ApacheCon 2014

Infinite Session Clustering with Apache Shiro & Cassandra

Les Hazlewood @lhazlewood
Apache Shiro Project Chair
CTO, Stormpath stormpath.com

#ApacheCon

Stormpath

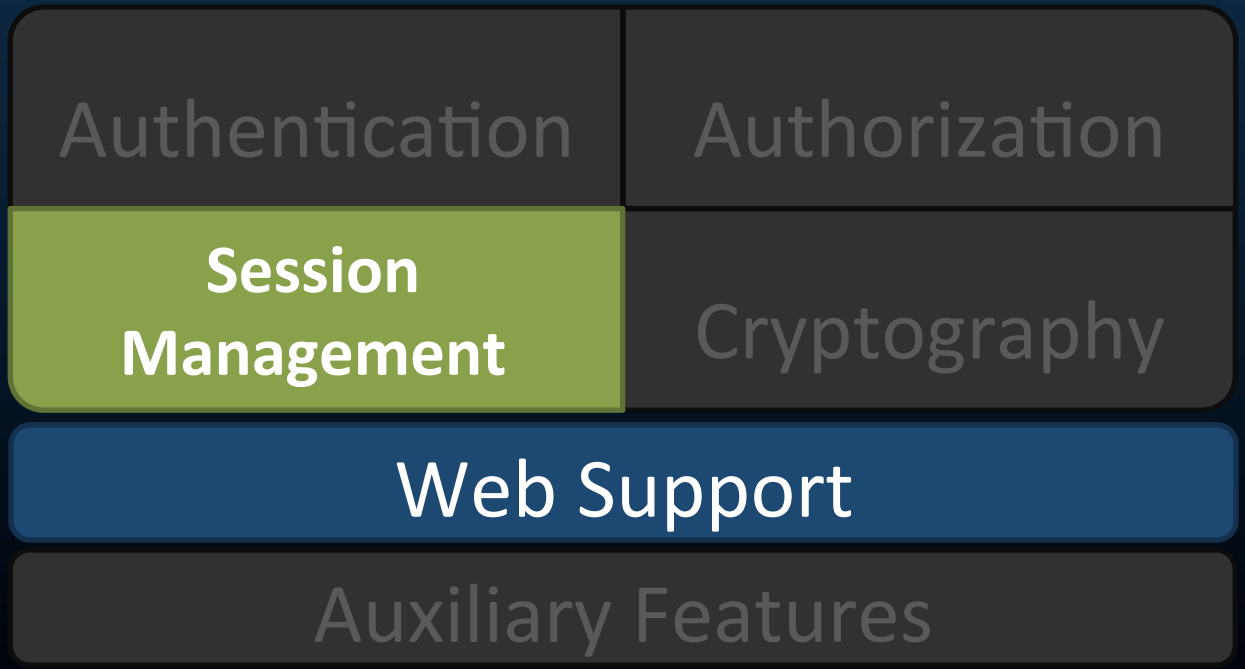
Stormpath .com

- User Management and Authentication API
- Security for ***your*** applications
- User security workflows
- Security best practices
- Developer tools, SDKs, libraries

What is Apache Shiro?

- Application security framework
- ASF TLP http://shiro.apache.org
- Quick and Easy
- Simplifies Security

Web Session Management



Quick Concepts

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

currentUser.login(...)
currentUser.isPermitted(...)
```

Session Management Defined

Managing the lifecycle of Subject-specific temporal data context

Session Management Features

- Heterogeneous client access
- POJO/J2SE based (IoC friendly)
- Event listeners
- Host address retention
- Inactivity/expiration support (touch())
- Transparent web use - HttpSession
- Container-Independent Clustering!

Acquiring and Creating Sessions

```
Subject subject =
    SecurityUtils.getSubject()

//guarantee a session
Session session = subject.getSession();

//get a session if it exists
subject.getSession(false);
```

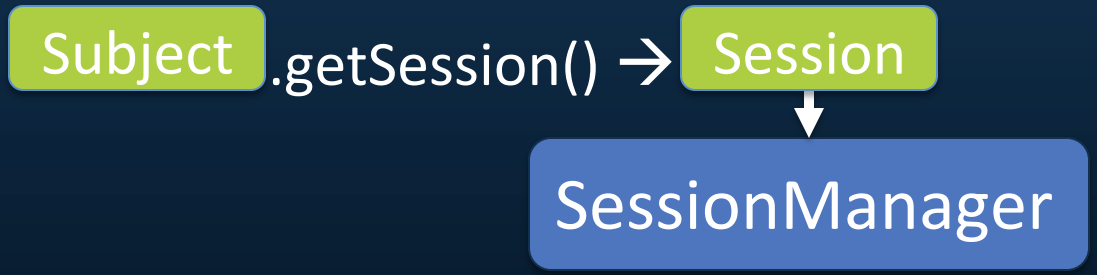
Session API

```
getStartTimestamp()

getLastAccessTime()

getAttribute(key)

setAttribute(key, value)

get/setTimeout(long)

touch()

...
```

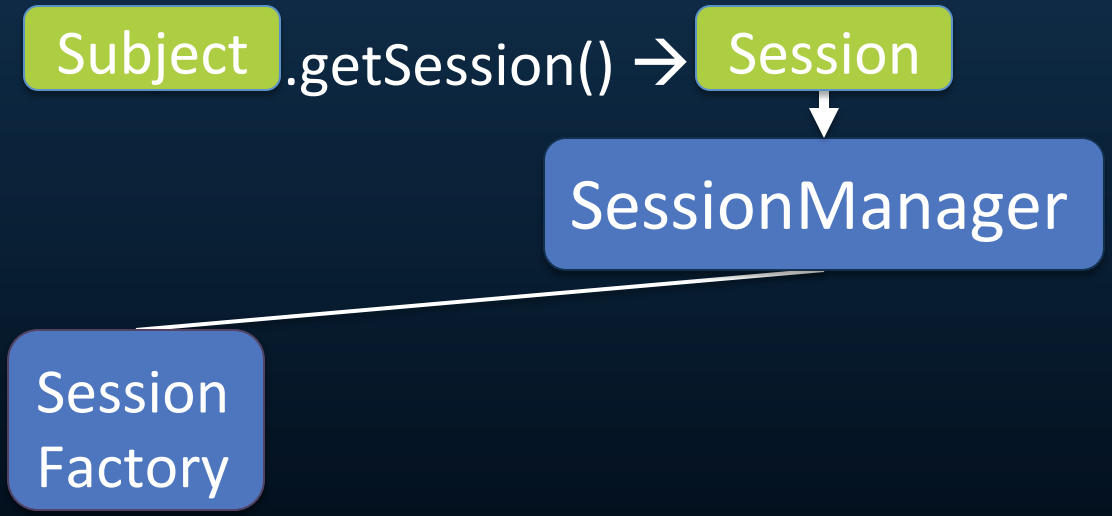
Session Management Architecture

Subject .getSession() → Session

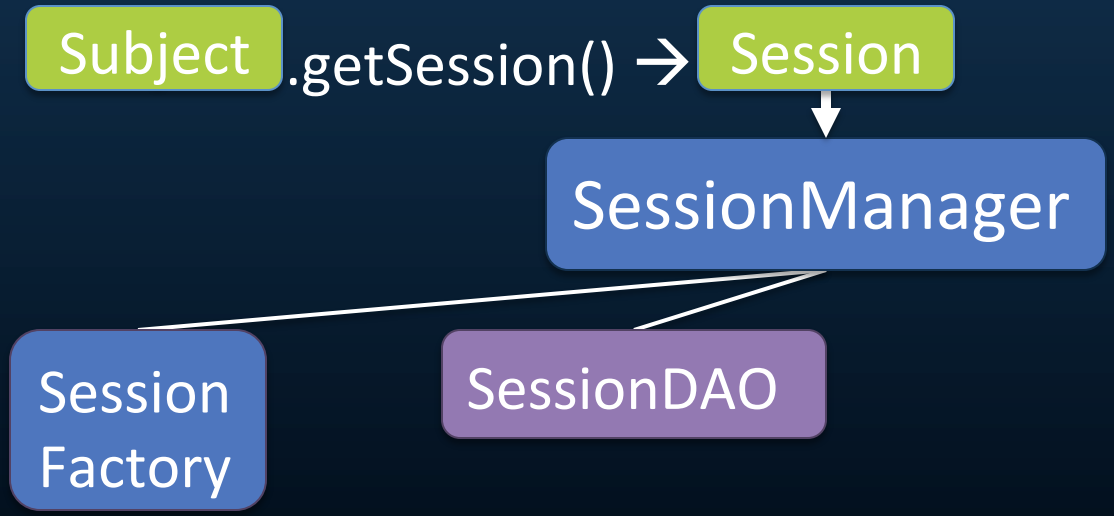
Session Management Architecture



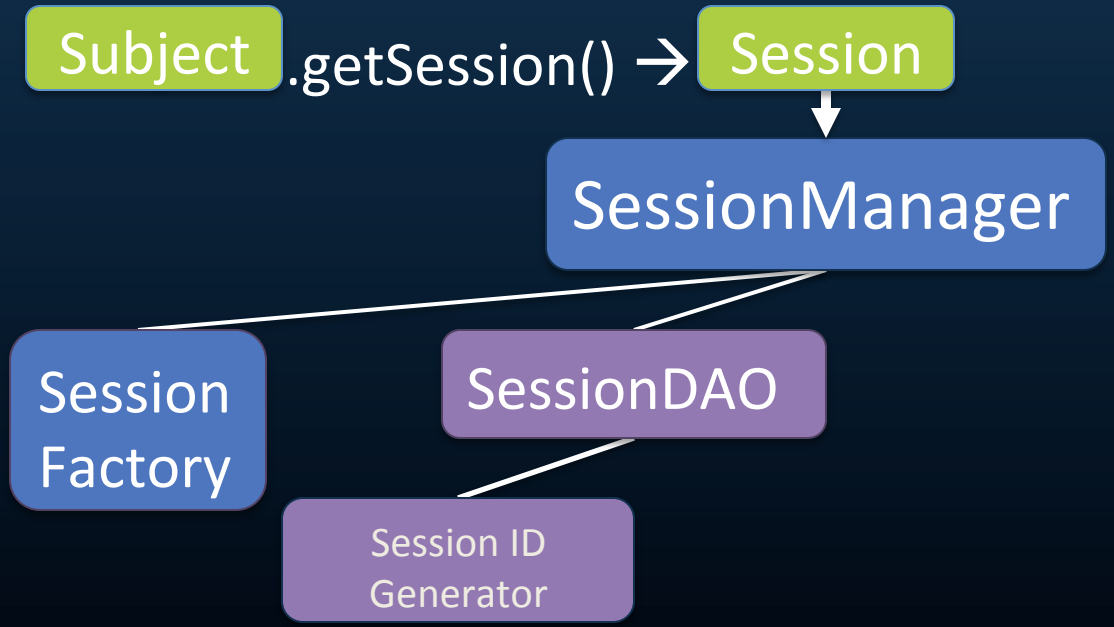
Session Management Architecture



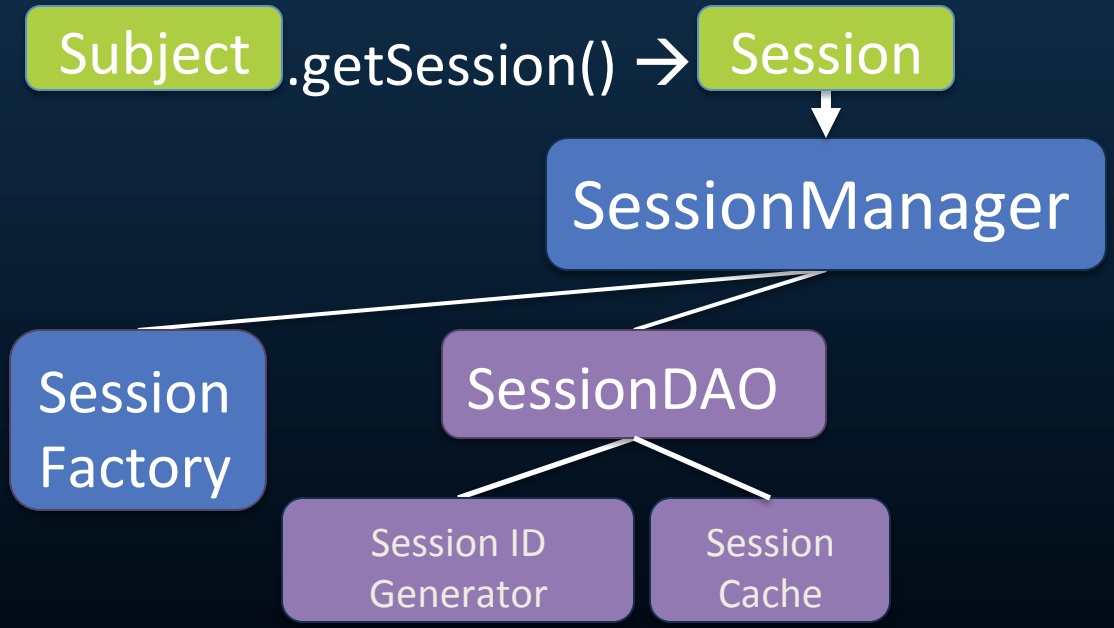
Session Management Architecture



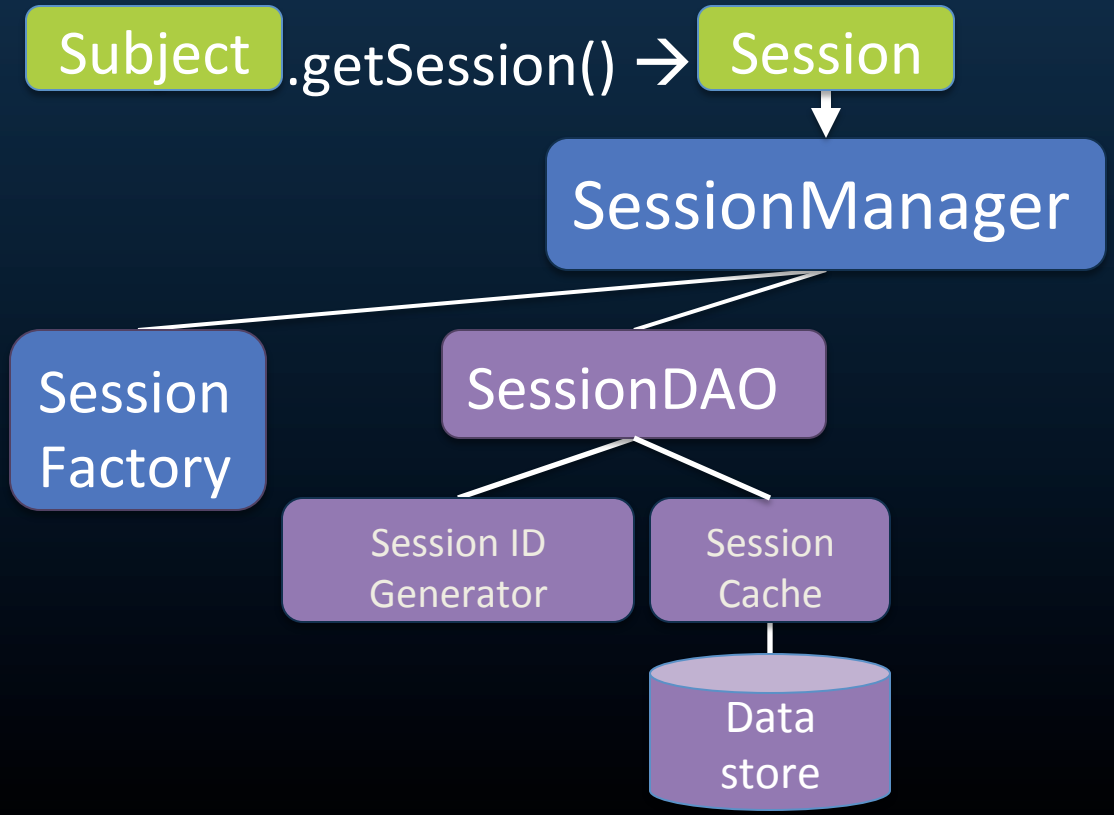
Session Management Architecture



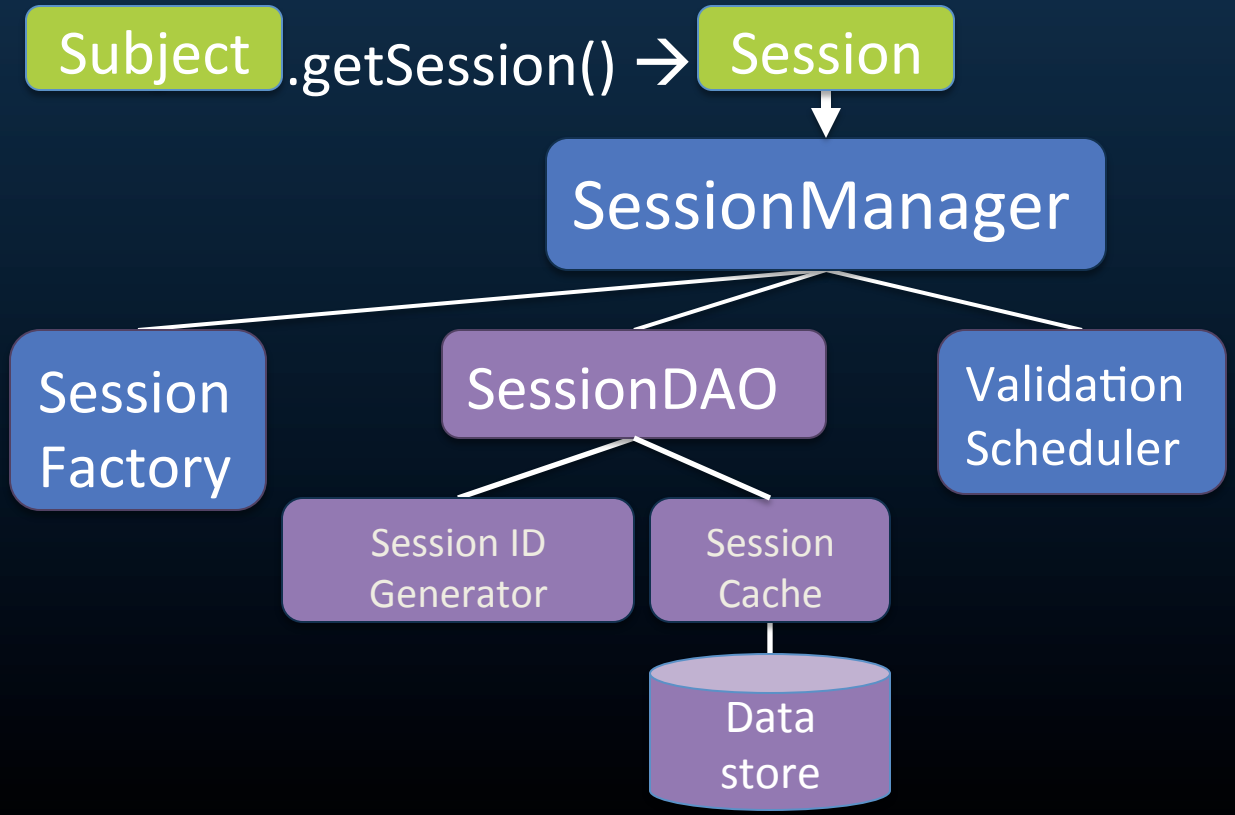
Session Management Architecture



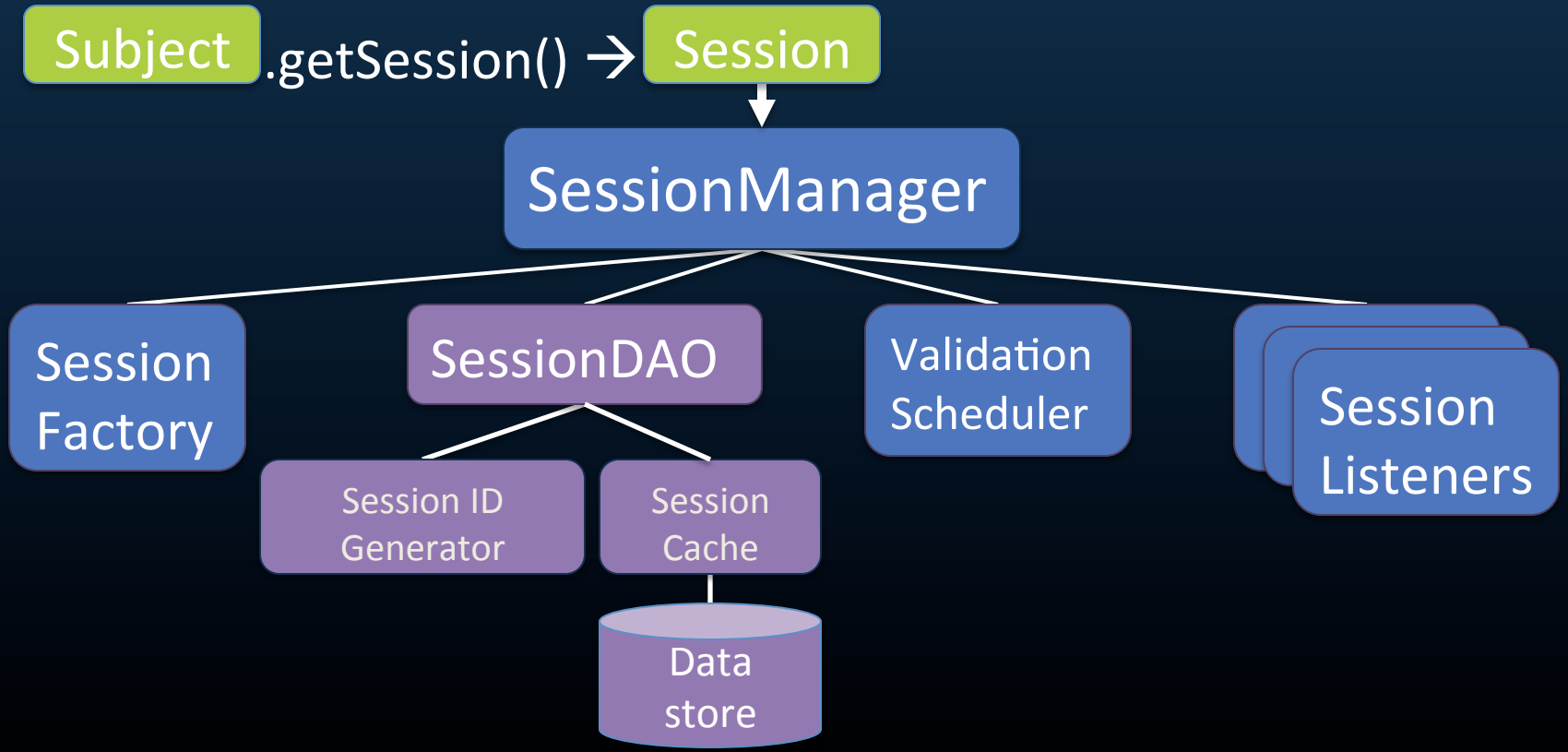
Session Management Architecture



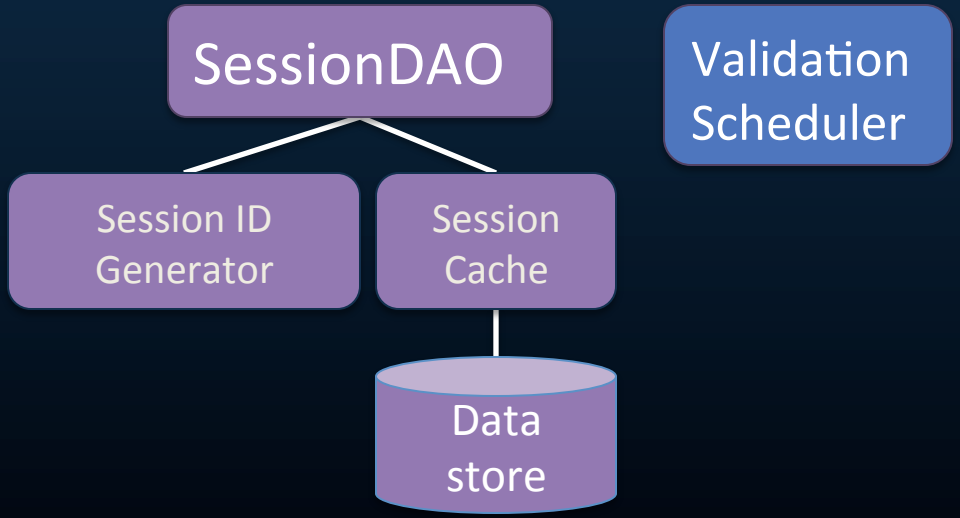
Session Management Architecture



Session Management Architecture



Session Clustering: Clustered Data Store of Choice



Web Configuration

- web.xml elements
- 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>
```

web.xml cont'd

```
<filter-mapping>
  <filter-name>ShiroFilter</filter-name>
  <url-pattern>/*</url-pattern>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
  <dispatcher>ERROR</dispatcher>
</filter-mapping>
```

shiro.ini overview

```
[main]
# bean config here

[users]
# optional static user accounts (and their roles) here

[roles]
# optional static roles (and their permissions) here

[urls]
# filter chains here
```

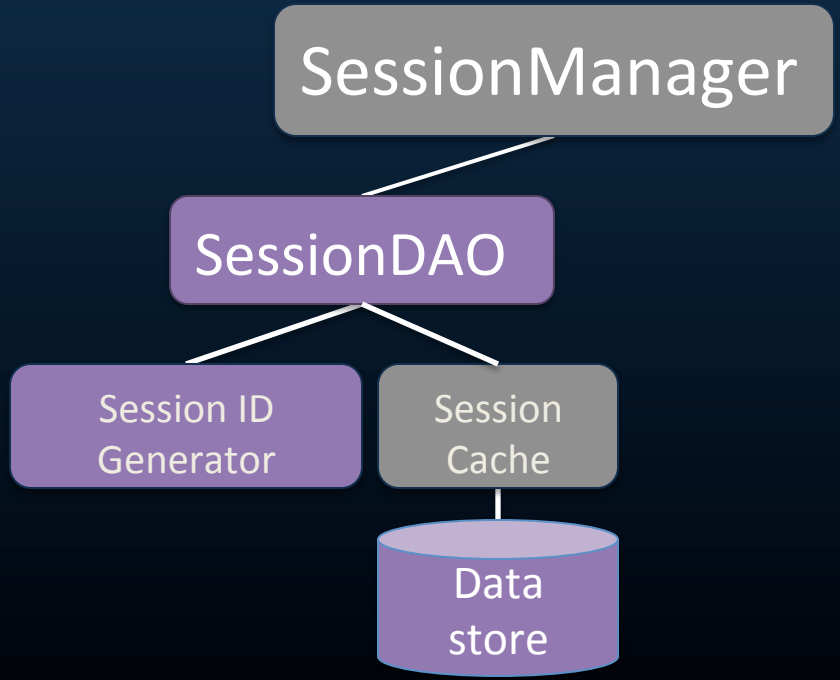
Session Clustering

Two Approaches

- Write a SessionDAO
- Use EnterpriseCacheSessionDAO and write a CacheManager

Cassandra SessionDAO

SessionDAO Concerns



Custom SessionDAO

```
public class MySessionDAO extends AbstractSessionDAO {
    protected void doCreate(Session s){...}
    protected void doReadSession(Serializable id){...}
    protected void delete(Session s){...}
    protected void update(Session s){...}
    Collection<Session> getActiveSessions(){...}
}
```

Or

```
public class MySessionDAO extends CachingSessionDAO {
... //enables write-through caching
}
```

Native Web Session Manager

```
[main]
sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
securityManager.sessionManager = $sessionManager
```

Cassandra SessionDAO

```
[main]
...

cassandraCluster = com.leshazlewood.samples.shiro.cassandra.ClusterFactory

sessionDAO = com.leshazlewood.samples.shiro.cassandra.CassandraSessionDAO
sessionDAO.cluster = $cassandraCluster
sessionDAO.keyspaceName = shirosessions
sessionDAO.tableName = sessions

...
```

Plug in the SessionDAO

```
[main]
...
sessionManager.sessionDAO = $sessionDAO
```

Sessions Table (CQL 3)

```
CREATE TABLE sessions (
    id timeuuid PRIMARY KEY,
    start_ts timestamp,
    stop_ts timestamp,
    last_access_ts timestamp,
    timeout bigint,
    expired boolean,
    host varchar,
    serialized_value blob
)
```

No Validation Scheduler?

No Validation Scheduler?

Use Cassandra's TTL

TTL for session timeout

```
[main]
# Cassandra can enforce a TTL.
# No need for Shiro to invalidate!

sessionManager.sessionValidationSchedulerEnabled = false
```

Session Upsert (CQL 3)

```
UPDATE sessions USING TTL $timeout SET
    start_ts = ?,
    stop_ts = ?,
    last_access_ts = ?,
    timeout = ?,
    expired = ?,
    host = ?,
    serialized_value = ?
WHERE
    id = ?
```

But what about tombstones!?!?

Sessions Table (revised)

```
CREATE TABLE sessions (
    id timeuuid PRIMARY KEY,
    start_ts timestamp,
    stop_ts timestamp,
    last_access_ts timestamp,
    timeout bigint,
    expired boolean,
    host varchar,
    serialized_value blob
) WITH gc_grace_seconds = 86400
  AND compacation = {'class':'LeveledCompactionStrategy'}
```

But what about row caching?

Row Cache?

Probably don't need it
(but maybe in some cases it would be useful)

- SSTable likely in Operating System page cache (off heap)
- DO use Key Cache (very important, enabled by default in 1.2)

Code

```
$ git clone https://github.com/lhazlewood/shiro-cassandra-sample.git

$ cd shiro-cassandra-sample

$ $CASSANDRA_HOME/bin/cassandra

$ mvn jetty:run

Open a browser to http://localhost:8080
```

Thank You!

- les@stormpath.com
- Twitter: @lhazlewood
- http://www.stormpath.com