## 1. Goals

The goal of the lesson is to clarify the foundational terms and concepts in the authorization flow and introduce a production ready topology to replace the flat structure in the official Spring Security docs.

## 2. Lesson Notes

## 2.1. Why Not a Flat Topology?

First, let's look at an example of why a flat topology isn't enough.

We have a simple operation in a hospital management system:

```
@PreAuthorize("hasRole('doctor')")
public Patient getPatientRecords() { ... }
```

When we need nurses to now have access to the patient records as well:

```
@PreAuthorize("hasRole('doctor') or hasRole('nurse')")
public Patient getPatientRecords(...) { ... }
```

Now, what if the hospital administrator needs the same kind of access:

```
@PreAuthorize("hasRole('doctor') or hasRole('nurse') or hasRole('hospitalAministrator')")
public Patient getPatientRecords() { ... }
```

You can start seeing why this isn't going to be a great solution.

## 2.2. The Basic Terms

Let's first list out the following terms and concepts:

- the Privilege
- the Role
- the (Granted) Authority
- the Permission
- the Right

The Spring Security docs use some of these interchangeably and with not a lot of rigor.

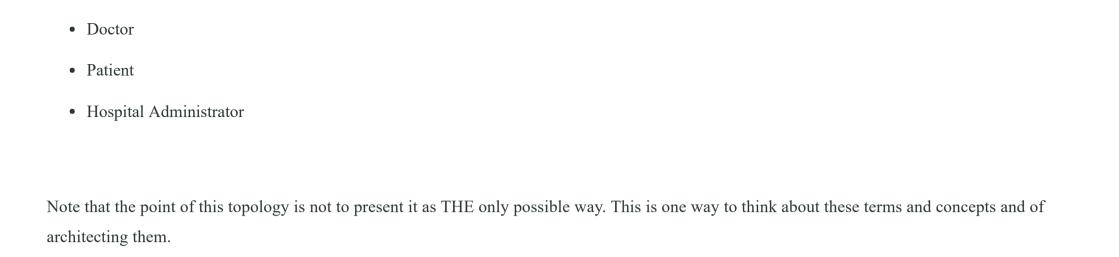
The first thing we need to do is to clearly decide and define what these mean for our system.

We are going to use Privilege to represent a granular, low level capability in the system - for example:

- can delete a Resource 1 that's a Privilege
- can update Resource 1 is another Privilege
- can delete Resource 2 is yet another, different Privilege

We're going to define Permission, Right and (Granted) Authority to mean the same thing as Privilege.

We'll define Role as a collection of Privileges - a higher level concept that's also user facing. For example:



What's important is that you make a decision, define the terms and use them consistently.