# Strapi Cheatsheet

List of the most common used strapi commands



#### How to know who is the logged in user?



ctx.request.body



## Request Body?



ctx.request.body



#### Request Query Parameters?



ctx request query



## **Dynamic Routes Parameters**



ctx.params

Endpoint GET: /users/:id

GET: /users/123

**const** { id } = ctx.params.id //id === 123



#### Interacting with Services and Controllers inside the code



strapi.services.SERVICENAME. **FUNCTION\_NAME** 

strapi.controllers.CONTROLLERNAME. **FUNCTION\_NAME** 

strapi.services.books.find()

strapi.controllers.books.create()

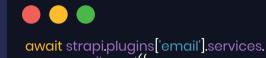


### Services vs Controllers

Services "Do" the actual work. Controllers are called from the API endpoint. They are useful for validation, sanitization and to check that certain conditions are met.



# Sending Emails



email.send({ to: 'paulbocuse@strapi.io', from: 'joelrobuchon@strapi.io', replyTo: 'no-reply@strapi.io', subject: 'Use strapi email provider successfully', text: 'Hello world!',

html: 'Hello world!', **})**;



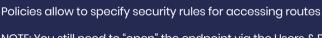
# Create a custom email sending service

npm strapi generate:service email



code API\_FOLDER/api/email/services /email.js

```
conts FROM_EMAIL = 'from@email.com'
conts FROM_EMAIL = 'reply@to.com'
module.exports = {
send: async (to, subject, html) => {
  strapi.log.info('services.email.sendEmail Sending email')
  await starpi plugins ['email'].services.email.send({}
    from: FROM_EMAIL,
    replyTo: REPLY_TO_EMAIL,
    subject,
    html
  strapi.log.info('services.email.send Email Sent')
```



#### NOTE: You still need to "open" the endpoint via the Users & Permissions in the /admin If you forget to activate the endpoint you'll see a 403 unauthorized

**Policies** 

isAuthenticated.js

```
module.exports = async (ctx, next) => {
    if (ctx.state.user) {
      return await next ();
    ctx.anauthorized ("You're not logged in!");
  };
isTargetUserLoggedIn.js
```



```
module.exports = async (ctx, next) => {
  if (ctx.state.user) {
  if (!ctx.request.query.user){
   return ctx.unauthorized('Specify a targer user equal to your own id ?user=${user.id}');
  consule.log('config/policies/isTargetUserLoggedIn ctx.request.query', ctx.request.query)
  const targetUser = ctx.request.query.user.toString()
  const loggedinUser = ctx.state.user.id.toString()
  consule.log('config/policies/isTargetUserLoggedIn targetUser', TargetUser)
  consule.log('config/policies/isTargetUserLoggedIn loggedInUser', loggedInUser)
  if (targetUser === loggedInUser){
         return await next ();
       } else {
        return ctx.unauthorized('target user is different from logged in user');
  ctx.unauthorized("You're not logged in!");
```