant par les données envoyées* form = SignupFormData(request.POST)  *#Tester si les données sont valides* if form.is\_valid():  *# Si oui obtenir les données nétoyées* cd = form.cleaned\_data  *#--------------------------------------------------------------------* return HttpResponseRedirect("/myform\_form/signup30\_reussi/")  *# --------------------------------------------------------------------* else:  form = SignupFormData()   return render(request, "myform\_form/signup30\_data.html", context={"form":form})** |
|  |
|  |
| **Exécution : http://127.0.0.1:8000/myform\_form/signup30\_data/** |
|  |
| Remplir le formulaire avec des données valides |
|  |
|  |
| En cliquant sur le bouton s’inscrire, nous obtenons le template ci-dessous |
|  |
|  |
|  |
|  |
| **Redirection vers une autre URL avec redirect et reverse** |
| **Modifier la view signup30\_data comme ci-dessous** |
| **from django.http import HttpResponseRedirect, HttpResponse from django.shortcuts import render, redirect from django.urls import reverse**  **def signup30\_data(request):  if request.method == "POST":  *#Créer le formulaire en le remplissant par les données envoyées* form = SignupFormData(request.POST)  *#Tester si les données sont valides* if form.is\_valid():  *# Si oui obtenir les données nétoyées* cd = form.cleaned\_data  *#--------------------------------------------------------------------* return redirect (reverse("signup30\_reussi"))  *# --------------------------------------------------------------------* else:  form = SignupFormData()   return render(request, "myform\_form/signup30\_data.html", context={"form":form})** |
|  |
| **Exécution : http://127.0.0.1:8000/myform\_form/signup30\_data/** |
|  |
|  |
| **Remplir le formulaire avec des données valides** |
|  |
|  |
| **En cliquant sur le bouton s’inscrire, nous obtenons le template ci-dessous** |
|  |
|  |

|  |
| --- |
| **Affichage des données saisies dans un formulaire dans un template** |
| **Créer le template : signup31\_data.html ci-dessous :** |
| **<!DOCTYPE html> <html lang="fr"> <head>  <meta charset="UTF-8">  <title>Récuperer et afficher les données d'un formulaire</title> </head> <body> <form action="{%url 'signup31\_affichage' %}"  method="POST">  {% csrf\_token %}  <table>  {{form.as\_table}}  </table>  <input type="submit" value="S'inscrire"> </form> </body> </html>** |
|  |
| **Créer le template : signup31\_affichage.html (Affiche les données saisies dans le formulaire)** |
| **<!DOCTYPE html> <html lang="en"> <head>  <meta charset="UTF-8">  <title>Affichage des données saisies</title> </head> <body> <h1>Données saisies</h1> <table border="1">  <tr>  <td>pseudo : </td>  <td>{{cd.pseudo}}</td>  </tr>  <tr>  <td>Email : </td>  <td>{{cd.email}}</td>  </tr>  <tr>  <td>Password : </td>  <td>{{cd.password}}</td>  </tr>  <tr>  <td>Job</td>  <td>{{cd.job}}</td>  </tr>**  **<tr>  <td>City</td>  <td>{{cd.city}}</td>  </tr>**  **<tr>  <td>Languages</td>  <td>{{cd.language}}</td>  </tr> </table> <hr> <h2>Obtention automatique des clés et valeurs</h2> <table>  {% for cle, valeur in cd.items %}  <tr>  <td>{{cle}} : </td>  <td>{{valeur}}</td>  </tr>**  **{% endfor %} </table> </body> </html>** |
|  |
| **views.py : Créer la view signup31\_data ci-dessous** |
| **def signup31\_data(request):  form = SignupFormData()  return render(request, "myform\_form/signup31\_data.html", context={"form" : form})** |
|  |
| **views.py : Créer la view signup31\_affichage ci-dessous** |
| **def signup31\_affichage(request):  form = SignupFormData(request.POST)  if form.is\_valid():  cd = form.cleaned\_data  print(cd)  return render(request, 'myform\_form/signup31\_affichage.html', {"cd":cd})  else:  return HttpResponse("Les données ne sont pas valides")** |
|  |
| **urls.py : Ajouter les deux lignes ci-dessous** |
| **path("signup31\_data/", views.signup31\_data, name="signup31\_data"), path("signup31\_affichage/", views.signup31\_affichage, name="signup31\_affichage"),** |
|  |
| **Exécution : http://127.0.0.1:8000/myform\_form/signup31\_data/** |
|  |
| **Remplir le formulaire avec des données correctes puis cliquer sur le bouton « S’inscrire »** |
|  |
| **L’application lit les données saisies dans le formulaire puis les affiche dans le template signup31\_affichage.html** |
|  |
|  |

**Validation personnalisé d’un formulaire**

Jusqu’à maintenant les seules vérifications qui sont faites ont été effectuées du côté client avec HTML. (required, min\_length, max\_length, …).

Il arrive que nous ayons besoin d’effectuer des vérifications du côté serveur. Par exemple on peut vouloir vérifier que le pseudo a un certain format.

Pour effectuer des vérifications côté serveur, nous devons ajouter des méthodes de vérification à notre classe. Ces méthodes sont exécutées automatiquement lorsqu’on appelle la méthode is\_valid(). Ces méthodes doivent avoir obligatoirement le nom **clean\_<Nom du champ>.**

|  |
| --- |
| Modifions notre classe **SignupFormData** pour lui ajouter la méthode **clean\_pseudo** pour vérifier la valeur du champ pseudo du côté serveur. Nous voulons que le pseudo ne contienne pas le caractère $ (remarquer qu’il est possible de faire cette vérification avec JavaScript du côté client, mais c’est juste pour montrer un exemple). |
| **class SignupFormData(forms.Form):  pseudo = forms.CharField(max\_length=8, required=False)  email = forms.EmailField()  password = forms.CharField(min\_length=6, widget=forms.PasswordInput)  jobs = forms.ChoiceField(choices=JOBS, widget=forms.RadioSelect)  language = forms.MultipleChoiceField(choices=LANGUAGES, widget=forms.CheckboxSelectMultiple)  cgu\_accept = forms.BooleanField(initial=True)   def clean\_pseudo(self):  pseudo = self.cleaned\_data.get("pseudo")  if '$' in pseudo:  raise forms.ValidationError("Le pseudo ne doit pas contenir le $")  return pseudo** |
| **Lançons la requête :** [**https://127.0.0.1:8000/signup\_data5**](https://127.0.0.1:8000/signup_data5)  **Et saisissons un $ dans le pseudo** |
|  |
| **Django répond par** |
|  |
|  |

Partie II – Création d’un formulaire avec HtML

Créer l’application myform\_html dans le projet myessential.

Ajouter l’application au projet

Faites le nécéssaire pour saisir et exécuter les codes ci-dessous

|  |
| --- |
| Structure de l’application myform\_html |
|  |
|  |
|  |
|  |
|  |
| Les fichiers templates de l’application myform\_html |
| **Base.html** |
| <html lang="en"> <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  *<!-- CSS only -->* <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous"> *<!-- JavaScript Bundle with Popper -->* <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>   <title>Document</title> </head> <body> |
|  |
| **Add.html** |
| {% extends "myform\_html/base.html" %}  {% block content%} <div class='container'>  <h1>ADD.html</h1>  <h2>Add a new car to the database:</h2>  <form action="" method='POST'>  {% csrf\_token %}  <div class="form-group">  <label for="brand">Brand:</label>  <input class="form-control" type="text" id='brand' name='brand'>  </div>  <div class="form-group">  <label for="year">Year:</label>  <input class="form-control" type="text" is='year' name='year'>  </div>  <input class='btn btn-primary' type="submit">  </form>  </div>  {% endblock%} |
|  |
| **List.html** |
| {% extends "myform\_html/base.html" %}  {% block content%} <div class='container'>  <h1>LIST.HTML</h1>  <ul>  {% for car in all\_cars%}  <li>{{car}}</li>  {% endfor %}  </ul> </div>  {% endblock%} |
|  |
| **delete.html** |
| {% extends "myform\_html/base.html" %}  {% block content%} <div class='container'>  <h1>DELETE.HTML</h1> <h1>Enter pk to delete car!</h1>  <form action="" method='POST'>   {% csrf\_token %}  <div class='form-group'>  <label for="pk">PK Number:</label>  <input class='form-control' type="text" id='pk' name='pk'>  </div>  <input class='btn btn-primary' type="submit">   </form>  </div>  {% endblock%} |