# Strapi Cheatsheet

List of the most common used strapi commands

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

```
//Retrieve logged in user
ctx.request.body
```

## 2 Request Body?

```
//Request body contents
ctx.request.body
```

## 3 Request Query Parameters?

```
//Request query parameters
ctx.request.query
```

## 4 Dynamic Routes Parameters

```
//Dynamic Routing Parameters
ctx.params

/* EXAMPLE */
Endpoint GET: /users/:id

GET: /users/123

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

## 5 Interacting with Services and Controllers inside the code

```
strapi.services.SERVICENAME.
FUNCTION_NAME

strapi.controllers.CONTROLLERNAME.
FUNCTION_NAME

//e.g.

strapi.services.books.find()

strapi.controllers.books.create()
```

## 6 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.

## 7 Sending Emails

```
await strapi.plugins['email'].services.
        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!',
});
```

## 8 Create a custom email sending service

```
# Generate a new service
npm strapi generate:service email
```

```
#Open the services file with your IDE
code API_FOLDER/api/email/services
/email.js
```

```
//Set defaults
conts FROM_EMAIL = 'from@email.com'
conts FROM_EMAIL = 'reply@to.com'

//Exports are you services + the core ones
//For more:
https://strapi.io/documentation/3.0.0-beta.x/concepts/services.html#core-services
module.exports = {
/**
   * Sends an email from the default FROM
   * @param to {string} Recipient
   * @param subject {string} Email Subject
   * @param html {string} Html contents of the email
   * /

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

## 9 Policies

Policies allow to specify security rules for accessing routes

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

isAuthenticated.js

```
//policies/isAuthenticated.js
module.exports = async (ctx, next) => {
  if (ctx.state.user) {
    // Go to next policy or will reach the controller's action.
    return await next ();
  }
  ctx.anauthorized ("You’re not logged in!");
};
```

## 10 isTargetUserLoggedIn.js

```
//Checks the target user from filter
//Checks the current logged in user
//If they are the same it let's execution run
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!");
};
```

**The Complete Strapi Developer Course - https://strapi.entreprenerd.xyz**