Client

EphemeralClient

```
def EphemeralClient(settings: Settings = Settings()) -> API
```

Creates an in-memory instance of Chroma. This is useful for testing and development, but not recommended for production use.

PersistentClient

```
def PersistentClient(path: str = "./chroma",
                     settings: Settings = Settings()) -> API
```

Creates a persistent instance of Chroma that saves to disk. This is useful for testing and development, but not recommended for production use.

Arguments:

- `path` - The directory to save Chroma's data to. Defaults to "./chroma".

HttpClient

```
def HttpClient(
    host: str = "localhost",
    port: str = "8000",
    ssl: bool = False,
    headers: Dict[str, str] = {},
    settings: Settings = Settings()) -> API
```

Creates a client that connects to a remote Chroma server. This supports many clients connecting to the same server, and is the recommended way to use Chroma in production.

Arguments:

- `host` - The hostname of the Chroma server. Defaults to "localhost".
- `port` - The port of the Chroma server. Defaults to "8000".
- `ssl` - Whether to use SSL to connect to the Chroma server. Defaults to False.
- `headers` - A dictionary of headers to send to the Chroma server. Defaults to {}.

Client

```
def Client(settings: Settings = __settings) -> API
```

Return a running chroma.API instance

Client Methods

```
class API(Component, ABC)
```

heartbeat

```
def heartbeat() -> int
```

Get the current time in nanoseconds since epoch. Used to check if the server is alive.

Returns:

- `int` - The current time in nanoseconds since epoch

list_collections

```
def list_collections() -> Sequence[Collection]
```

List all collections.

Returns:

- `Sequence[Collection]` - A list of collections

Examples:

```
client.list_collections()
# [collection(name="my_collection", metadata={})]
```

create_collection

```
def create_collection(name: str,
                      metadata: Optional[CollectionMetadata] = None,
                      embedding_function: Optional[EmbeddingFunction] = ef.
                      DefaultEmbeddingFunction(),
                      get_or_create: bool = False) -> Collection
```

Create a new collection with the given name and metadata.

Arguments:

- `name` - The name of the collection to create.
- `metadata` - Optional metadata to associate with the collection.
- `embedding_function` - Optional function to use to embed documents. Uses the default embedding function if not provided.
- `get_or_create` - If True, return the existing collection if it exists.

Returns:

- `Collection` - The newly created collection.

Raises:

- `ValueError` - If the collection already exists and get_or_create is False.
- `ValueError` - If the collection name is invalid.

Examples:

```
client.create_collection("my_collection")
# collection(name="my_collection", metadata={})
```

```
client.create_collection("my_collection", metadata={"foo": "bar"})
# collection(name="my_collection", metadata={"foo": "bar"})
```

get_collection

```
def get_collection(
    name: str,
    embedding_function: Optional[EmbeddingFunction] = ef.
    DefaultEmbeddingFunction()
) -> Collection
```

Get a collection with the given name.

Arguments:

- `name` - The name of the collection to get
- `embedding_function` - Optional function to use to embed documents. Uses the default embedding function if not provided.

Returns:

- `Collection` - The collection

Raises:

- `ValueError` - If the collection does not exist

Examples:

```
client.get_collection("my_collection")
# collection(name="my_collection", metadata={})
```

get_or_create_collection

```
def get_or_create_collection(
    name: str,
    metadata: Optional[CollectionMetadata] = None,
    embedding_function: Optional[EmbeddingFunction] = ef.
    DefaultEmbeddingFunction()
) -> Collection
```

Get or create a collection with the given name and metadata.

Arguments:

- `name` - The name of the collection to get or create
- `metadata` - Optional metadata to associate with the collection
- `embedding_function` - Optional function to use to embed documents

Returns:

The collection

Examples:

```
client.get_or_create_collection("my_collection")
# collection(name="my_collection", metadata={})
```

delete_collection

```
def delete_collection(name: str) -> None
```

Delete a collection with the given name.

Arguments:

- `name` - The name of the collection to delete.

Raises:

- `ValueError` - If the collection does not exist.

Examples:

```
client.delete_collection("my_collection")
```

reset

```
def reset() -> bool
```

Resets the database. This will delete all collections and entries.

Returns:

- `bool` - True if the database was reset successfully.

get_version

```
def get_version() -> str
```

Get the version of Chroma.

Returns:

- `str` - The version of Chroma

get_settings

```
def get_settings() -> Settings
```

Get the settings used to initialize the client.

Returns:

- `Settings` - The settings used to initialize the client.

Edit this page