윈도우10 운영체제에서 특정URL 주소로 접속해서 매시간 화면을 캡쳐하는 프로그램을 만들고 싶어.

좋아요, 딱 필요한 기능이네요. 윈도우10에서 “지정 URL → 매시간 자동 접속 → 화면 캡처 저장” 하는 방법, 두 가지 루트로 바로 쓸 수 있게 정리해드릴게요.

---

1) 가장 안정적: Python + Playwright (권장)

Playwright는 브라우저를 알아서 설치/관리해주고, “페이지 전체(full page)” 캡처가 깔끔합니다.

설치

1. Python 설치(이미 있으면 생략)
2. 터미널(파워셀)에서:

```
powershell

pip install playwright
python -m playwright install chromium
```

스크립트 (예: `C:\capture\capture_site.py`)

```
python

import os, time, asyncio, glob
from datetime import datetime, timedelta
from playwright.async_api import async_playwright

URL      = os.getenv("TARGET_URL", "https://example.com")   # 캡처할 URL
OUTDIR   = os.getenv("OUTDIR", r"C:\capture\shots")         # 저장 폴더
VIEW_W   = int(os.getenv("VIEW_W", "1920"))                 # 뷰포트 가로
VIEW_H   = int(os.getenv("VIEW_H", "1080"))                 # 뷰포트 세로
WAIT_MS  = int(os.getenv("WAIT_MS", "3000"))                # 로딩 대기(ms)
KEEP_DAYS= int(os.getenv("KEEP_DAYS", "30"))                # 보관 일수

os.makedirs(OUTDIR, exist_ok=True)

async def run():
    async with async_playwright() as p:
        browser = await p.chromium.launch(headless=True)
        context = await browser.new_context(
            viewport={"width": VIEW_W, "height": VIEW_H},
            device_scale_factor=1.0,
            java_script_enabled=True,
        )
        page = await context.new_page()
        try:
            # 페이지 열기 및 네트워크 안정화 대기
            await page.goto(URL, wait_until="domcontentloaded", timeout=60000)
            await page.wait_for_load_state("networkidle")
            await page.wait_for_timeout(WAIT_MS)
```

```
            # 파일명: 2025-08-28_09-00-00.png
            ts = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
            out_path = os.path.join(OUTDIR, f"snap_{ts}.png")

            # 전체 페이지 캡처
            await page.screenshot(path=out_path, full_page=True)
            print(f"[OK] saved: {out_path}")

        finally:
            await context.close()
            await browser.close()

    # 오래된 파일 정리
    cutoff = datetime.now() - timedelta(days=KEEP_DAYS)
    for f in glob.glob(os.path.join(OUTDIR, "*.png")):
        try:
            mtime = datetime.fromtimestamp(os.path.getmtime(f))
            if mtime < cutoff:
                os.remove(f)
        except Exception:
            pass

if __name__ == "__main__":
    asyncio.run(run())
```

바꿔 쓸 수 있는 환경변수

- **`TARGET_URL`** : 캡처할 주소
- **`OUTDIR`** : 저장 경로
- **`KEEP_DAYS`** : 보관 기간(일)

스케줄(매시간) 등록

작업 스케줄러로 돌리는 걸 추천합니다.

1. 관리자 파워셸에서 경로 확인:

```
powershell

$PY = (Get-Command python).Source
$SCRIPT = "C:\capture\capture_site.py"
```

2. **매시간** 실행 작업 생성 (아침 9시 시작 예시):

```
powershell

schtasks /Create /TN "CaptureHourly" /SC HOURLY /ST 09:00 `
 /TR "`"$PY`" `"$SCRIPT`"" /F
```

- 시작 시간을 지금 이후로 잡으세요(스케줄러 규칙).
- 특정 URL/폴더를 고정하려면 작업의 “동작 → 프로그램/스크립트”에 **`python`**, “인수 추가”에 **`C:\capture\capture_site.py`** 지정.
- 프록시/로그인 등 특수 상황도 Playwright로 스크립트 내부에서 처리 가능합니다(쿠키/헤더 주입, 클릭/입력 등).

---

2) 초간단/무설치 느낌: PowerShell + Edge(Headless)

브라우저 설치/드라이버 없이, **Microsoft Edge**의 headless 모드를 직접 호출합니다.
단점: "완전한 전체 페이지"가 아니라 **지정 뷰포트 크기**만 캡처합니다. 긴 페이지는 **`--window-size`**를 크게 잡아 우회합니다.

스크립트 (예: `C:\capture\shot.ps1`)

```
powershell


param(
  [string]$Url = "https://example.com",
  [string]$OutDir = "C:\capture\shots",
  [int]$Width = 1920,
  [int]$Height = 5000  # 페이지가 길면 크게!
)

New-Item -ItemType Directory -Force -Path $OutDir | Out-Null
$ts = Get-Date -Format "yyyy-MM-dd_HH-mm-ss"
$outFile = Join-Path $OutDir "snap_$ts.png"

$edge = "$env:ProgramFiles(x86)\Microsoft\Edge\Application\msedge.exe"
if (!(Test-Path $edge)) { $edge = "$env:ProgramFiles\Microsoft\Edge\Application\msedge.exe" }

& $edge --headless=new --disable-gpu --window-size="$Width,$Height" `
  --screenshot="$outFile" "$Url"

Write-Host "[OK] saved: $outFile"
```

스케줄(매시간) 등록

```
powershell


schtasks /Create /TN "CaptureHourlyEdge" /SC HOURLY /ST 09:00 /F `
 /TR "powershell -ExecutionPolicy Bypass -File C:\capture\shot.ps1 -Url https://example.com -
OutDir C:\capture\shots -Width 1920 -Height 6000"
```

현장 팁(운영 환경에서 자주 막히는 포인트)

- **로그인 필요한 페이지**:
  - Playwright 사용 시 최초 1회 수동 로그인 후 **`storage_state`** 저장해 재사용하면 편합니다.
  - Edge headless는 세션 유지가 까다로워, 로그인 필요한 곳은 Playwright를 추천.
- **완전 렌더링 보장**: 데이터가 늦게 뜨면 **`WAIT_MS`** 늘리거나 **`page.wait_for_selector(...)`**로 특정 요소가 뜰 때까지 기다리세요.
- **경로/권한**: 저장 폴더는 시스템 계정으로도 쓰기 가능한 위치여야 합니다. 스케줄러가 "다른 사용자"로 돌 때 경로/권한 이슈가 흔합니다.
- **프록시/내부망**: Playwright 브라우저 컨텍스트에 프록시 설정 가능. 내부 CA 인증서가 필요하면 OS 트러스트스토어에 추가하세요.
- **장애 진단**: 스케줄러 "작업 기록(히스토리)" 켜고, 표준출력을 로그 파일로 리다이렉션하면 추적이 쉽습니다.

원하시면 **로그인 유지**(쿠키 저장) 버전이나 **특정 요소 스크린샷**, **여러 URL/여러 해상도 동시 캡처**, **파일명에 페이지 타이틀 포함** 등으로 바로 확장해드릴게요.

python -m playwright install chromium 설치할때, Error: unable to verify the first certificate (code:'UNABLE_TO_VERIFY_LEAF_SIGNATURE') 이런 오류가 나느데.

이 오류는 **사내 프록시/SSL 가로채기(사설 루트 CA)** 환경에서 자주 나옵니다.
`python -m playwright install chromium`은 내부적으로 **Node.js TLS**로 브라우저 바이너리를 내려받는데, **루트 인증서 체인**을 못 믿어서 터집니다. 해결 루트는 세 가지예요(안전한 순서대로).

A. 회사 루트 CA를 Node에 알려주기(권장)

1. **회사 루트/중간 인증서**를 **Base-64 PEM**(.pem) 파일로 준비
   - 보안팀/IT에서 공용 배포되는 루트 CA 파일을 받거나, Windows 인증서 저장소에서 내보내기("Base-64 encoded X.509 (.CER)").
2. 예: `C:\certs\corp-root.pem` 라고 저장했다고 가정
3. PowerShell(관리자)에서 **환경변수** 등록

```
powershell

# Node가 추가 CA를 믿도록 지정
setx NODE_EXTRA_CA_CERTS "C:\certs\corp-root.pem" /M

# 프록시 쓰는 경우(필요 시)
setx HTTPS_PROXY "http://user:pass@proxy.company.local:8080" /M
setx HTTP_PROXY  "http://user:pass@proxy.company.local:8080" /M
```

4. 새 터미널을 열고 재시도

```
powershell

python -m playwright install chromium
```

> 참고: Python용 Playwright도 브라우저 다운로드는 Node 스택을 써서,
> `NODE_EXTRA_CA_CERTS`가 핵심 스위치입니다.

B. 사내 미러(캐시) 또는 다운로드 호스트 변경

보안팀이 **사내 미러**를 운영한다면, 아래처럼 **호스트를 바꿔** 받게 할 수 있어요.

```
powershell

# 예) 내부 미러 사용(주소는 환경마다 다름)
setx PLAYWRIGHT_DOWNLOAD_HOST "https://playwright.mirror.company.local" /M
# (A안의 NODE_EXTRA_CA_CERTS도 함께 쓰는 걸 권장)
```

그다음:

```
powershell

python -m playwright install chromium
```

> 기본 호스트는 **`https://playwright.azureedge.net`** 입니다.
> 외부 접속이 막히면 미러가 가장 깔끔해요.

C. 오프라인 복사(인터넷 되는 PC에서 받아와서 옮기기)

1. 인터넷 되는 PC에서:

```
powershell

pip install playwright
python -m playwright install chromium
```

2. 그러면 **브라우저 바이너리**가 여기에 생깁니다:

```
shell

%USERPROFILE%\AppData\Local\ms-playwright\
```

3. 이 폴더 전체를 **오프라인 PC 동일 경로**에 복사
   (또는 공용 경로를 만들고)
4. 오프라인 PC에서 환경변수로 경로를 명시(선택)

```
powershell

# 공용 경로를 지정하고 싶다면(예: D:\tools\ms-playwright)
setx PLAYWRIGHT_BROWSERS_PATH "D:\tools\ms-playwright" /M
# 위 경로에 인터넷 PC에서 가져온 ms-playwright 내용을 그대로 둡니다.
```

5. 이제 바로 사용 가능:

```
powershell

python - << 'PY'
from playwright.sync_api import sync_playwright
with sync_playwright() as p:
    b = p.chromium.launch()
    b.new_page().goto("https://example.com")
    b.close()
PY
```

추가 체크리스트

- **프록시 인증**: 사내 프록시가 있으면 **`HTTPS_PROXY`/`HTTP_PROXY`**를 꼭 지정하세요.
  형식 예) **`http://user:password@proxy.company.local:8080`**
- **인증서 체인**: 루트뿐 아니라 **중간CA**가 필요한 경우도 있습니다. 여러 개를 **한 PEM 파일**에 이어붙여 두세요.

- **파워셸 재실행**: `setx`는 **새 터미널**에서만 반영됩니다.
- **최후의 수단(권장 X)**: 일시적으로만 테스트할 때

```
powershell

setx NODE_TLS_REJECT_UNAUTHORIZED 0 /M  # 전체 TLS 검증 비활성화 (보안상 비권장)
```

테스트 후 반드시 원복하세요:

```
powershell

reg delete "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment" /v
NODE_TLS_REJECT_UNAUTHORIZED /f
```

필요하시면

- 회사 CA 내보내기 스텝(스크린샷 포함)
- 사내 미러용 IIS/Nginx 설정 예시
- 오프라인 복사용 정확한 폴더 구조(브라우저별)
  를 바로 드릴게요. 환경을 한 번만 잡아놓으면 이후 설치가 쭉 편합니다.