`Artifact`

The `Artifact` class represents a file artifact, encapsulating the file's path, type, contents, versions, and edit count. This class provides a comprehensive way to manage file versions, edit contents, and handle various file-related operations such as saving, loading, and exporting to JSON.

The `Artifact` class is particularly useful in contexts where file version control and content management are essential. By keeping track of the number of edits and maintaining a version history, it allows for robust file handling and auditability.

Class Definition

Artifact

Attribute	Type	Default Value	Description
`file_path`	`str`	N/A	The path to the file.
`file_type`	`str`	N/A	The type of the file.
`contents`	`str`	`""`	The contents of the file.
`versions`	`List[FileVersion]`	`[]`	The list of file versions.
`edit_count`	`int`	`0`	The number of times the file has been edited.

Parameters and Validation

- `file_path`: A string representing the file path.

- `file_type`: A string representing the file type. This attribute is validated to ensure it matches supported file types based on the file extension if not provided.
- `contents`: A string representing the contents of the file. Defaults to an empty string.
- `versions`: A list of `FileVersion` instances representing the version history of the file. Defaults to an empty list.
- `edit_count`: An integer representing the number of edits made to the file. Defaults to 0.

Methods

The `Artifact` class includes various methods for creating, editing, saving, loading, and exporting file artifacts.

`create`

Parameter	Type	Description
`initial_content`	`str`	The initial content of the file.

Usage Example:

```python
artifact = Artifact(file_path="example.txt", file_type="txt")
artifact.create(initial_content="Initial file content")
```

The file type parameter supports the following file types: `.txt`, `.md`, `.py`, `.pdf`.

`edit`

Parameter	Type	Description
`new_content`	`str`	The new content of the file.

Usage Example:

```python
artifact.edit(new_content="Updated file content")
```

`save`

Usage Example:

```python
artifact.save()
```

`load`

Usage Example:

```python
artifact.load()
```

```

#### `get_version`

| Parameter | Type | Description |
|---|---|---|
| `version_number` | `int` | The version number to retrieve. |

**Usage Example:**

```python
version = artifact.get_version(version_number=1)
```

`get_contents`

Usage Example:

```python
current_contents = artifact.get_contents()
```

`get_version_history`
```

Usage Example:

```python
version_history = artifact.get_version_history()
```

`export_to_json`

Parameter	Type	Description
`file_path`	`str`	The path to the JSON file to save the artifact.

Usage Example:

```python
artifact.export_to_json(file_path="artifact.json")
```

`import_from_json`

Parameter	Type	Description
`file_path`	`str`	The path to the JSON file to import the artifact from.

Usage Example:

```python
imported_artifact = Artifact.import_from_json(file_path="artifact.json")
```

`get_metrics`

Usage Example:

```python
metrics = artifact.get_metrics()
```

`to_dict`

Usage Example:

```python
artifact_dict = artifact.to_dict()
```

`from_dict`

Parameter	Type	Description

| `data` | `Dict[str, Any]` | The dictionary representation of the artifact. |

Usage Example:

```python
artifact_data = {
    "file_path": "example.txt",
    "file_type": "txt",
    "contents": "File content",
    "versions": [],
    "edit_count": 0
}
artifact = Artifact.from_dict(artifact_data)
```

Additional Information and Tips

- The `Artifact` class uses the `pydantic` library to handle data validation and serialization.
- When editing the artifact, ensure that the `file_path` is set correctly to avoid file operation errors.
- Use the `get_version` and `get_version_history` methods to maintain a clear audit trail of changes to the file.
- The `export_to_json` and `import_from_json` methods are useful for backing up and restoring the state of an artifact.

References and Resources

- [Pydantic Documentation](https://pydantic-docs.helpmanual.io/)
- [Python os.path module](https://docs.python.org/3/library/os.path.html)
- [JSON Documentation](https://docs.python.org/3/library/json.html)

Examples of Usage

Example 1: Creating and Editing an Artifact

```python
from datetime import datetime
from pydantic import BaseModel, Field, validator
from typing import List, Dict, Any, Union
import os
import json

# Define FileVersion class
class FileVersion(BaseModel):
    version_number: int
    content: str
    timestamp: datetime

# Artifact class definition goes here

# Create an artifact
artifact = Artifact(file_path="example.txt", file_type="txt")
artifact.create(initial_content="Initial file content")
```

```
# Edit the artifact
artifact.edit(new_content="Updated file content")

# Save the artifact to a file
artifact.save()

# Load the artifact from the file
artifact.load()

# Print the current contents of the artifact
print(artifact.get_contents())

# Print the version history
print(artifact.get_version_history())
```

Example 2: Exporting and Importing an Artifact

```python
# Export the artifact to a JSON file
artifact.export_to_json(file_path="artifact.json")

# Import

 the artifact from a JSON file
```

```
imported_artifact = Artifact.import_from_json(file_path="artifact.json")

# Print the metrics of the imported artifact
print(imported_artifact.get_metrics())
```

Example 3: Converting an Artifact to and from a Dictionary

```python
# Convert the artifact to a dictionary
artifact_dict = artifact.to_dict()

# Create a new artifact from the dictionary
new_artifact = Artifact.from_dict(artifact_dict)

# Print the metrics of the new artifact
print(new_artifact.get_metrics())
```