# 🧠 DAOkit on Gno.land

## Building Modular DAOs with Gno

## 🏛️ What is a DAO?

A Decentralized Autonomous Organization (DAO) is a self-governing entity operating through smart contracts It enables decentralized transparent decision-making.

## 🔑 Key Concepts

- **Proposal**: A request to execute an action within the DAO.
- **Resource**: An executable action triggered through proposals.
- **Condition**: Rules that must be met for a proposal to be executed.
- **Role**: Labels assigned to users granting specific permissions.

## 🧩 DAOkit Components

## 📦 DAOkit

Core package for building DAOs.

## 🧱 basedao

Extension handling membership and roles.

## ⚙️ daocond

Condition engine for complex proposal requirements.

## 🛠️ Implementing DAOkit

## 🧪 daocond

Defines conditions like thresholds and logical operations.

```
type Condition interface {
    Eval(votes map[string]Vote) bool
    Signal(votes map[string]Vote) float64
    Render() string
    RenderWithVotes(votes map[string]Vote) string
}
```

# 🧰 daokit

Manages resources and proposals.

```go
type Core struct {
    Resources *ResourcesStore
    Proposals *ProposalsStore
}
```

# 🧑‍🤝‍🧑 basedao

Handles members and roles.

```go
type Config struct {
    Name              string
    Description       string
    ImageURI          string
    Members           *MembersStore
    InitialCondition  daocond.Condition
}
```

## 🧪 Code Example: Basic DAO

```go
var (
    DAO        daokit.DAO
    daoPrivate *basedao.DAOPrivate
)

func init() {
    memberStore := basedao.NewMembersStore(initialRoles, initialMembers)

    condition := daocond.And(
        daocond.MembersThreshold(0.6, memberStore.IsMember, memberStore.MembersCount),
        daocond.RoleCount(1, "finance-officer", memberStore.HasRole),
    )

    DAO, daoPrivate = basedao.New(&basedao.Config{
        Name:             "Demo DAO",
        Description:      "A demo DAO built with DAOkit",
        Members:          memberStore,
        InitialCondition: condition,
    })
}
```

## 🧱 Creating Custom Resources

Implementing custom actions within your DAO.

# layout: two-cols

## 📝 Define Action

```go
type ActionNewPost struct {
    Title   string
    Content string
}
```

## ⚙️ Create Handler

```go
func NewPostHandler(blog *Blog) daokit.ActionHandler {
    return daokit.NewActionHandler("NewPost", func(payload interface{}) {
        action := payload.(*ActionNewPost)
        blog.NewPost(action.Title, action.Content)
    })
}
```

## 🎯 Use Case: Spending Funds

- **Resource**: SpendMoney
- **Condition**: Requires 50% approval from the administration board and the CFO.
- **Execution**: Proposal is executed only if the condition is met.

## 📚 Learn More

Explore the full documentation and examples:

- DAOkit README
- Gno.land