Sails.js Cheatsheet

A quick reference guide for Sails.js, covering core concepts, commands, configurations, and best practices for building scalable Node.js applications.

Core Concepts & Setup

Installation & Project Setup

Install Sails globally:

```
npm install -g sails
```

Create a new Sails project:

```
sails new my-app
```

Lift Sails server:

```
sails lift
```

Generate a model:

```
sails generate model User
```

Generate a controller:

```
sails generate controller UserController
```

Generate an API (model and controller):

```
sails generate api Product
```

Key Concepts

Models	Represent the data structure. Define attributes and their types (e.g., String, Number, Boolean).
Controllers	Handle incoming requests and define actions to be performed.
Views	Templates rendered to the client. Supports various templating engines (e.g., EJS, Jade).
Routes	Map URLs to controllers and actions. Can be configured in `config/routes.js`.
Policies	Middleware functions that run before actions. Used for authentication, authorization, etc.
Services	Reusable logic. Can be injected into controllers, models, or other services.

Models & ORM

Model Definition

Models are defined in `api/models/`. Each file represents a model.

Example: `api/models/User.js`

```
module.exports = {
  attributes: {
    firstName: { type: 'string', required: true },
    lastName: { type: 'string' },
    email: { type: 'string', required: true, unique: true },
    age: { type: 'number', defaultsTo: 18 }
  }
};
```

Attribute Types: `string`, `number`, `boolean`, `date`, `json`, `array`, `ref` (reference to another model)

Waterline ORM

Create

```
User.create({ firstName: 'John', lastName: 'Doe',
email: 'john@example.com' })
  .then(user => console.log(user))
  .catch(err => console.error(err));
```

Find

```
User.find({ lastName: 'Doe' })
  .then(users => console.log(users))
  .catch(err => console.error(err));
```

Update

```
User.update({ email: 'john@example.com' }, { lastName:
'Smith' })
  .then(users => console.log(users))
  .catch(err => console.error(err));
```

Destroy

```
User.destroy({ email: 'john@example.com' })
  .then(() => console.log('User deleted'))
  .catch(err => console.error(err));
```

findOne

```
User.findOne({ id: 1 })
  .then(user => console.log(user))
  .catch(err => console.error(err));
```

count

```
User.count()
  .then(count => console.log('User count: ', count))
  .catch(err => console.error(err));
```

Controllers & Routing

Controller Actions

Controllers are defined in `api/controllers/`. Each file represents a controller.

Example: `api/controllers/UserController.js`

```
module.exports = {
  index: function(req, res) {
    User.find().then(users => {
      return res.view('user/index', {
users: users });
    }).catch(err =>
res.serverError(err));
  },

  create: function(req, res) {
    User.create(req.body).then(user => {
      return res.redirect('/user');
    }).catch(err =>
res.serverError(err));
  }
};
```

Each function within `module.exports` is an action.

Routing

Configuration	Routes are configured in `config/routes.js`.
Example - Explicit Route	`'/users': { controller: 'UserController', action: 'index' }`
Example - Shortcut Route	`'get /users': 'UserController.index'`
Example - RESTful Route	Sails automatically generates RESTful routes for APIs. For example, `GET /user` maps to `UserController.find`.
Custom Routes	You can also define custom routes with parameters: `'/user/:id': { controller: 'UserController', action: 'show' }`

Request and Response Objects

req (Request)	Represents the HTTP request and has properties like `req.params`, `req.body`, `req.query`, `req.headers`.
res (Response)	Represents the HTTP response and has methods like `res.ok()`, `res.view()`, `res.redirect()`, `res.json()`, `res.serverError()`.

Views & Assets

Views

Views are located in `views/`. Sails uses EJS (Embedded JavaScript) by default, but you can configure other templating engines.

Example: `views/user/index.ejs`

```
<h1>Users</h1>
<ul>
  <% users.forEach(function(user) { %>
    <li><%= user.firstName %> <%=
user.lastName %></li>
  <% }); %>
</ul>
```

Use `res.view()` in your controller to render a view:

```
res.view('user/index', { users: users
});
```

Assets

Location	Assets (CSS, JavaScript, images) are stored in the `assets/` directory.
Pipeline	Sails uses Grunt to manage and pipeline assets. Configure in `tasks/pipeline.js`.
Linking	Link assets in your views using `<link>` and `<script>` tags. Sails provides helpers to automatically include assets.
Example	`<link rel="stylesheet" href="/styles/importer.css">` `<script src="/js/dependencies/sails.io.js"></script>`

Policies

Policies are middleware functions that run before your controller actions. They are located in `api/policies/`.

Example: `api/policies/isAuthenticated.js`

```
module.exports = function(req, res,
next) {
  if (req.session.user) {
    return next();
  }
  return res.forbidden('You are not
allowed to perform this action.');
};
```

Apply policies to controllers/actions in `config/policies.js`.