SoftwareEngineering

Assignment (7)

Created By :
Dheya Al-Hadhrami

Supervisor :
Eng. Malek Al-Mossanif

Introduction

(Django)

سنغوص أولاً في عالم `QuerySet` لنكتشف كي

يمكنك سحب البيانات من قاعدة البيانات بذكاء وكفاءة،

متجنبين الأخطاء الشائعة التي تبطئ أداء الموقع.

ثم ننتقل إلى الجزء الثاني لنتناول نماذج الإدخال (`Forms`)،

حيث سنقارن بين بناء النماذج يدوياً والاعتماد على `ModelForm`،

Assignment Objectives

.Django Authentication Systems -

.Assestian Libaraties -

Part I:

Django Authentication Systems

JWT | Token-based | Session-based

1. المصادقة القائمة على الجلسات (Session-based)

هي الطريقة التقليدية والمدمجة في Django. عندما يسجل المستخدم دخوله، يُنشئ الخادم معرف جلسة (Session ID) فريد، يخزنه في قاعدة البيانات، ويرسله للعميل عبر كوكي. في كل طلب لاحق، يُرسل الكوكي ويستخدمه الخادم للتحقق من هوية المستخدم.

المميزات ✅

- بسيطة وسهلة التنفيذ (مدمجة في Django).
- آمنة ضد هجمات XSS لأن الكوكي يكون `httpOnly`.
- التحكم الكامل في الجلسة من جانب الخادم (مثل تسجيل الخروج القسري).

العيوب ❌

- تتطلب تخزينًا على الخادم لكل مستخدم نشط (Stateful).
- صعبة التوسع في البيئات الموزعة.
- عرضة لهجمات CSRF إذا لم يتم تأمينها بشكل صحيح (يتطلب `CsrfViewMiddleware`).

اعتبارات أمنية وحالات استخدام

- **الحماية من CSRF:** تأكد دائمًا من تفعيل `CsrfViewMiddleware` واستخدام `{% csrf_token %}` في النماذج.
- **أمان الكوكي:** في بيئة الإنتاج، اضبط `SESSION_COOKIE_SECURE=True` لإرسال الكوكي عبر HTTPS فقط.
- **حالة الاستخدام المثالية:** التطبيقات المتجانسة (Monolithic) التقليدية التي يقدم فيها Django الواجهات الأمامية والخلفية معًا.

2. المصادقة القائمة على التوكن (Token-based)

تُستخدم هذه الطريقة مع Django Rest Framework عبر `authtoken`. عند تسجيل الدخول، يُنشئ الخادم رمزًا فريدًا، يربطه بالمستخدم في قاعدة البيانات، ثم يرسله إلى العميل. يقوم العميل بتخزينه (عادة في Local Storage) وإرساله مع كل طلب في ترويسة `Authorization: Token `.

المميزات ✅

- عديمة الحالة (Stateless)، لا يحتاج الخادم لتخزين معلومات الجلسة.
- ممتازة للتطبيقات أحادية الصفحة (SPAs) وتطبيقات الموبايل.
- سهلة التوسع أفقيًا عبر عدة خوادم.

العيوب ❌

- يتطلب استعلامًا لقاعدة البيانات للتحقق من التوكن في كل طلب.
- تخزين التوكن في Local Storage يجعله عرضة لهجمات XSS.
- إبطال التوكن يتطلب حذفه من قاعدة البيانات.

اعتبارات أمنية وحالات استخدام

- **الحماية من XSS:** كن حذرًا عند تخزين التوكن في `localStorage`. الخيار الأكثر أمانًا هو تخزينه في كوكي `httpOnly`.
- **HTTPS إلزامي:** يجب دائمًا استخدام HTTPS لمنع سرقة التوكن أثناء النقل (Man-in-the-middle attack).
- **حالة الاستخدام المثالية:** واجهات برمجة تطبيقات (APIs) بسيطة تخدم تطبيقات الموبايل أو SPAs حيث لا تكون هناك حاجة ماسة لإبطال التوكن من جانب الخادم.

3. JSON Web Tokens (JWT)

JWT هو توكن موقّع رقميًا يحتوي على بيانات (claims) بتنسيق JSON. عند تسجيل الدخول، يُصدر الخادم JWT ويُرسله للعميل. يمكن للخادم التحقق من صحة التوقيع دون الحاجة لقاعدة البيانات، مما يعزز الأداء بشكل كبير.

المميزات ✅

- عديمة الحالة تمامًا (Truly Stateless): لا حاجة لقاعدة البيانات للتحقق.
- أداء عالٍ بسبب عدم وجود استعلامات قاعدة بيانات.
- تحتوي على بيانات مدمجة (مثل صلاحيات المستخدم).

العيوب ❌

- بمجرد إصداره، لا يمكن إبطاله قبل انتهاء صلاحيته (مشكلة كبيرة).
- تخزينه في Local Storage يجعله عرضة لهجمات XSS.
- أكثر تعقيدًا في التنفيذ مقارنة بالطرق الأخرى.

اعتبارات أمنية وحالات استخدام

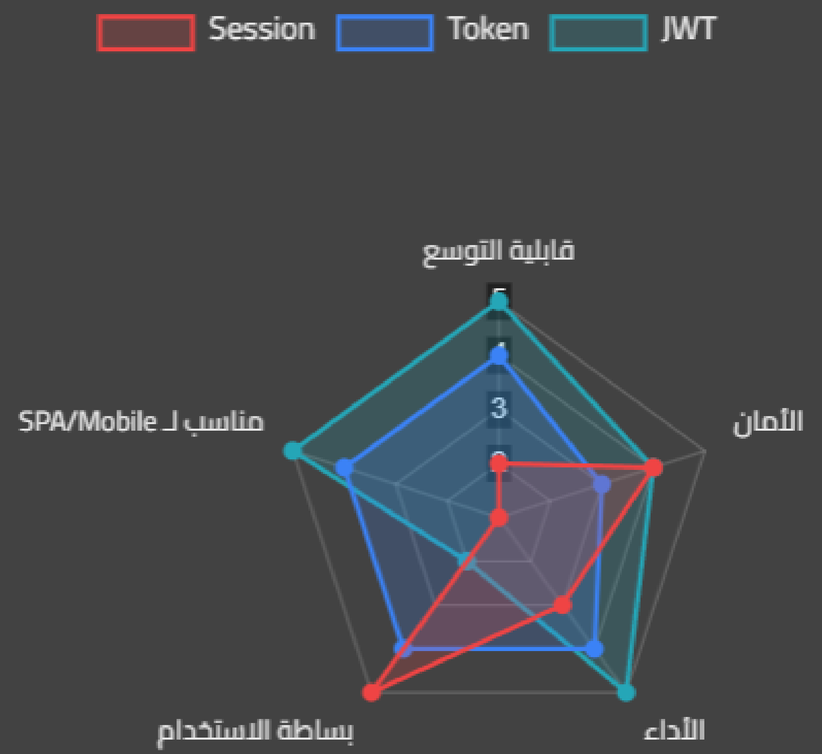
- **مشكلة الإبطال:** الحل الشائع هو استخدام توكن وصول قصير العمر (Access Token) وتوكن تحديث طويل العمر (Refresh Token)، مع الاحتفاظ بقائمة سوداء (Blocklist) للتوكنات المبطلة.
- **سرية المفتاح:** يجب الحفاظ على سرية المفتاح الخاص بالتوقيع (`SECRET_KEY`) بأي ثمن.
- **حالة الاستخدام المثالية:** الخدمات المصغرة (Microservices)، تطبيقات تتطلب مصادقة مع أطراف ثالثة، أو أنظمة معقدة تحتاج لبيانات الصلاحيات داخل التوكن نفسه.

مقارنة شاملة بين انظمة المصادقة

جدول المقارنة التفصيلي

الميزة	Session	Token	JWT
الحالة (State)	Stateful	Stateless	Stateless
قابلية التوسع	منخفضة	عالية	عالية جدًا
الأداء	متوسط	جيد	ممتاز
الإبطال (Revocation)	سهل	سهل	صعب
أمان CSRF	يتطلب حماية	آمن افتراضيًا	آمن افتراضيًا
أمان XSS	آمن نسبيًا	يتطلب حماية	يتطلب حماية

مقارنة الخصائص الرئيسية



Part II:

Django Widget Tweaks

هذه المكتبة تمنحك القدرة على تعديل خصائص

حقول النماذج مباشرة من القالب (Template) دون الحاجة للمس ملف `forms.py`.

يمكنك بسهولة إضافة كلاسات CSS، أو `placeholders`، أو أي سمة HTML أخرى.

مثال عملي:

لنفترض أن لديك هذا الحقل في القالب:

```
{{ form.email }}
```

باستخدام المكتبة، يمكنك تحويله إلى:

```
{% load widget_tweaks %}

{# Add class, placeholder, and disable autocomplete #}
{% render_field form.email class="form-input"
    placeholder="your@email.com" autocomplete="off" %}

{# Add dynamic class on error #}
{% if form.email.errors %}
    {% render_field form.email class="form-input-error" %}
{% endif %}
```

Thank You

Looking forward to seeing you
in the second assignment ;)