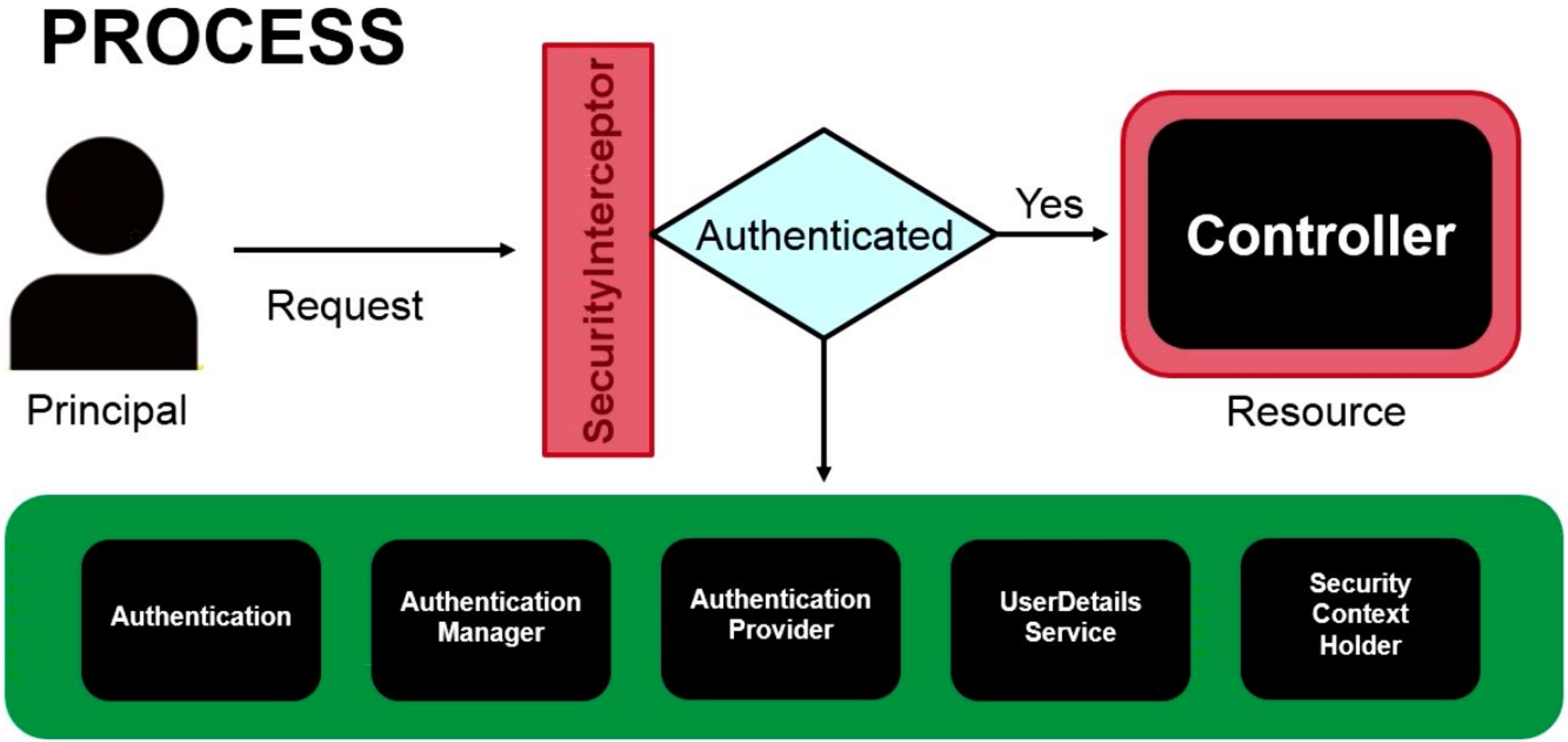
# DAO Based Authentication

## Spring Security Process



## UserDetailsService

- Configure Authentication Manager

```
@Autowired
public void configureGlobal(AuthenticationManagerBuilder auth) {
       auth.userDetailsService(userDetailsService);
}
```

- Retrieving details of the user used for authentication

```
@Component
public class CustomUserDetailsService implements UserDetailsService {

 private final IPlayerService userService;

 @Override
 public UserDetails loadUserByUsername(String userName) throws UsernameNotFoundException { ...
 }
}
```

## UserDetails

- Principal

```
public class SecurityUser implements UserDetails {
   @Override
   public Collection<GrantedAuthority> getAuthorities() {
       return authorities;
   }

   @Override
   public String getPassword() {
       return password;
   }
   @Override
   public String getUsername() {
       return username;
   }

   @Override
   public boolean isCredentialsNonExpired() {
       return true;
   }
}
```

## Managing Roles

```java
@Entity
@Table(name = "s_role")
public class Role implements Serializable {
   private static final long serialVersionUID = 1L;
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Integer id;

   @Column(name = "ROLE_NAME", nullable = false)
   private String roleName;

   @ManyToMany(cascade = CascadeType.PERSIST)
   @JoinTable(name = "ROLE_PLAYER",
           joinColumns = @JoinColumn(name = "ROLE_ID", referencedColumnName = "ID"),
           inverseJoinColumns = @JoinColumn(name = "PLAYER_ID", referencedColumnName = "ID")
   )
   private List<Player> player;
}
```

## Tying Together

- Loading Roles

```
@Query("select r.roleName from Role r left join r.player p where  p.credentials.username = :userName")
List<String> findPlayerRoles(@Param("userName") String userName);
```

- Adding Roles to Security User

```
@Override
public UserDetails loadUserByUsername(String userName)
   throws UsernameNotFoundException {
 Player user = userService.findByUsername(userName);
 List<String> roles = userService.findPlayerRoles(userName);
 return new SecurityUser(user, roles);
}
```