The main components involved are:

1. **URL Strategies and RetailerFactory**: These define how URLs are constructed based on the retailer and strategy.
2. **RetailerBase and Retailer-specific Logic**: These handle processing of the JSON and HTML data scraped from the retailer's websites.
3. **WebScraperEngine**: This is the core engine that coordinates the scraping process.
4. **Main Function**: This initiates the entire flow.

**Step 1: URL Strategy Classes**

* The URLStrategy classes define different ways to generate URLs based on provided data (e.g., SKU, Description).
* These strategies (e.g., FindBySKUURL, FindByDescURL) are methods to dynamically generate a URL when scraping a retailer's site.

Example:

* FindBySKUURL: Generates a URL using the SKU (e.g., www.example.com/product/12345).
* FindByDescURL: Generates a URL using the description (e.g., www.example.com/product/description).

These strategies are essential because different retailers may require different URLs to retrieve the product information.

**Step 2: RetailerFactory**

* **Purpose**: The RetailerFactory is responsible for loading the configuration from a file (url\_strategies.json) and creating retailer-specific objects.
* It reads the retailer names and corresponding strategies from this configuration file.
* **Retailer Initialization**: For each retailer (like Amazon, Walmart, etc.), it sets up the strategies and links them to the RetailerBase subclass that represents that specific retailer.

In short, the factory:

1. Loads the URL strategies for each retailer.
2. Creates an instance of the retailer (e.g., AmazonRetailer or WalmartRetailer).
3. Initializes the retailer with its own strategy.

**Step 3: RetailerBase Class and Specific Retailer Logic**

* The RetailerBase class is an abstract base class (ABC) that outlines how each retailer processes its data. Each retailer (like Amazon or Walmart) must:
  + **Process JSON Data**: Process data returned by JSON endpoints (e.g., product information, pricing).
  + **Process HTML Data**: Parse and process HTML content scraped from the website.
  + **Merge Data**: Combine the processed JSON and HTML data to form a complete dataset.

Each retailer (e.g., AmazonRetailer, WalmartRetailer) implements these methods differently, as the structure and layout of the data can vary significantly between retailers.

For example:

* **AmazonRetailer** might extract pricing from JSON and the product description from HTML.
* **WalmartRetailer** might extract similar data but using different parsing logic.

**Step 4: WebScraperEngine**

The WebScraperEngine is the heart of the scraping process. Here's what it does step by step:

**1. Configuration Loading:**

* It loads the url\_strategies.json file through the RetailerFactory to get all the strategies for a particular retailer.

**2. Iterating through Strategies:**

* The scrape\_data method is where the strategy iteration happens.
* For each retailer (e.g., amazon), it loops through all strategies defined in url\_strategies.json.
  + It tries one strategy at a time (e.g., generate a URL based on SKU).
  + It generates a URL using the selected strategy and passes it to the scraper.

**3. Scraping Data:**

* For each strategy, it calls:
  + **json\_engine(url)**: This method fetches and simulates JSON data from the retailer’s website.
  + **html\_engine(url)**: This method fetches and simulates HTML content from the retailer’s webpage.

The purpose of these two methods is to simulate scraping the webpage for both structured (JSON) and unstructured (HTML) data.

**4. Processing Data:**

* After scraping the JSON and HTML data, the RetailerBase methods come into play:
  + **process\_json**: Retailer-specific logic to process the JSON data.
  + **process\_html**: Retailer-specific logic to process the HTML data.
  + **merge\_data**: The retailer-specific logic to merge both JSON and HTML into a complete dataset.

**5. Fallback if Strategy Fails:**

* If a strategy fails (due to an exception or no valid data), the scraper moves to the next strategy.
* It tries each strategy in sequence until one succeeds or all fail.

**6. Returning Data:**

* Once valid data is found, the merged data is returned.
* If none of the strategies succeed, an error message or empty data is returned.

**Step 5: Storing Data**

After scraping and processing, the scraped data is stored using the store\_data() method:

* This method is currently a placeholder and simply prints the data.
* In a real-world scenario, this is where the data would be saved to a database, file system, or cloud storage.

**Step 6: Main Function**

The main() function is where the entire process starts:

1. **Initialize Scraper**: The WebScraperEngine is initialized with the url\_strategies.json file.
2. **Choose Retailer and Strategy**: The retailer name (e.g., 'amazon') is passed to the scraper.
3. **Scrape Data**: The scrape\_data() method is called, which:
   * Iterates through strategies.
   * Tries generating URLs and scraping data.
   * Merges JSON and HTML data.
4. **Store Data**: After scraping, the merged data is stored or printed.

**Complete Flow Summary:**

1. **URL Strategy Creation**:
   * URL strategies are loaded for each retailer from the config file (url\_strategies.json).
2. **Retailer Factory**:
   * A RetailerFactory object creates retailer-specific objects (like AmazonRetailer) and associates them with their strategies.
3. **WebScraperEngine**:
   * The engine tries all available strategies for a retailer to generate URLs.
   * It fetches JSON and HTML data from the generated URLs.
   * The data is processed and merged using retailer-specific logic.
   * If a strategy fails, it moves to the next one until valid data is found.
4. **Data Storage**:
   * The scraped and processed data is stored or printed.