Custom & Multiple Authentication Providers in Single Spring Boot Application

1. Create Custom Authentication Provider class by implementing AuthenticationProvider Interface and override Authenticate and Support method.
   Support Method Checks for Type of Authenticate Request.
   Authenticate Method where actual authentication takes place.

```
package com.example.springsecurityDatabase.AuthProvider;

@Component
public class CustomAuthenticationProvider implements AuthenticationProvider {

    @Autowired
    UserDetailsService userDetailsService;

    @Override
    public Authentication authenticate(Authentication authentication) throws
AuthenticationException {
        final String username = (authentication.getPrincipal() == null) ? "NOT Provided"
: authentication.getName();

        if(StringUtils.isEmpty(username))
        {
            throw new BadCredentialsException("Invalid User Credentials");
        }

        UserDetails user = null;

        try {
            user = userDetailsService.loadUserByUsername(username);
        }
        catch (UsernameNotFoundException ex)
        {
            throw new BadCredentialsException("Invalid Login Credentials");
        }

        return createSuccessfulAuthentication(authentication,user);
    }

    private Authentication createSuccessfulAuthentication(Authentication authentication,
UserDetails user) {
        UsernamePasswordAuthenticationToken token = new
UsernamePasswordAuthenticationToken(user.getUsername(),user.getPassword());
        token.setDetails(authentication.getDetails());
        return token;
    }

    @Override
    public boolean supports(Class<?> authentication) {
        return authentication.equals(UsernamePasswordAuthenticationToken.class);
    }
}
```

2. Modifying SpringSecurity Configuration File.
   Create Instance of CustomAuthenticationProvder .
   Create Bean of DAOAuthenticationProvider.
   On Type of Request and output of Support Method Particular Authentication Provider will be get selected and whose Authenticate method will be get invoked thorugh Authenticate method of Authentication Manager which will contain logic for actual Authentication .

```
@Resource
CustomAuthenticationProvider customAuthenticationProvider;
```

```
@Bean
public DaoAuthenticationProvider authProvider()
{
    DaoAuthenticationProvider daoAuthenticationProvider = new
DaoAuthenticationProvider();
    daoAuthenticationProvider.setPasswordEncoder(encodepwd());
    daoAuthenticationProvider.setUserDetailsService(userDetailsService);
    return daoAuthenticationProvider;
}
```

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
   // auth.userDetailsService(userDetailsService).passwordEncoder(encodepwd());
    auth.authenticationProvider(authProvider())
            .authenticationProvider(customAuthenticationProvider);
}
```