

# OpenMRS Roles and Privileges

A programmer's guide

# Introduction

- ▶ Medical data is usually sensitive information
- ▶ Data privacy must be taken very seriously
- ▶ OpenMRS has a security model based around roles and privileges



# What's a privilege?

- ▶ It's the ability to do a specific action, e.g.
  - View patient records
  - Edit encounters for a patient
- ▶ In OpenMRS these usually follow a naming convention of "<Verb> <Noun>", e.g.
  - View Patients
  - Edit Encounters
  - Add Location
  - Delete Observation



# What's a role?

- ▶ It's a collection of privileges required to do a particular job, e.g.
  - **Provider** requires
    - View Patients, Edit Patients
    - View Encounters, Edit Encounters
    - etc
  - **Data clerk** requires:
    - Add Encounters,
  - **Administrator** requires:
    - *ALL privileges!*



# Managing roles and privileges

- ▶ Administrators can manage roles and privileges from inside OpenMRS..



# Defining privileges

- ▶ A module can define new privileges
- ▶ These must be added to its `config.xml`
- ▶ For example, in the Appointments module...

```
<privilege>
  <name>View Appointments</name>
  <description>Able to view patient appointments</description>
</privilege>
<privilege>
  <name>Edit Appointments</name>
  <description>Able to edit patient appointments</description>
</privilege>
```

# Checking privileges

- ▶ There are several points in a module where we can and should check privileges...
  - On service methods
  - On JSP pages
  - On certain extension points that we are using

# Privileges via Extension Points

Privileges are built in to extension points. A class that extends PatientDashboardTabExt has a method for privileges. Anyone without privilege will not see the extension point.

```
public class PatientDashboardExt extends  
    PatientDashboardTabExt {
```

```
@Override
```

```
public String getRequiredPrivilege() {  
    return "View Appointments";  
}
```

Simply change the return for getRequiredPrivilege() to match the privilege specified in the config file.



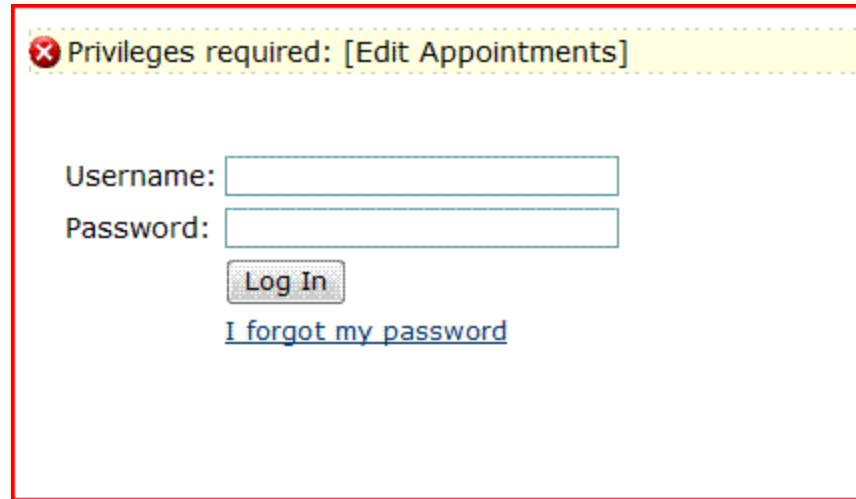
# Privileges in the Model

To allow or deny access to data accessed by a specific method use annotated privileges.

```
//Inside the AppointmentService interface:  
/**  
 * Get the appointments from the database for  
 * @return  
 */  
public Collection<Appointment> getAppointments();  
  
/**  
 * Save an appointment to the database  
 * @param appointment  
 */  
@Authorized( { "Edit Appointments" } )  
public Appointment saveAppointment(Appointment appointment);
```

Use the  
**@Authorized** tag  
to set privileges  
at a method level.

# Annotated Privilege Result



A screenshot of a web application's login interface. At the top, a yellow banner with a red 'X' icon contains the text "Privileges required: [Edit Appointments]". Below this, the form includes labels for "Username:" and "Password:" followed by text input fields. A "Log In" button is positioned below the password field. At the bottom of the form, there is a blue hyperlink that reads "I forgot my password". The entire form is enclosed in a red rectangular border.

This is the result of attempting to submit an appointment form as the user “doctor” who has a role of type Provider when Provider can view appointments but not edit them.

# Privileges in the Controller

You might need access to the privilege as you write logic for your page. Imagine, you want to send the user to one page if they have privileges and a different page if they don't or you want to process your data in one way if there are privileges and another if there aren't (maybe you want to change a name to XXX before displaying a patient record for a given privilege).

You can control the privileges at the web layer as follows:

```
if(Context.getUserContext().hasPrivilege("Edit Appointments")) {  
    // Logic for editing  
} else {  
    // Logic for viewing  
}
```

# Modifying the View Based on Privileges

For instance, on the form to enter appointments, you can let people view the information without letting them try to submit:

```
<tr>
<td>Reason:</td>
<td><form:input path="reason" /></td>
<td><form:errors path="reason" cssClass="error" /></td>
</tr>
</table>
<br>
<c:choose>
<c:when test="${privilege == 'edit'}">
<input type="submit" align="center" value="Do It!">
</c:when>
<c:otherwise>You do not have permissions to add/edit this
appointment</c:otherwise>
</c:choose>
```

# Modifying the View Based on Privileges—Another Way

- ▶ Used to hide page elements if user does not have a required privilege
- ▶ Elements to be hidden are put in the tag body, e.g.

```
<openmrs:hasPrivilege privilege="Edit Patients">  
  ...  
  Page items that user should only be able to see if they  
  have the "Edit Patients" privilege  
  ...  
</openmrs:hasPrivilege>
```