infogain

Spring Security

Objectives

www.infogain.com

- Introduction to Spring Security
- Configure Spring Security
- Using Spring Security Standard Name
- Get Current Logged in Username
- Overriding Default Error Message

Prerequisite for Application Development

- **Prior Knowledge for this Session**
  - JSTL (JSP Standard Tag Library)
  - Spring MVC

- **Environment/Software required for this session**
  - Java 7/8 installed
  - Tomcat 7/8 installed

- **Dependence's**
  - Spring 3.2.8.RELEASE
  - Spring Security 3.2.3.RELEASE
  - JSTL 1.2 JAR

Spring Security

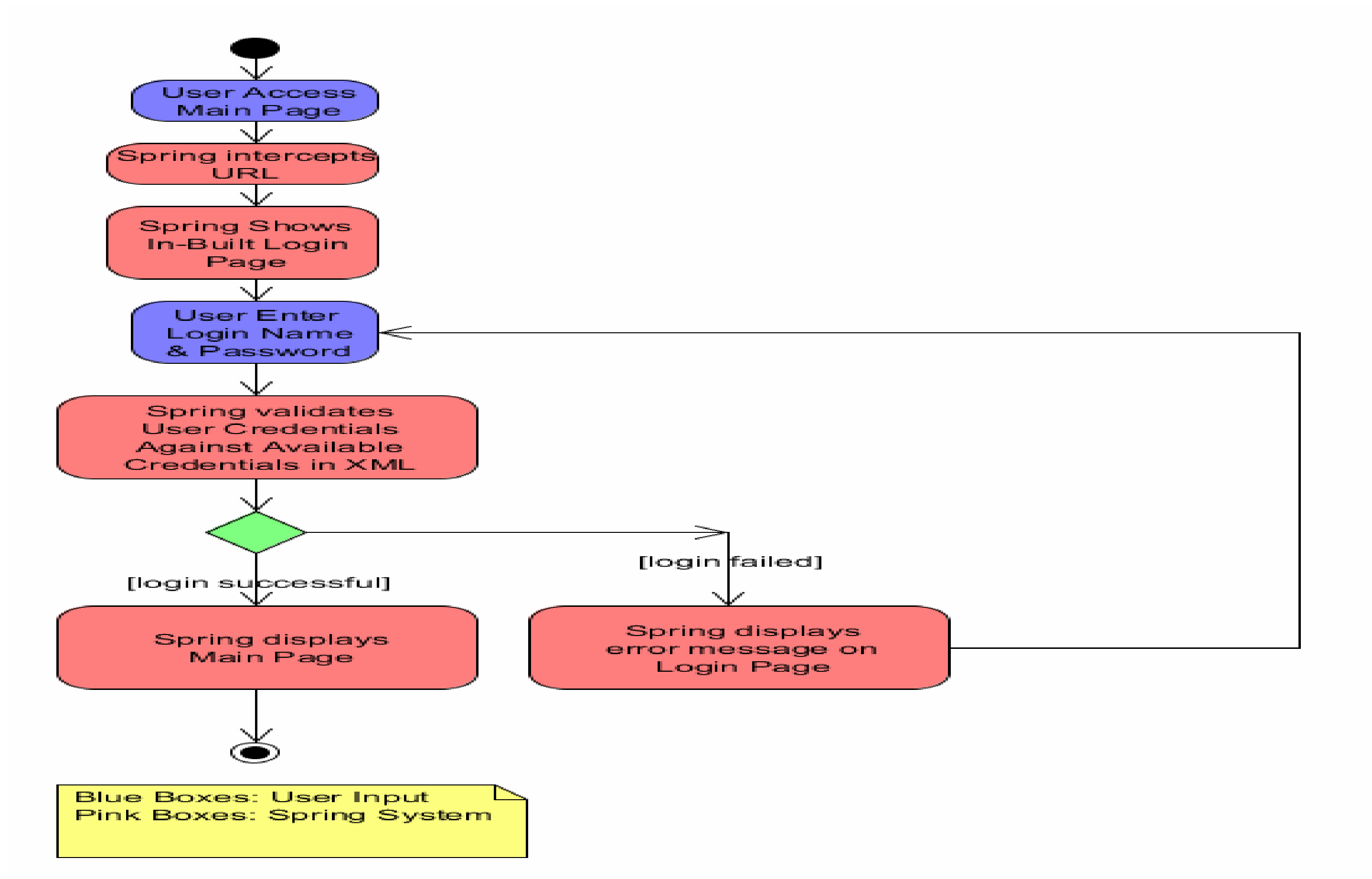
- **What is Spring Security**
  - It is a powerful framework that focuses on providing authentication and access control to secure Spring-based Java web application.
  - This framework targets two major areas of application they are authentication and authorization.
  - Authentication is the process of knowing and identifying the user that wants to access a resource.
  - Authorization is the process to allow authority to perform actions in the application.

- **Spring Project Modules**
  - In Spring Security 3.0, the Security module is divided into separate jar files. Based on their functionalities, so, the developer can integrate according to their requirement.

- **The following are some jar files that are included into Spring Security module.**
  - spring-security-core.jar
  - spring-security-web.jar
  - spring-security-config.jar
  - spring-security-ldap.jar

Interaction Diagram

- This is a simple example that intercepts a user request and presents a login page. Upon successful login it shows the success page and if unsuccessful, it shows an error message.


User Access Main Page
Spring intercepts URL
Spring Shows In-Built Login Page
User Enter Login Name & Password
Spring validates User Credentials Against Available Credentials in XML
[login successful]
[login failed]
Spring displays Main Page
Spring displays error message on Login Page
Blue Boxes: User Input
Pink Boxes: Spring System

Include Spring Security in your Project

- **To include spring security in your project, include below dependency:**
- **spring-security-core.jar**
  - core jar file is required for every application that wants to use Spring Security. This jar file includes **core access-control** and **core authentication classes** and interfaces.
- **spring-security-web**
  - This jar is useful for Spring Security web authentication and **URL-based access control**. It includes filters and **web-security infrastructure.**
  - All the classes and interfaces are located into the **org.springframework.security.web** package.
- **Spring-security-config**
  - This jar file is required for Spring Security configuration using **XML and Java** both. It includes Java configuration code and security namespace parsing code. All the classes and interfaces are stored in **org.springframework.security.config** package.

HTTP Basic Authentication

- **HTTP Basic Authentication**
  - Basic authentification is a standard **HTTP** header with the user and **password** encoded in **base64**
  - The userName and password is encoded in the format **username:password.**
  - This is one of the technique to protect the resources using URL it does not require cookies. session identifiers or any login pages.
  - In case of basic authentication, the username and password is only encoded with Base64, but not encrypted or hashed in any way.

Web.xml Configuration for Spring Security

infogain
www.infogain.com

- Just like Spring MVC, we need to bootstrap Spring Security
- **Context Loader Listener**

```xml
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

- Config Location

```xml
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/config/security-config.xml
    </param-value>
</context-param>
```

- Application Entry Point

```xml
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>

<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

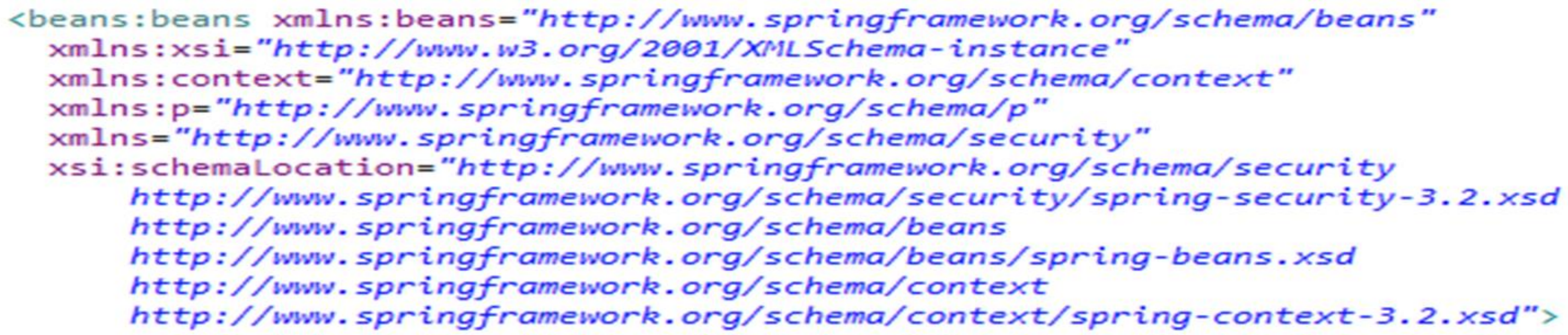
DelegatingFilterProxy

www.infogain.com

- **DelegatingFilterProxy**
  - It is a application filter which is used to intercepting the HTTP requests and performing authentication related tasks.
  - It must be defined as a Spring bean in your application context. So you need to register a bean named as **"springSecurityFilterChain",** which is an internal infrastructure bean created by spring container to handle web security.
  - Once it is added to your web.xml, you're ready to use Web security services configured using the <http> element

Security-config.xml Configuration

- **Another XML file**
    - Src/main/webapp/Web-INF/config
    - Configure Spring Security namespace



```
<beans:beans xmlns:beans="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns="http://www.springframework.org/schema/security"
  xsi:schemaLocation="http://www.springframework.org/schema/security
      http://www.springframework.org/schema/security/spring-security-3.2.xsd
      http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd
      http://www.springframework.org/schema/context
      http://www.springframework.org/schema/context/spring-context-3.2.xsd">
```

Security-config.xml Configuration(Contd..)

```
<http auto-config="true">
<intercept-url pattern="/profile**" access="ROLE_ADMIN" />
<form-login login-page='/login'
username-parameter="username"
password-parameter="password"
default-target-url="/profile"
authentication-failure-url="/login?authfailed" />
<logout logout-success-url="/login?logout" />
</http>
 <authentication-manager>
 <authentication-provider> <user-service>
  <user name=“abc” password="password" authorities=" ROLE_ADMIN" " /> </user-service>  </authentication-provider>
</authentication-manager>
```

Security-config.xml Configuration(Contd..)

- **http : **Include configuration related to url level security.
- **auto-config="true"** : Automatically registers a login form, BASIC authentication, logout services, remember-me and **servlet-api-integration**.
- **The <authentication-provider>** : Provides user information that will be used by the authentication manager to process authentication requests.
- **intercept-url** : This will match the requested url pattern from request and will decide what action to take based on access value.
- **form-login** : This will come into picture when user will try to access any secured URL. A login page mapped to "login-page" attribute will be served for authentication check. If not provided, spring will provide an inbuilt login page to user.

Security-config.xml configuration(contd..)

- **login-page Mapping** : URL of the custom login page. If not defined, then Spring Security will create a default URL at **'/spring_security_login'** and render a default login form.
- **username-parameter** Request parameter name which contains the username. Default is **'j_username'**.
- **password-parameter** Request parameter name which contains the password. Default is **'j_password'**.
- **default-target-url:** User will be redirected to this URL after successful login.
- **logout** : This will help to find the next view if logout is called in application.
- **authentication-failure-ur**l : If authentication failed, then user will be forwarded to this URL. Default is **/spring_security_login?login_error**'.
- **j_spring_security_check** : It is a Servlet where the actual authentication is made you must map the action of your login form to this Servlet.
- **your login page –**

```
<form id="Form1" name="myForm" method="post"
 action="j_spring_security_check">...</form>
```

Spring Security Standard Name

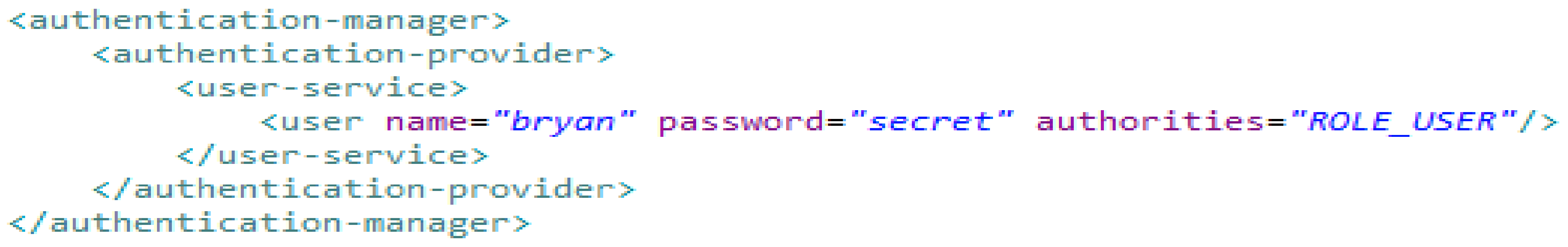
- **JSP Views In custom login form, you have to follow Spring Security standard name :**
  - j_spring_security_check - Login service
  - j_spring_security_logout - Logout service
  - j_username – Username
  - j_password – Password

- **To display authentication error messages, use this :**
  - ${sessionScope["SPRING_SECURITY_LAST_EXCEPTION"].message}

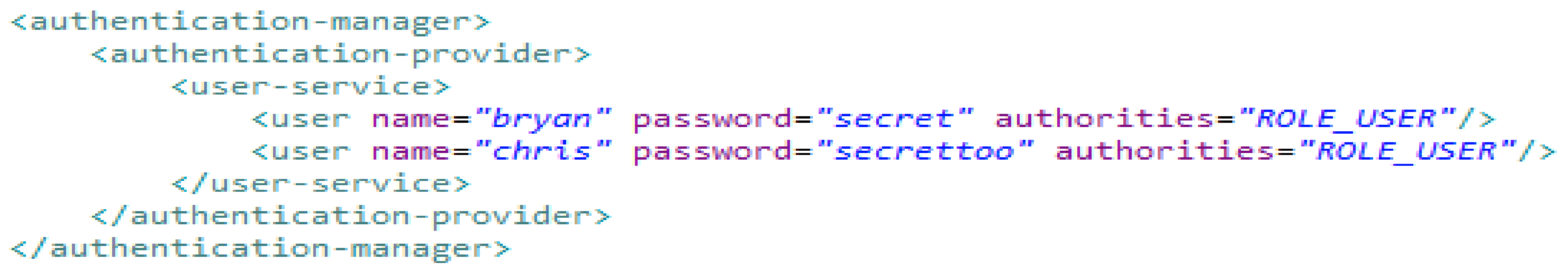
Storage

infogain
www.infogain.com

- In memory
- database
- **In memory**



```
<authentication-manager>
    <authentication-provider>
        <user-service>
            <user name="bryan" password="secret" authorities="ROLE_USER"/>
        </user-service>
    </authentication-provider>
</authentication-manager>
```



```
<authentication-manager>
    <authentication-provider>
        <user-service>
            <user name="bryan" password="secret" authorities="ROLE_USER"/>
            <user name="chris" password="secrettoo" authorities="ROLE_USER"/>
        </user-service>
    </authentication-provider>
</authentication-manager>
```

Database

- Spring Security contains a JdbcDaoImpl that can be configured to any database. Certain features are only available once connected to a database.
- **We need to create two tables**
  - Users Table
  - Authorities Table
- User Table

```
create table users(
    username varchar(50) not null primary key,
    password varchar(50) not null,
    enabled boolean not null);
```

- **Authorities Table**

```
create table authorities (
    username varchar(50) not null,
    authority varchar(50) not null,
    constraint fk_authorities_users
    foreign key(username) references users(username));
    create unique index ix_auth_username on authorities (username,authority);
```

Create User/Authority

infogain
www.infogain.com

- User SQL:

```
insert into users (username, password, enabled)
    values ("bryan", "secret", true);
```

- Authorities SQL:

```
insert into authorities (username, authority)
    value ("bryan", "ROLE_USER");
```

Configuring Data Source

www.infogain.com

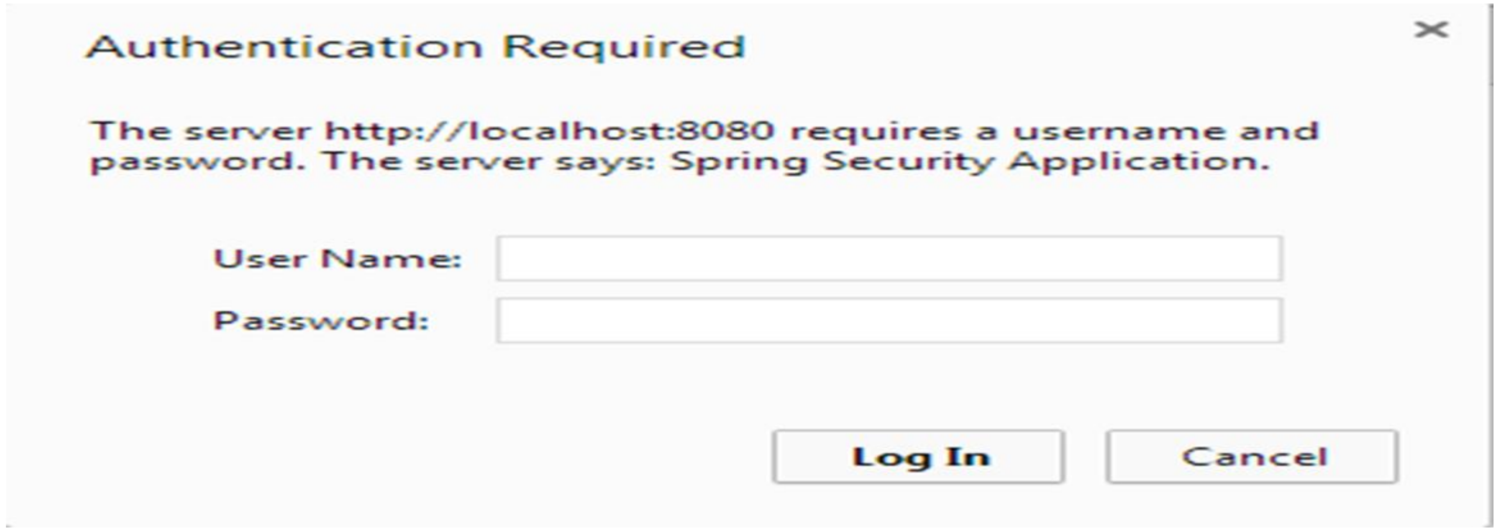
- **DriverManagerDataSource**
  - Used to contain the information about the database such as driver class name, connection URL, username and password
  - **It contains data base specific information**

- **Create a datasource for the database connection**
  - driverClassName = oracle.jdbc.driver.OracleDriver
  - url = jdbc:oracle:thin:@localhost:1521:xe
  - username = system
  - password = password

Basic Authentication

www.infogain.com

- Often used for RESTful web service authentication
  - Default login page for Basic auth
  - Useful for small apps or service based apps



Authentication Required

The server http://localhost:8080 requires a username and password. The server says: Spring Security Application.

User Name:

Password:

Login Form

www.infogain.com

- **Multiple steps for a Custom Login Page:**
  - form-login element
  - intercept-url element
  - LoginController
  - login.jsp

Login Controller

- Directs to our login.jsp
- Allows us to return more data to our index page
- Depends on your configuration

login.jsp

- Standard jsp page with a few key points:
    - j_spring_security_check
    - j_username
    - j_password

- Hosted through Spring MVC LoginController

Login Error

- **Login Error**
    - Add error param to our response
    - form-element authentication-failure-url
    - intercept-url for loginFailed
    - loginFailed in LoginController

Logout

- **Logout**
    - logout element
    - <intercept-url pattern=“/logout.html”/>
    - LoginController
    - logout.jsp

Get Current Logged in Username in Spring Security

- **UsernamePasswordAuthenticationToken(Class)**
  - UsernamePasswordAuthenticationToken(Class) injects UsernamePasswordAuthenticationToken into the Principal interface at runtime.
  - Using Principal getName() method you can get the user name.

```
 @Controller
 public class LoginController {
@RequestMapping(value="/login", method = RequestMethod.GET)
public String printWelcome(ModelMap model, Principal principal ) {
String name = principal.getName(); //get logged in username
model.addAttribute("username", name);
return "hello";
  }
```

infogain

Thank You

www.infogain.com

Infogain Corporation, HQ
485 Alberto Way Los Gatos,
CA 95032 USA
Phone: 408-355-6000
Fax: 408-355-7000

Pune
7th Floor, Bhalerao Towers, CTS No.1669 -
1670, Behind Hotel Pride,
Shivaji Nagar, Pune - 411005
Phone : +91-20-66236700

Infogain Irvine
41 Corporate Park,
Suite 390 Irvine, CA 2606 USA
Phone: 949-223-5100
Fax: 949-223-5110

Infogain Austin
Stratum Executive Center Building D
11044 Research Boulevard Suite 200
Austin, Texas 78759

Noida
A-16, Sector 60, Noida Gautam Budh agar,
201301 (U.P.) India
Phone: +91-120-2445144
Fax: +91-120-2580406

Dubai
P O Box 500588 Office No.105,
Building No. 4, Dubai Outsource Zone,
Dubai, United Arab Emirates
Tel: +971-4-458-7336