



# Painless Application Security

**Les Hazlewood**

Apache Shiro Project Chair

CTO, Katasoft Inc / CloudDirectory

# What is Apache Shiro?

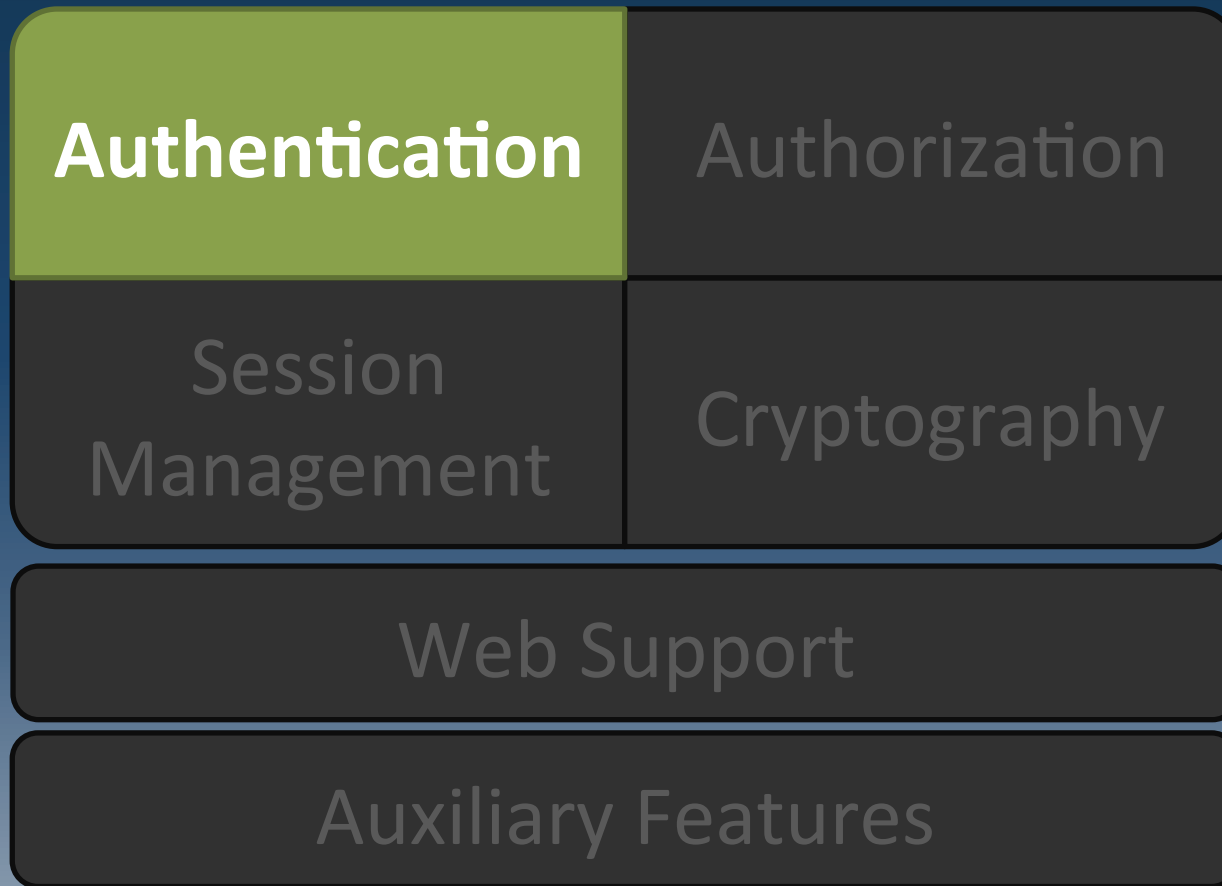
- Application security framework
- ASF TLP - <http://shiro.apache.org>
- Quick and Easy
- Comprehensive
- Simplifies Security Concepts & Design



# Agenda

Authentication	Authorization
Session Management	Cryptography
Web Support	
Auxiliary Features	

# Authentication



# Authentication Defined

Identity verification:

Proving a user is who he says he is



# Shiro Authentication Features

- Subject-based (current user)
- Single method call
- Rich Exception Hierarchy
- 'Remember Me' built in
- Event listeners



# How to Authenticate with Shiro

## Steps

1. Collect principals & credentials
2. Submit to Authentication System
3. Allow, retry, or block access



# Step 1: Collecting Principals & Credentials

```
UsernamePasswordToken token = new  
    UsernamePasswordToken(username, password);  
  
// "Remember Me" built-in:  
token.setRememberMe(true);
```

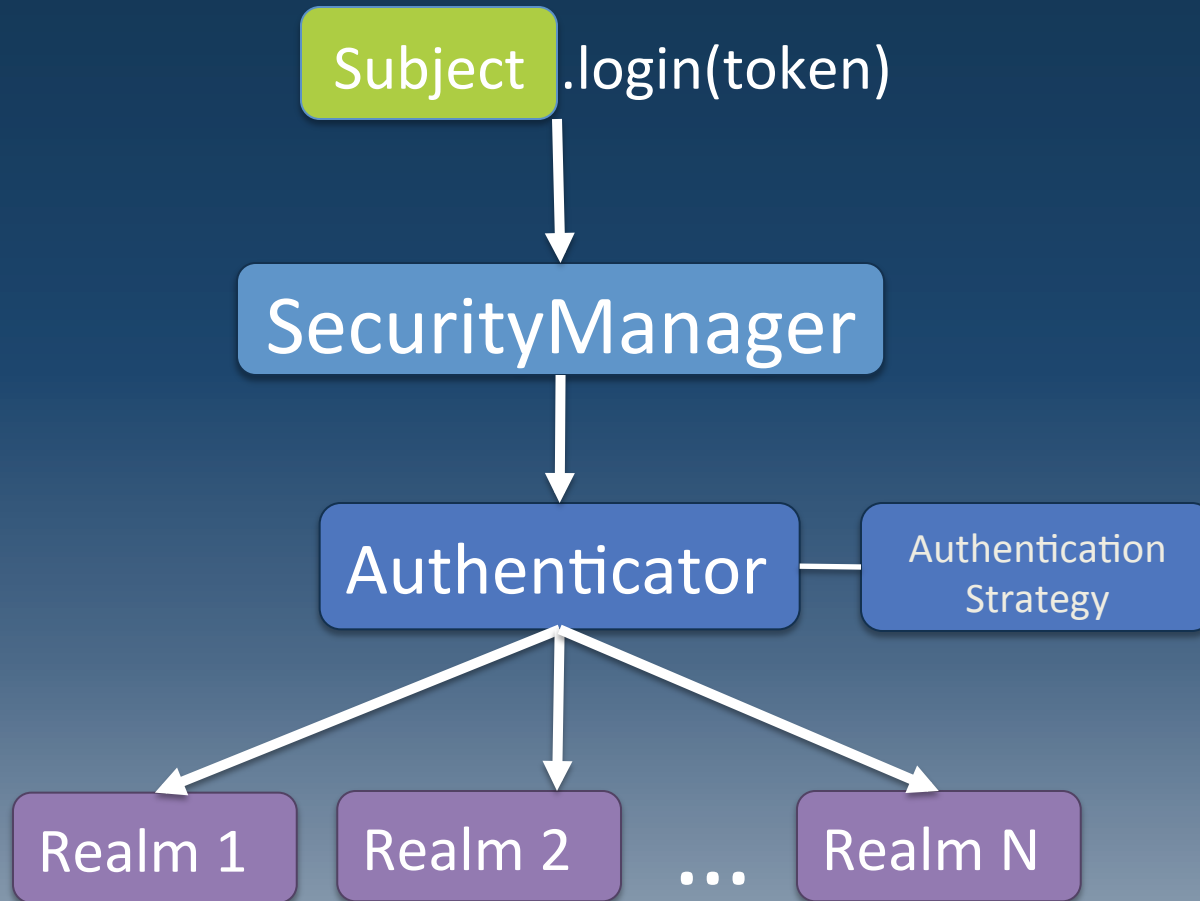
## Step 2: Submission

```
Subject currentUser =  
    SecurityUtils.getSubject();  
  
currentUser.login(token);
```

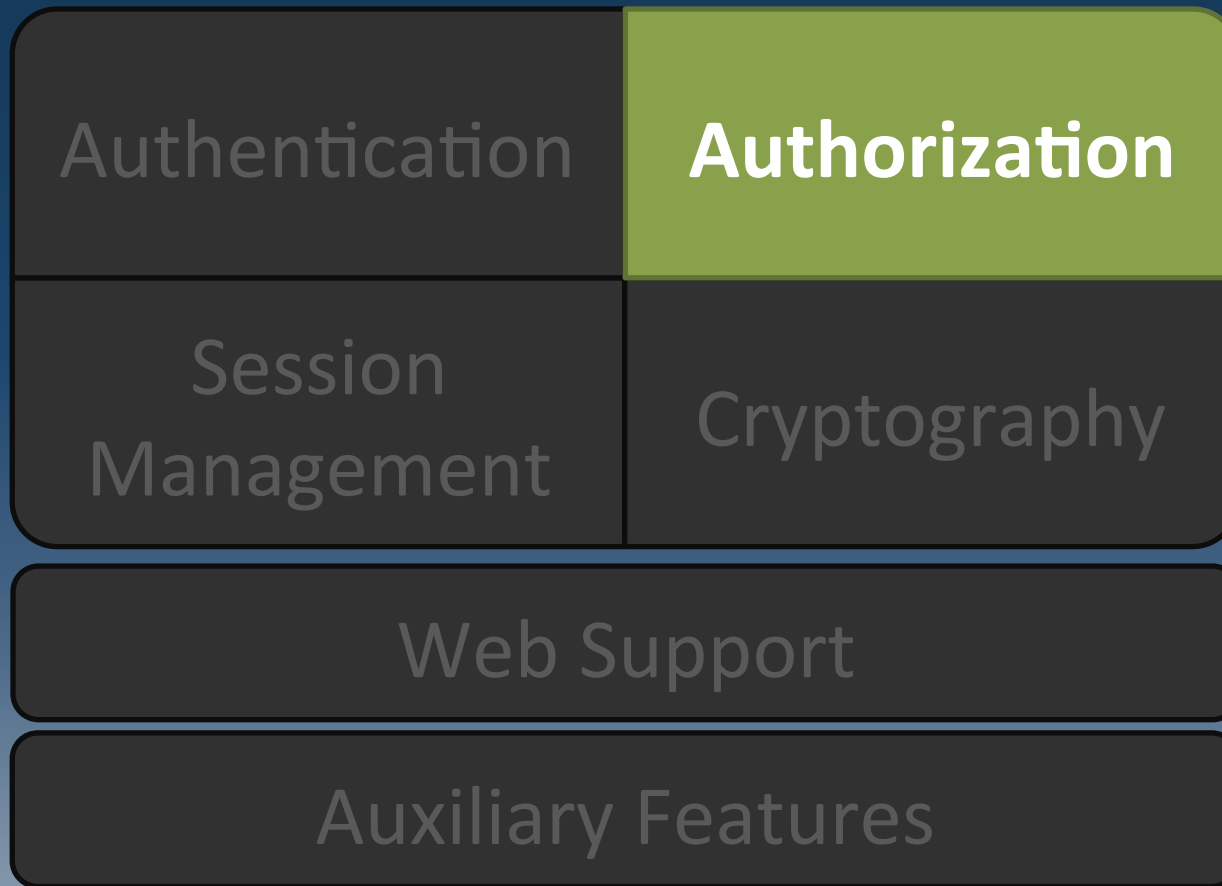
## Step 3: Grant Access or Handle Failure

```
try {
    currentUser.login(token);
} catch (UnknownAccountException uae ) { ...
} catch (IncorrectCredentialsException ice { ...
} catch ( LockedAccountException lae ) { ...
} catch ( ExcessiveAttemptsException eae ) { ...
} ... catch your own ...
} catch ( AuthenticationException ae ) {
    //unexpected error?
}
//No problems, show authenticated view...
```

# How does it work?



# Authorization



# Authorization Defined

Process of determining “who can do what”  
AKA Access Control

## Elements of Authorization

- Permissions
- Roles
- Users



# Authorization Features

- Subject-centric (current user)
- Checks based on roles or permissions
- Powerful out-of-the-box WildcardPermission
- Any data model – Realms decide



# How to Authorize with Shiro

Multiple means of checking access control:

- Programmatically
- Java Annotations & AOP
- JSP/GSP/JSF TagLibs (web support)



# Programmatic Authorization

## Role Check

```
//get the current Subject
Subject currentUser =
    SecurityUtils.getSubject();

if (currentUser.hasRole("administrator")) {
    //show the 'delete user' button
} else {
    //don't show the button?
}
```

# Programmatic Authorization

## Permission Check (String-based)

```
String perm = "user:delete:jsmith";

if (currentUser.isPermitted(perm) ) {
    //show the 'delete user' button
} else {
    //don't show the button?
}
```

# Annotation Authorization

## Role Check

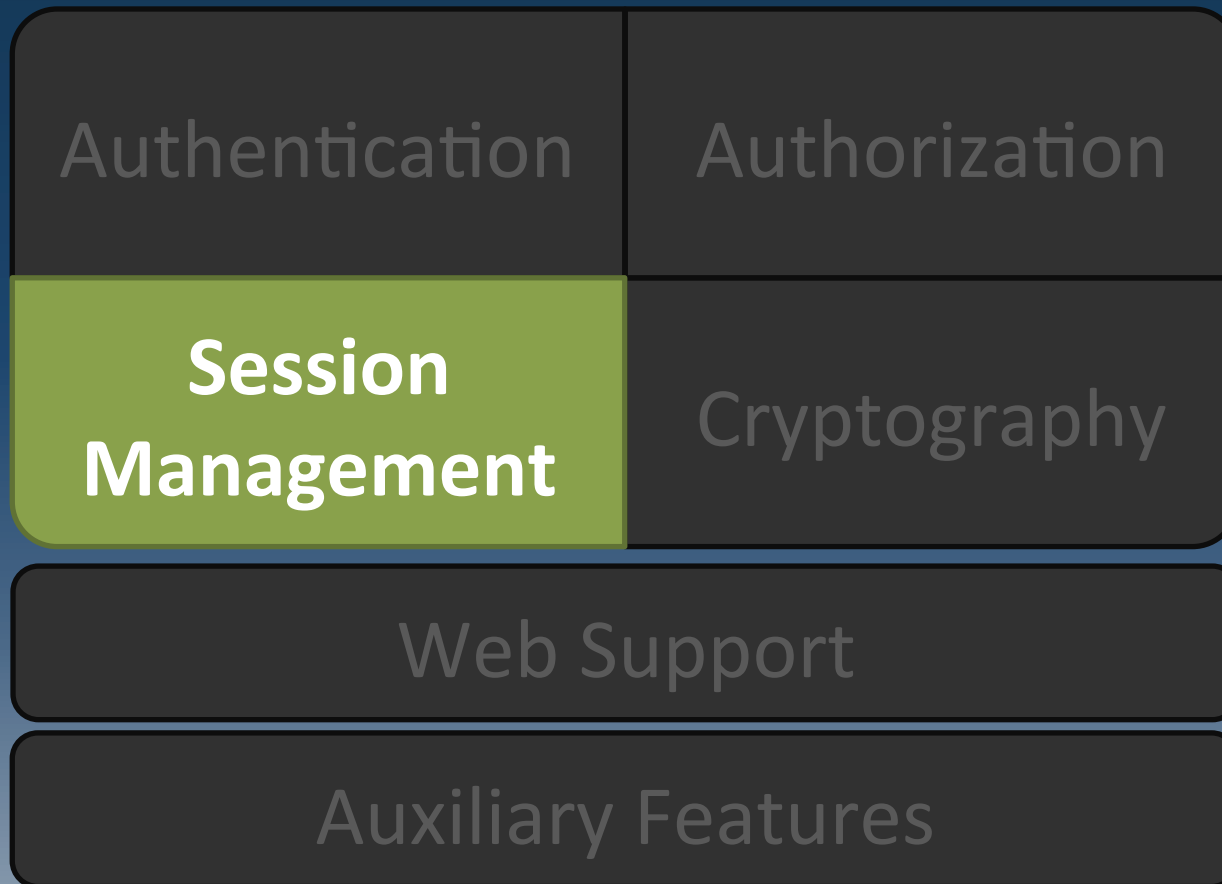
```
@RequiresRoles( "teller" )  
public void openAccount(Account a) {  
    //do something in here that  
    //only a 'teller' should do  
}
```

# Annotation Authorization

## Permission Check

```
@RequiresPermissions("account:create")
public void openAccount(Account a) {
    //create the account
}
```

# Enterprise Session Management



# Session Management Defined

Managing the lifecycle of Subject-specific temporal data context



# Session Management Features

- Heterogeneous client access
- POJO/JSE based (IoC friendly)
- Event listeners
- Host address retention
- Transparent web use - HttpSession
- Container-Independent Clustering!

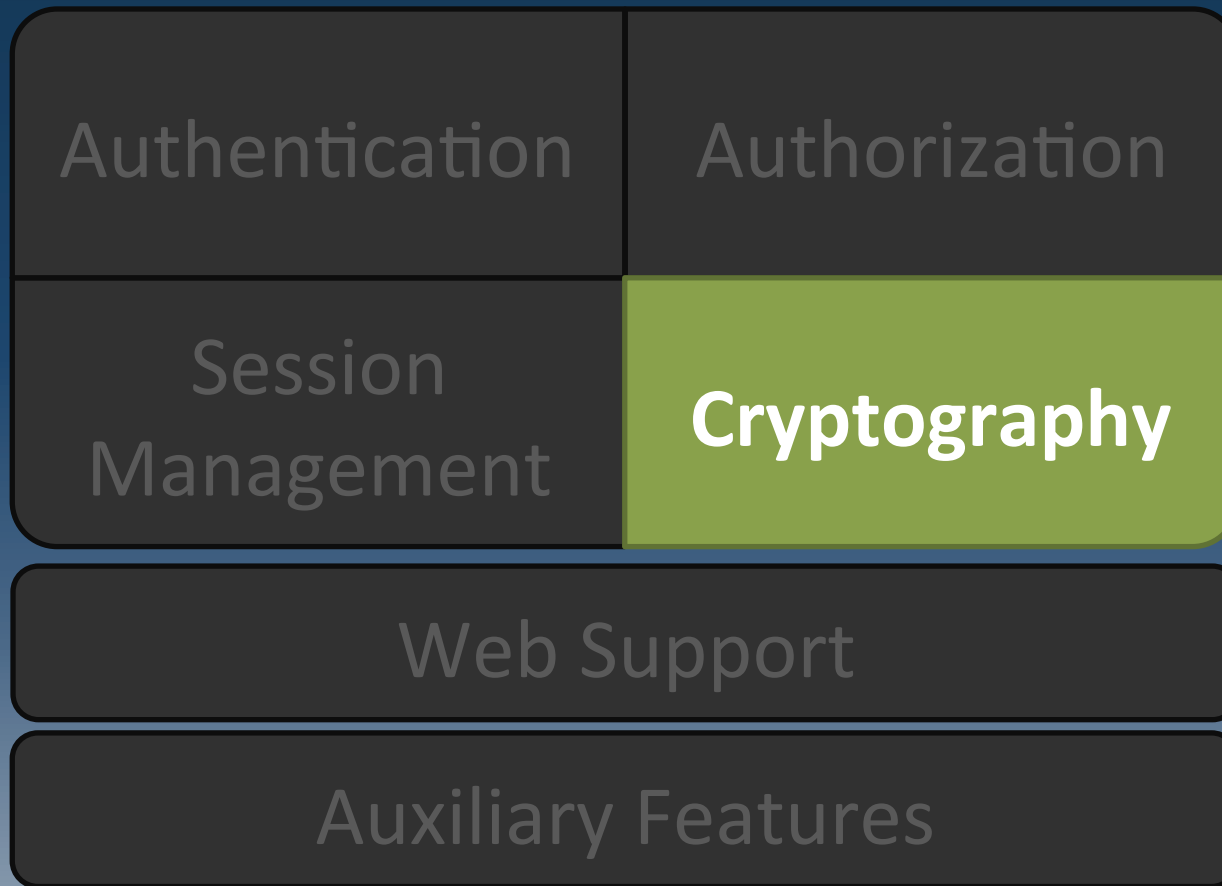
# Acquiring and Creating Sessions

```
Subject currentUser =  
    SecurityUtils.getSubject()  
  
//guarantee a session  
Session session = subject.getSession();  
  
//get a session if it exists  
subject.getSession(false);
```

# Session API

```
getStartTimestamp()  
getLastAccessTime()  
getAttribute(key)  
setAttribute(key, value)  
get/setTimeout(long)  
touch()  
...
```

# Cryptography



# Cryptography Defined

Protecting information from undesired access by hiding it or converting it into nonsense.

## Elements of Cryptography

- Ciphers
- Hashes

# Cryptography Features

## Simplicity

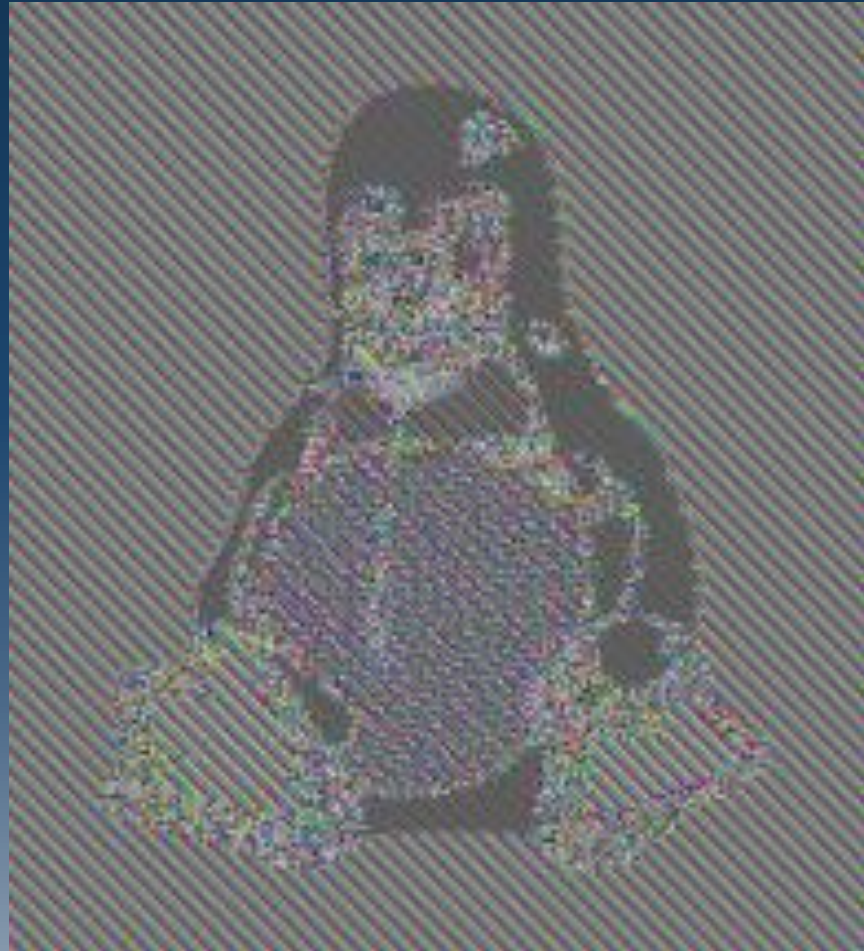
- Interface-driven, POJO based
- Simplified wrapper over JCE infrastructure
- “Object Orientifies” cryptography concepts
- Easier to understand API
- More Secure Defaults than the JDK!

# Example: Plaintext



(image courtesy Wikipedia)

# Example: ECB Mode (JDK Default!)



(image courtesy Wikipedia)

# Example: Shiro Defaults



(image courtesy Wikipedia)

# CipherService Example

```
AesCipherService service =  
    new AesCipherService();  
service.setKeySize(256);  
  
byte[] key = service.generateNewKey()  
    .getEncoded();  
  
byte[] encrypted =  
    service.encrypt(rawData, key);
```

# Hash Features

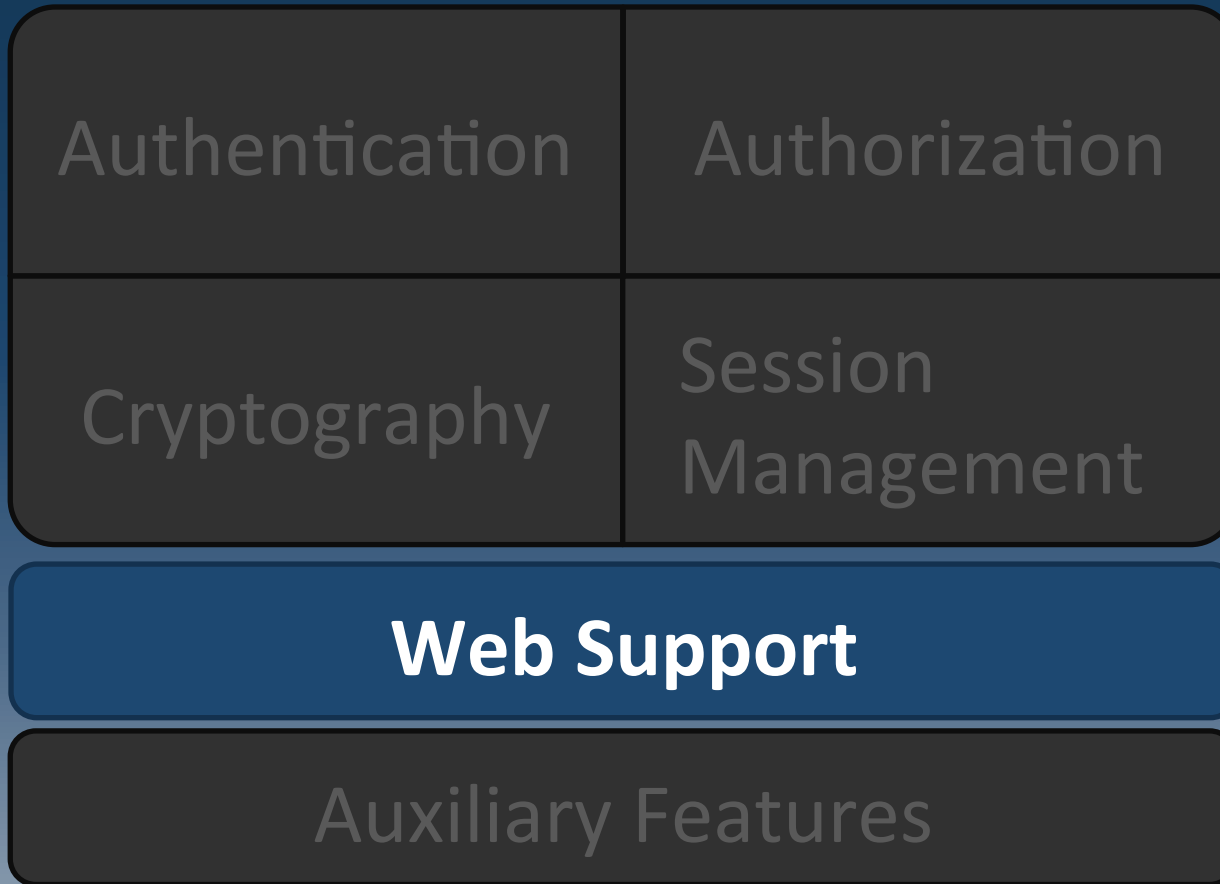
- Default interface implementations  
MD5, SHA1, SHA-256, et. al.
- Built in Hex & Base64 conversion
- Built-in support for Salts and repeated hashing



# Intuitive OO Hash API

```
//some examples:  
new Md5Hash("foo").toHex();  
  
//File MD5 Hash value for checksum:  
new Md5Hash( aFile ).toHex();  
  
//store password, but not plaintext:  
new Sha512Hash(aPassword, salt,  
               iterations).toBase64();
```

# Web Support



# Web Support Features

- Simple ShiroFilter web.xml definition
- Protects all URLs
- Innovative Filtering (URL-specific chains)
- JSP Tag support
- Transparent HttpSession support



# web.xml

```
<listener>
  <listener-class>
    org.apache.shiro.web.env.EnvironmentLoaderListener
  </listener-class>
</listener>

...

<filter>
  <filter-name>ShiroFilter</filter-name>
  <filter-class>org.apache.shiro.web.servlet.ShiroFilter
  </filter-class>
</filter>

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



# shiro.ini

```
[main]
ldapRealm = org.apache.shiro.realm.ldap.JndiLdapRealm
ldapRealm.userDnTemplate = uid={0},ou=users,dc=mycompany,dc=com
ldapRealm.contextFactory.url = ldap://ldapHost:389

securityManager.realm = $realm

[urls]
/images/** = anon
/account/** = authc
/rest/** = authcBasic
/remoting/** = authc, roles[b2bClient], ...
```



# JSP TagLib Authorization

```
<%@ taglib prefix="shiro"  
      uri="http://shiro.apache.org/tags" %>  
<html>  
<body>  
  <shiro:hasRole name="administrator">  
    <a href="manageUsers.jsp">  
      Click here to manage users  
    </a>  
  </shiro:hasRole>  
  <shiro:lacksRole name="administrator">  
    No user admin for you!  
  </shiro:lacksRole>  
</body>  
</html>
```



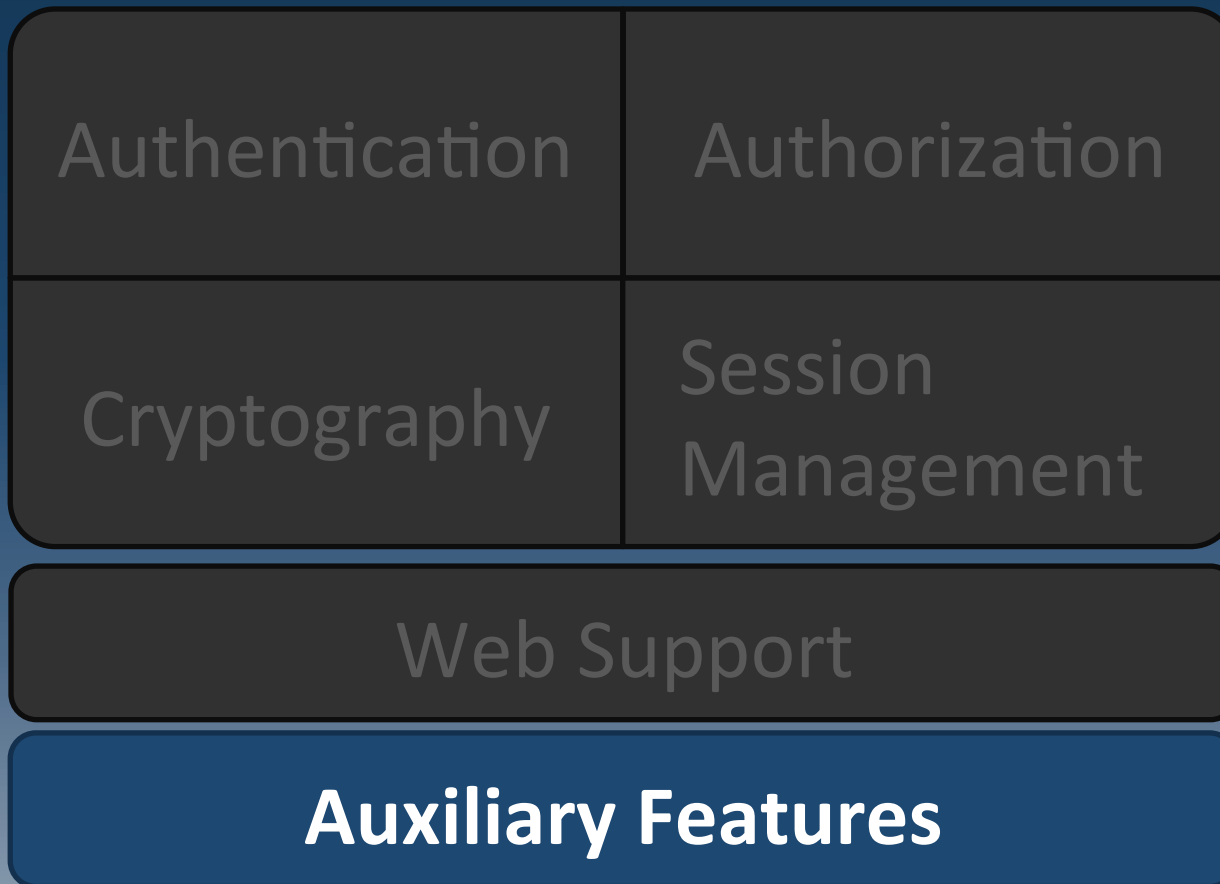
# JSP TagLibs

```
<%@ taglib prefix="shiro" uri=  
http://shiro.apache.org/tags %>
```

```
<!-- Other tags: -->  
<shiro:guest/>  
<shiro:user/>  
<shiro:principal/>  
<shiro:hasRole/>  
<shiro:lacksRole/>  
<shiro:hasAnyRoles/>  
<shiro:hasPermission/>  
<shiro:lacksPermission/>  
<shiro:authenticated/>  
<shiro:notAuthenticated/>
```



# Auxiliary Features



# Auxiliary Features

- Threading & Concurrency
  - Callable/Runnable & Executor/ExecutorService
- “Run As” support
- Ad-hoc Subject instance creation
- Unit Testing
- Remembered vs Authenticated

# Logging Out

```
//Logs the user out, relinquishes account  
//data, and invalidates any Session  
SecurityUtils.getSubject().logout();
```

## App-specific log-out logic:

Before/After the call

Listen for Authentication or StoppedSession events.



# CloudDirectory

First Cloud Identity Management system for *your* apps

- Features

  - Authentication & Access Control

  - Directory Support

  - Pre-Built Secure Workflows

- Sign-up for Beta: [CloudDirectory.com](http://CloudDirectory.com)

