Web Applications Developments

ITWT413

LECTURE 12: JAKARTA EE WEB PROFILE

JAKARTA WEBSOCKET

DR. HALA SHAARI

Jakarta EE Tutorial- Structure

Basic Platform

- Resource Creation
- Injection
- Packaging

Jakarta EE Core Profile

- Jakarta CDI Lite
- Jakarta REST
- Jakarta JSON

Jakarta EE Web Profile

- Jakarta CDI Full
- Jakarta Validation
- Jakarta Security
- Jakarta Servlets
- Jakarta Faces
- Jakarta WebSocket
- Jakarta Persistence
- Jakarta Enterprise Beans Lite

Jakarta EE Platform

- Jakarta Mail
- Jakarta Messaging
- Jakarta Batch

Lecture Agenda

- Jakarta Bean Validation
  - Jakarta Bean Validation Constraints
  - Repeating Annotations
  - Validating Null and Empty Strings
  - Validating Constructors and Methods
- Jakarta Security
  - Guiding principles in Jakarta Security
  - Jakarta Security artifacts
  - Provided authentication mechanisms and identity stores
  - Securing an endpoint with Basic authentication
  - Declare the authentication mechanism

Overview of Jakarta Bean Validation

- Validating input received from the user to maintain data integrity is an important part of application logic.
- Validation of data can take place at different layers in even the simplest of applications.
- Jakarta Bean Validation provides a facility for validating objects, object members, methods, and constructors.
- In Jakarta EE environments, Jakarta Bean Validation integrates with Jakarta EE containers and services to allow developers to easily define and enforce validation constraints.

Using Jakarta Bean Validation Constraints

- The Jakarta Bean Validation model is supported by constraints in the form of annotations placed on a field, method, or class of a JavaBeans component, such as a managed bean.
- Constraints can be built in or user defined. User-defined constraints are called custom constraints.
- Several built-in constraints are available in the jakarta.validation.constraints package.

```
public class Name {
    @NotNull
    private String firstname;

    @NotNull
    private String lastname;
    ...
}
```

```
@USPhoneNumber
public String getPhone() {
    return phone;
}
```

Repeating Annotations

- From Bean Validation 2.0 onwards, you can specify the same constraint several times on a validation target using repeating annotation.

```
public class Account {

    @Max (value = 2000, groups = Default.class, message = "max.value")
    @Max (value = 5000, groups = GoldCustomer.class, message = "max.value")
    private long withdrawalAmount;
}
```

Validating Null and Empty Strings

- The Java programming language distinguishes between null and empty strings.
- An empty string is a string instance of zero length, whereas a null string has no value at all.
- An empty string is represented as " ". It is a character sequence of zero characters. A null string is represented by null. It can be described as the absence of a string instance.
- In order for the Bean Validation model to work as intended, you must set the context parameter jakarta.faces.INTERPRET_EMPTY_STRING_SUBMITTED_VALUES_AS_NULL to true in the web deployment descriptor file, web.xml.

```
<context-param>
    <param-name>jakarta.faces.INTERPRET_EMPTY_STRING_SUBMITTED_VALUES_AS_NULL</pa
    <param-value>true</param-value>
</context-param>
```

Validating Constructors and Methods

- Jakarta Bean Validation constraints may be placed on the parameters of nonstatic methods and constructors and on the return values of nonstatic methods.
- Static methods and constructors will not be validated.

```
public class Employee {
...
  public Employee (@NotNull String name) { ... }

  public void setSalary(
      @NotNull
      @Digits(integer=6, fraction=2) BigDecimal salary,
      @NotNull
      @ValidCurrency
      String currencyType) {
    ...
  }
...
}
```

Jakarta Security

- Jakarta Security is the overarching security API in Jakarta EE.
- Overarching here means that it strives to address the security needs of all other APIs in Jakarta EE in a holistic way.
- Due to historical and political reasons, a number of security features are still distributed among several other APIs in Jakarta EE.
- Sometimes they overlap, and sometimes such features are only accessible from these other APIs.

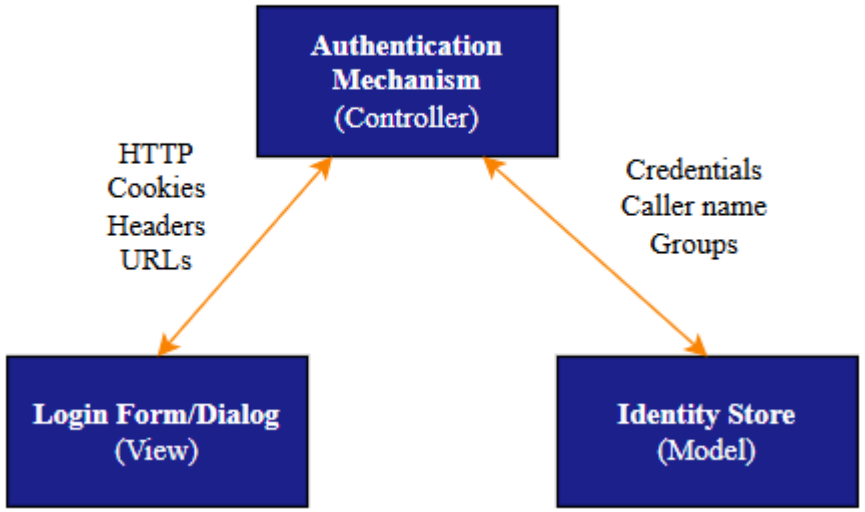
Some of the guiding principles in Jakarta Security

1. It should work directly out of the box, without requiring vendor-specific configuration.
2. It leverages Jakarta CDI as much as possible. Most artifacts are CDI beans, and many features are done via CDI interceptors.
3. The difference between framework-provided artifacts and custom (user provided) artifacts is minimal or non-existent.
4. It fully integrates with security features from other Jakarta EE APIs and proprietary (vendor-specific) artifacts.

Jakarta Security Artifacts

Jakarta Security defines several distinct artifacts that play an important role in the security process:

1. Authentication Mechanism
2. Identity Store
3. Permission Store



Provided authentication mechanisms and identity stores

- Jakarta Security provides a number of built-in authentication mechanisms and identity stores
- Authentication mechanisms:
  - Basic
  - Form
  - Custom Form
  - Open ID Connect (OIDC)
- Identity stores:
  - Database
  - LDAP (Lightweight Directory Access Protocol)

Securing an endpoint with Basic authentication

```java
@Path("/resource")
@RequestScoped
public class Resource {

    @Inject
    private SecurityContext securityContext;

    @GET
    @Produces(TEXT_PLAIN)
    public String getCallerAndRole() {
        return
            securityContext.getCallerPrincipal().getName() + " : " +
            securityContext.isCallerInRole("user");
    }

}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee https://jakarta.
         version="6.0">

    <security-constraint>
        <web-resource-collection>
            <web-resource-name>protected</web-resource-name>
            <url-pattern>/rest/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>user</role-name>
        </auth-constraint>
    </security-constraint>

</web-app>
```

Declare the authentication mechanism

- To declare the usage of a specific authentication mechanism, Jakarta EE provides [XYZ]MechanismDefinition annotations. Such an annotation is picked up by the security system, and in response to it a CDI bean that implements the HttpAuthenticationMechanism is enabled for it.

```
@ApplicationScoped
@BasicAuthenticationMechanismDefinition(realmName = "basicAuth")
@DeclareRoles({ "user", "caller" })
@ApplicationPath("/rest")
public class ApplicationConfig extends Application {

}
```

Jakarta EE Platform

- Jakarta Mail
  - The Jakarta Mail API provides a set of abstract classes defining objects that comprise a mail system. The API defines classes like Message, Store and Transport. The API can be extended and can be subclassed to provide new protocols and to add functionality when necessary.
- Jakarta Messaging
  - Messaging is a method of communication between software components or applications. A messaging system is a peer-to-peer facility: A messaging client can send messages to, and receive messages from, any other client. Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages.
- Jakarta Batch
  - Some enterprise applications contain tasks that can be executed without user interaction. These tasks are executed periodically or when resource usage is low, and they often process large amounts of information such as log files, database records, or images. Batch processing refers to running batch jobs on a computer system. Jakarta EE includes a batch processing framework that provides the batch execution infrastructure common to all batch applications, enabling developers to concentrate on the business logic of their batch applications.

Questions?