Spring Security – pokročilé

Spring Security příklady

- Všechny oficiální příklady konfigurace Spring Security pro autorizaci / autentizaci pomocí různých způsobů naleznete zde:
  - https://github.com/spring-projects/spring-security/tree/master/samples

Spring Security & LDAP

- Práce s LDAP má tři fáze:
  - Získání LDAP DN (Distinguished Name) z přihlašovacího jména. Tato operace vyžaduje vyhledání v LDAPu. Například:
    - přihlašovací jméno: `jirkapinkas`
    - DN: `uid=jirkapinkas,ou=users,dc=javaskoleni,dc=cz`
  - Autentizace uživatele
  - Načtení autorit uživatele (autorizace)

Konfigurace LDAP serveru

- Nejprve je nutné nakonfigurovat přístup k LDAP serveru. V případě externího LDAP serveru se použije následující tag:

```
<ldap-server url="ldap://springframework.org:389/dc=springframework,dc=org" />
```

- Pokud je nutné se k LDAP serveru přihlásit, pak použijte atributy `manager-dn` a `manager-password`.

- Také můžete použít embedded server, který je velice užitečný pro testování, k tomu použijete tuto variantu `<ldap-server>`:

```
<ldap-server root="dc=springframework,dc=org" />
```

- Když použijete tento tag, tak se v classpath hledají `ldif` soubory. Pokud je nemáte přímo v classpath, pak jejich umístění specifikujete atributem `ldif`.

Embedded LDAP server

- Pro práci s embedded LDAP serverem se používá Apache Directory.
- V současnosti je podporována verze Apache Directory 1.5.5, vyšší nefungují … ale pro testování to stačí.
- Pro rozchození embedded LDAP serveru je nutné mít v classpath knihovny serveru! Viz. ukázkový příklad.
- Pozor na správné dependency – je nejlepší jich používat co nejméně. Existuje ještě dependency `apacheds-all` kde je vše, ale uvnitř je i SLF4J 1.5.6 a pak si musíte dát velký pozor abyste v classpath neměli vyšší verzi jakékoli SLF4J knihovny, jinak dostanete tuto chybu:
  - java.lang.NoSuchMethodError: org.slf4j.spi.LocationAwareLogger.log(Lorg/slf4j/Marker;Ljava/lang/String;ILjava/lang/String;[Ljava/lang/Object;Ljava/lang/Throwable;)

Autentizace a autorizace

- V nejjednodušším případě (když jsou všichni uživatelé v jednom uzlu) je možné použít toto:

```
<authentication-manager>
   <ldap-authentication-provider user-dn-pattern="uid={0},ou=people"/>
</authentication-manager>
```

- V reálném projektu obyčejně použijete LDAP search filter:

```
<authentication-manager>
   <ldap-authentication-provider
        user-search-filter="(uid={0})" user-search-base="ou=people"
        group-search-filter="(uniqueMember={0})" group-search-base="ou=groups"
        group-role-attribute="cn" role-prefix="ROLE_" />
</authentication-manager>
```

LDAP filtr pro hledání uživatelů

Kde jsou uživatelé

LDAP filtr pro hledání rolí

Kde jsou role

Atribut s názvem role.

Prefix, který bude přidán do názvu role získané z LDAPu.

Poznámka: Vyhledává se uvnitř DN, které je definované pomocí tagu `ldap-server`

Pokročilejší použití

- Pokud potřebujete výše uvedený způsob práce s LDAPem customizovat, pak nemusíte používat security tagy, ale přímo do Spring kontextu přidat beany, které ve finále udělají to samé. Jak na to je v dokumentaci:
  - http://docs.spring.io/spring-security/site/docs/3.2.x/reference/htmlsingle/#ldap

Spring Security & AD

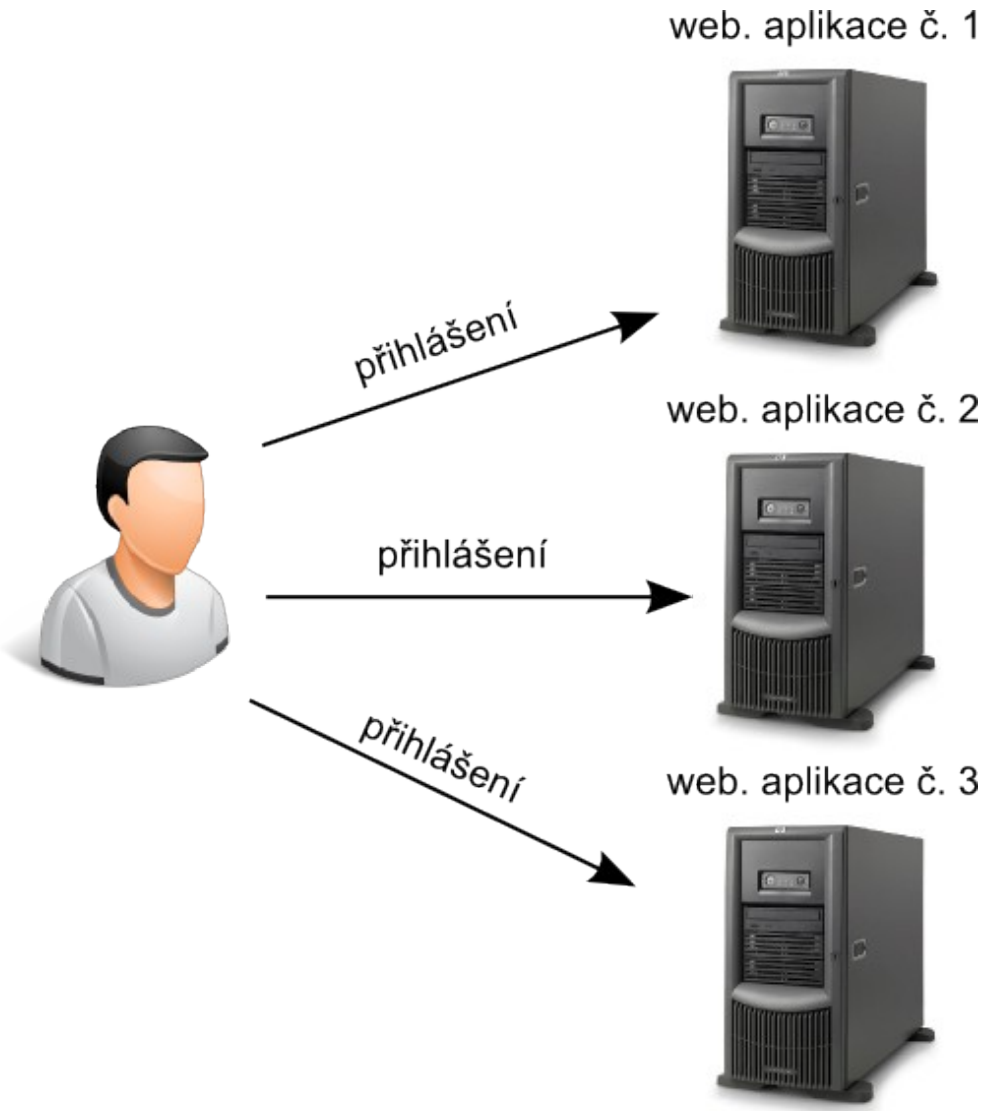
- Autorizace a autentizace pomocí Active Directory (AD) je relativně jednoduchá. Musíte do Spring kontextu přidat bean, kde je doménové jméno a URL LDAPu:

```
    <bean id="adAuthenticationProvider"
class="org.springframework.security.ldap.authentication.ad.ActiveDirectoryLdapAuthenticationP
rovider">

        <constructor-arg value="mydomain.com" />

        <constructor-arg value="ldap://adserver.mydomain.com/" />

    </bean>
```

- Uživatel bude mít role, které jsou definované v atributu `memberOf`.
- Bean má následující užitečné property:
  - `userDetailsContextMapper`: Pro customizaci uživatele
  - `authoritiesMapper`: Pro customizaci rolí

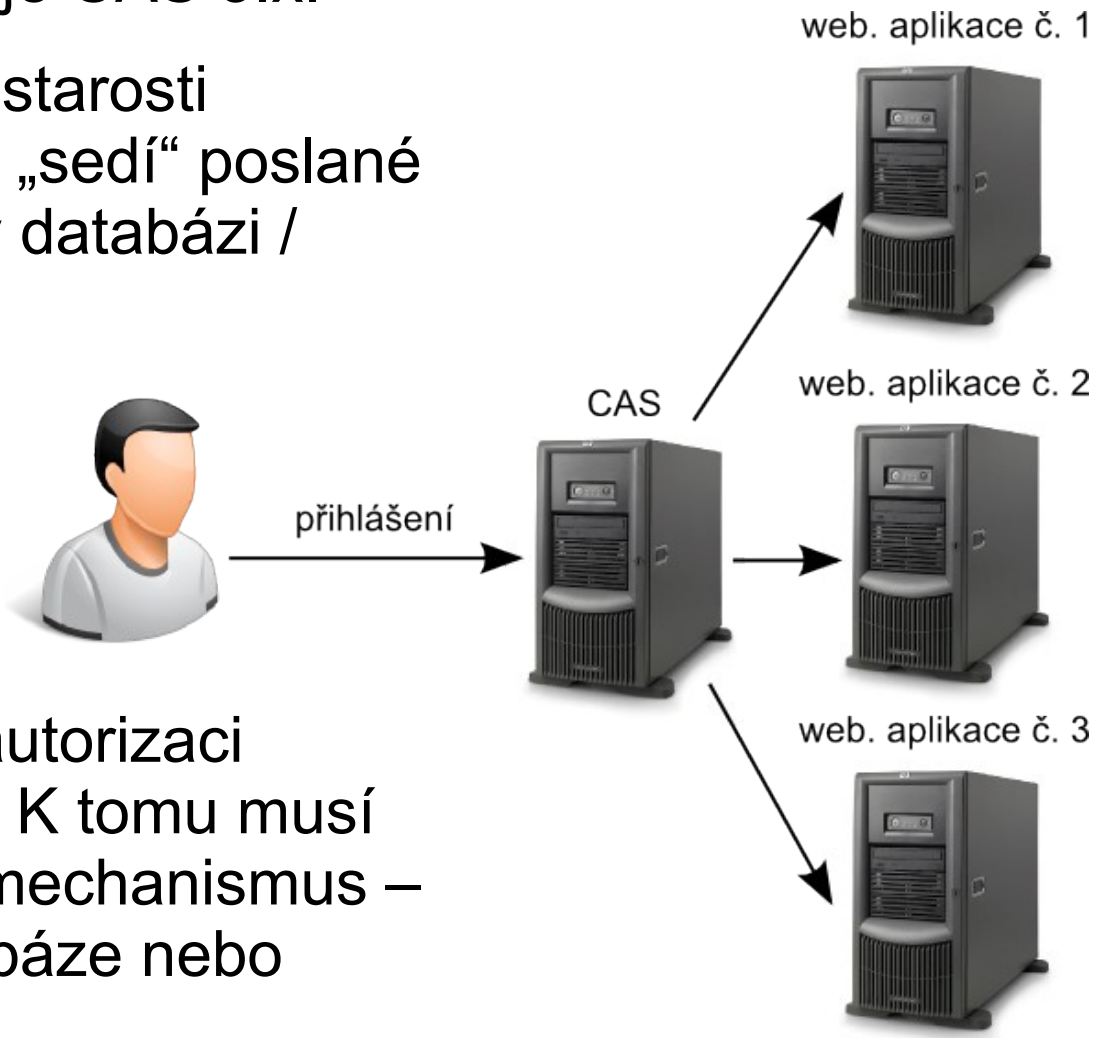
CAS (Central Authentication Service) I.

- CAS je open source enterprise single sign on (SSO) systém.
  - http://www.jasig.org/cas
- CAS řeší následující problém:
  - Web. aplikací je spousta, pokud má klient pracovat s více z nich, pak se musí do každé z nich přihlásit zvlášť.
  - Nebylo by pěkné přihlásit se pouze jednou?
  - Právě k tomu slouží CAS.



CAS (Central Authentication Service) II.

- Spring Security 3 podporuje CAS 3.x.
- Pozor! CAS má pouze na starosti autentizaci! (ověřuje, jestli „sedí“ poslané už. jméno a heslo s daty v databázi / LDAPu atd.)



- CAS se vůbec nestará o autorizaci (přidělení rolí uživatelům). K tomu musí web. aplikace použít jiný mechanismus – typicky načtení rolí z databáze nebo LDAPu.

Autentizace

- Uživatel přistoupí na zabezpečenou stránku, což vyhodí výjimku typu `AccessDeniedException` nebo `AuthenticationException`.
- Spring Security provede redirect uživatele na CAS server. Předá také parametr s názvem `service`, ve kterém je callback URL.
- Pokud je autentizace uživatele na CAS serveru úspěšná, pak CAS server zavolá callback URL s parametrem `ticket`.
- Webová aplikace vytvoří objekt typu `Principal` a vyvolá validaci ticketu (za tímto účelem opět zavolá CAS server).
- Pokud je ticket platný, pak webová aplikace přiřadí uživateli role.
- Webová aplikace vyvolá redirect na původní stránku, která vyvolala výjimku typu `AuthenticationException`.