Virtual DocTypes

Summary:
Virtual DocTypes in Frappe are an extension of the standard DocType, allowing developers to define custom DocTypes without creating a corresponding table in the database.

This feature is particularly useful for integrating data from diverse sources like external APIs, secondary databases, or even JSON/CSV files (*This enables the developers to plug-in database backends other than MariaDB and Postgres*).

While they offer backend flexibility, Virtual DocTypes maintain the same frontend appearance as regular DocTypes, ensuring a consistent user experience.

Key Insights:

1. **Purpose**: Virtual DocTypes are designed to integrate custom data sources while leveraging the frontend, resource APIs, roles, and permissions provided by the Frappe Framework.
2. **Creation**: When defining a Virtual DocType, developers need to check the "Virtual DocType" checkbox.
3. **Custom Controller**: Developers have the flexibility to define custom controllers for Virtual DocTypes.

```
DATA_FILE = "data_file.json"

@staticmethod
def get_current_data() -> dict[str, dict]:
    """Read data from disk"""
    if not os.path.exists(VirtualDoctype.DATA_FILE):
        return {}

    with open(VirtualDoctype.DATA_FILE) as f:
        return json.load(f)

@staticmethod
def update_data(data: dict[str, dict]) -> None:
    """Flush updated data to disk"""
    with open(VirtualDoctype.DATA_FILE, "w+") as data_file:
        json.dump(data, data_file)

def db_insert(self, *args, **kwargs):
    d = self.get_valid_dict(convert_dates_to_str=True)

    data = self.get_current_data()
    data[d.name] = d

    self.update_data(data)
```

An example provided showcases a controller that uses a JSON file as its data source. This controller has methods to read from and write to the JSON file, effectively using it as a "backend" for the Virtual DocType.

1. **Frontend Consistency**: Despite the backend differences, the frontend for Virtual DocTypes remains unchanged, ensuring that users interact with them just as they would with regular DocTypes.
2. **API Compatibility**: All `/api/resource` methods provided by the framework are compatible with Virtual DocTypes.