Table Of Content

Package event

Class Summary

Event

A run-time representation of an Event persistent Object.

EventBuilder

A Builder class which creates new Events.

EventList

A class that aggregates Events.

EventListBuilder

A Builder which creates new EventList objects.

EventManager

EventManager, which is a Singleton which manages all the Event functions.

---

event

Class Event

```
java.lang.Object
  |
  +--event.Event
```

< Constructors >

---

public class **Event**
extends java.lang.Object

A run-time representation of an Event persistent Object. This class is used as an intermediary for creation, retrieval, and modification of Event data within the Java code (and the JVM). It is encodable (or serializable) to a database format (e.g., SQL Entry).

Constructors

Event

`protected  Event()`

Constructs a new Event class. Called through the EventBuilder class. Attribute assignations are done through protected scope.

---

event

Class EventBuilder

```
java.lang.Object
    |
    +--event.EventBuilder
```

< Constructors > < Methods >

---

public class **EventBuilder**
extends java.lang.Object

A Builder class which creates new Events. It is used to decouple the parts of the process of creating a new Event from the actual Event class, which is intended to only be a data wrapper class. Namely, this class implements the checks and validations necessary to create a valid Event and will reject invalid ones.

Constructors

EventBuilder

`public  EventBuilder()`

Creates a new EventBuilder to instantiate the new Event.

Methods

BuildEvent

`public Event BuildEvent()`

Finalizes the Event creation and returns it.

Returns:

returns the final Event.

Validate

`protected boolean Validate()`

Checks the current Event, returning True if it is valid so far and False otherwise.

Returns:

True, if the Event is valid so far. False if otherwise.

---

setDate

```
public EventBuilder setDate(java.lang.String date)
```

Adds a date value to the current Event.

Parameters:

date - the value to be added.

Returns:

the current builder.

---

setDescription

```
public EventBuilder setDescription(java.lang.String description)
```

Adds a description value to the current Event.

Parameters:

description - the value to be added.

Returns:

the current builder.

---

setEventtype

```
public EventBuilder setEventtype(java.lang.String eventType)
```

Adds a Event Type value to the current Event.

Parameters:

eventType - the value to be added.

Returns:

the current builder.

---

setHost

```
public EventBuilder setHost(java.lang.String host)
```

Adds a host value to the current Event.

Parameters:

host - the value to be added.

Returns:

the current builder.

---

setLocation

```
public EventBuilder setLocation(java.lang.String location)
```

Adds a location value to the current Event.

Parameters:

location - the value to be added.

Returns:

the current builder.

---

setName

```
public EventBuilder setName(java.lang.String name)
```

Adds a name value to the current Event.

Parameters:

name - the value to be added.

Returns:

the current builder.

---

setTime

```
public EventBuilder setTime(java.lang.String time)
```

Adds a time value to the current Event.

Parameters:

time - the value to be added.

Returns:

the current builder.

---

setVisibility

```
public EventBuilder setVisibility(java.lang.String visibility)
```

Adds a visibility value to the current Event.

Parameters:

visibility - the value to be added.

Returns:

the current builder.

---

event

Class EventList

```
java.lang.Object
    |
    +--event.EventList
```

< Constructors >

public class **EventList**
extends java.lang.Object

A class that aggregates Events.

Constructors

EventList

```
protected  EventList()
```

Constructs a new EventList class. Called through the EventListBuilder class. Attribute assignations are done through protected scope.

event

Class EventListBuilder

```
java.lang.Object
    |
    +--event.EventListBuilder
```

< Constructors > < Methods >

public class **EventListBuilder**
extends java.lang.Object

A Builder which creates new EventList objects. As other builders, it is used to decouple the process of creating a new EventList from the actual EventList class, and also provides functions implementing attribute-base filtering (e.g., filter by location, or by hosting organization, etc.)

Constructors

EventListBuilder

```
public  EventListBuilder()
```

Creates a new EventListBuilder to instantiate the new EventList.

Methods

setAddeventtolist

```
public EventListBuilder setAddeventtolist(java.lang.String addEventToList)
```

Adds an Event value to the current EventList.

Parameters:

addEventToList - the value to be added.

Returns:

the current builder.

---

event

Class EventManager

```
java.lang.Object
    |
    +--event.EventManager
```

< Constructors > < Methods >

---

public class **EventManager**
extends java.lang.Object

EventManager, which is a Singleton which manages all the Event functions. This class receives dispatched actions from the SOS Dispatcher and completes that action using objects internal to its subsystem. It also is in charge of interacting with the SOS Data Store Façade directly. Part of the role of this class is to parse front-end format data (e.g., JSON-String description of new Events) and calling the appropriate functions on other classes according to that data. It is also in charge of encoding Event objects into database-format (e.g., SQL Table entries). Another role is to create EventLists based on filter requests through the EventListBuilder.

Constructors

EventManager

```
protected  EventManager()
```

A protected or private constructor ensures that no other class has access to the Singleton.

Methods

cancelEvent

```
public void cancelEvent(int event_id)
```

Sets the is_cancelled property of the given Event to True.

Parameters:

event_id - the wanted Event.

---

createEvent

```
public Event createEvent(java.lang.String jsonString)
```

Creates a new Event from a json Event description. Done by calling the EventBuilder class.

Parameters:

jsonString - the JSON object describing the new Event.

Returns:

a Event object with the given attributes.

---

getEventDetails

```
public Event getEventDetails(int event_id)
```

Loads an Event with the given Event id.

Parameters:

event_id - the id of the wanted Event.

Returns:

the Event with the corresponding id.

---

instance

```
public static EventManager instance()
```

Returns:

The unique instance of this class.

---

markAttendance

```
public void markAttendance(int user_id,
                           int event_id)
```

Marks a User as attending an Event by creating an entry on the Attendance table.

Parameters:

user_id - the id of the User
event_id - the id of teh Event

---

retrieveListOfEvents

```
public EventList retrieveListOfEvents(int[] event_ids)
```

Retrieves a list of Events in the for of an EventList. This is done through an EventListBuilder.

Parameters:

event_ids - the ids of the Events to be added.

Returns:

the EventList containing the given Events.

Package organization

Class Summary

Organization

A run-time representation of an Organization persistent object.

OrganizationBuilder

A Builder which creates new Organization objects.

OrganizationLoader

A class which creates an Organization object from an Organization database object.

OrganizationManager

A Singleton which manages all the Organization functions.

---

organization

Class Organization

```
java.lang.Object
   |
   +--organization.Organization
```

< Constructors >

public class **Organization**
extends java.lang.Object

A run-time representation of an Organization persistent object. This class is used as an intermediary for creation, retrieval, and modification of Organization data within the Java code (and the JVM). It is encodable (or serializable) to a database format (e.g., SQL Entry)

Constructors

Organization

```
protected  Organization()
```

Constructs a new Organization class. Called through the OrganizationBuilder class. Attribute assignations are done through protected scope.

organization

Class OrganizationBuilder

```
java.lang.Object
    |
    +--organization.OrganizationBuilder
```

< Constructors > < Methods >

public class **OrganizationBuilder**
extends java.lang.Object

A Builder which creates new Organization objects. It is used to decouple the process, including validations and checks, of creating an Organization from the actual Organization class itself.

Constructors

OrganizationBuilder

`public  OrganizationBuilder()`

Creates a new OrganizationBuilder to instantiate the new Event.

Methods

CreateNewOrganization

`public Organization CreateNewOrganization()`

Finalizes the Organization creation and returns it.

Returns:

returns the final Organization.

ValidateOrganizationDetails

`protected boolean ValidateOrganizationDetails()`

Checks the current Organization, returning True if it is valid so far and False otherwise.

Returns:

True, if the Organization is valid so far. False if otherwise.

setDescription

```
public OrganizationBuilder setDescription(java.lang.String description)
```

Adds a description value to the current Organization.

Parameters:

description - the value to be added.

Returns:

the current builder.

---

setName

```
public OrganizationBuilder setName(java.lang.String name)
```

Adds a name value to the current Organization.

Parameters:

name - the value to be added.

Returns:

the current builder.

---

setPrivacy

```
public OrganizationBuilder setPrivacy(java.lang.String privacy)
```

Adds a privacy value to the current Organization.

Parameters:

privacy - the value to be added.

Returns:

the current builder.

---

setRequirements

```
public OrganizationBuilder setRequirements(java.lang.String requirements)
```

Adds a requirements value to the current Organization.

Parameters:

requirements - the value to be added.

Returns:

the current builder.

organization

Class OrganizationLoader

```
java.lang.Object
    |
    +--organization.OrganizationLoader
```

< Constructors > < Methods >

public class **OrganizationLoader**
extends java.lang.Object

A class which creates an Organization object from an Organization database object. This class decouples the parsing from the database to the system logic from the OrganizationManager class and can be extended to include internal checks for data integrity purposes.

Constructors

OrganizationLoader

```
public  OrganizationLoader()
```

Methods

LoadOrganization

```
public static Organization LoadOrganization(java.lang.String sqlEntry)
```

Creates a Organization from a database-format entry.

Parameters:

sqlEntry - a sql entry for the given organization.

Returns:

a Organization object with the given attributes.

organization

Class OrganizationManager

```
java.lang.Object
    |
    +--organization.OrganizationManager
```

< Constructors > < Methods >

public class **OrganizationManager**
extends java.lang.Object

A Singleton which manages all the Organization functions. This class receives dispatched actions from the SOS Dispatcher and completes that action using objects internal to its subsystem. It also is in charge of interacting with the SOS Data Store Façade directly. Part of the role of this class is to parse front-end format data (e.g., JSON-String description of new Organization) and calling the appropriate functions on other classes according to that data. Another job of this class is to manage Role creation and assignment, as well as mediate the modification of data in an Organization, and of Event hosting.

Constructors

OrganizationManager

```
protected  OrganizationManager()
```

A protected or private constructor ensures that no other class has access to the Singleton.

Methods

grantRole

```
public void grantRole(int userId,
                      int orgId,
                      int[] privIds)
```

Grants a number of privileges to a User for a given Organization.

Parameters:

userId - the unique id of the User
orgId - the unique id of the Organization
privIds - the unique ids of the Privileges given to the User.

---

instance

```
public static OrganizationManager instance()
```

Returns:

The unique instance of this class.

Package security

Class Summary

AccessManager

A Singleton dealing with access control actions.

PasswordManager

A Singleton which deals with password control actions.

security

Class AccessManager

```
java.lang.Object
    |
    +--security.AccessManager
```

< Constructors > < Methods >

public class **AccessManager**
extends java.lang.Object

A Singleton dealing with access control actions. It implements most of the back-end side of the access policy for SOS and host the relevant Enumerations for access permissions and other privileges. It also must be called to do checks on the relevant actions, such as creating events, deleting profiles, etc.

Constructors

AccessManager

```
protected  AccessManager()
```

A protected or private constructor ensures that no other class has access to the Singleton.

Methods

CheckPrivileges

```
public boolean CheckPrivileges()
```

Returns:

The result of privilege check for the current user class.

instance

```
public static AccessManager instance()
```

Returns:

The unique instance of this class.

---

security

Class PasswordManager

```
java.lang.Object
   |
   +--security.PasswordManager
```

< Constructors > < Methods >

---

public class **PasswordManager**
extends java.lang.Object

A Singleton which deals with password control actions. It implements most of the back-end side of the password policy for SOS, including resolving passwords and checking the input password against the database.

Constructors

PasswordManager

```
protected  PasswordManager()
```

The constructor could be made private to prevent others from instantiating this class. But this would also make it impossible to create instances of PasswordManager subclasses.

Methods

EncryptPassword

```
public static java.lang.String EncryptPassword(java.lang.String password)
```

Parameters:

password - is a String to be validated

Returns:

will return an encrypted version of the password as a String

ValidateLogInCredentials

```
public static boolean ValidateLogInCredentials(java.lang.String username,
                                               java.lang.String pwd)
```

Parameters:

username - is the user name for log in
pwd - is the user's password for log in

Returns:

is the validation of the login credentials

ValidatePassword

```
public static boolean ValidatePassword(java.lang.String password)
```

Parameters:

password - as a String to be validated

Returns:

is true if password successfully validates

---

instance

```
public static PasswordManager instance()
```

Returns:

The unique instance of this class.

Package sosInterface

Class Summary

SOSDispatcher

A Command class which propagates the front-end requests to their specific target controllers.

SOSServer

SOSServer communicates with the front-end for creation of events.

SOSSessionManager

SOSSessionManager keeps track of each session for every user.

---

sosInterface

Class SOSDispatcher

```
java.lang.Object
    |
    +--sosInterface.SOSDispatcher
```

< Constructors > < Methods >

public class **SOSDispatcher**
extends java.lang.Object

A Command class which propagates the front-end requests to their specific target controllers. The requests messages which are parsed and pre-processed by the SOS Server then are used to call an appropriate dispatch from the SOS Dispatcher, which is in charge of directly calling all other controllers. Instead of being their own classes, each subcommand is defined in terms of parametrizations of the dispatch function within the SOS Dispatcher. Internally, SOS Dispatcher also keeps track of these requests and stores them in the Database

Constructors

SOSDispatcher

```
public  SOSDispatcher()
```

Methods

Dispatch

```
public void Dispatch()
```

The method for dispatching events.

GetAllEvents

```
public java.util.ArrayList GetAllEvents()
```

This method returns all of the events.

Returns:

is an ArrayList contain of Strings of events.

sosInterface

Class SOSServer

```
java.lang.Object
    |
    +--sosInterface.SOSServer
```

< Constructors > < Methods >

public class **SOSServer**
extends java.lang.Object

SOSServer communicates with the front-end for creation of events. Also it is held responsible for managing user sessions and keeping track of them, as well as dispatching messages through the system.

Constructors

SOSServer

```
protected  SOSServer()
```

The constructor could be made private to prevent others from instantiating this class. But this would also make it impossible to create instances of SOSServer subclasses.

Methods

CreateEvent

```
public static boolean CreateEvent(java.lang.String event)
```

This method creates an event and stores its data.

Parameters:

event - is a String coming from the front-end including a JSON which stores all of the event details, including name, type, location, etc.

Returns:

is true if event is successfully created and false otherwise.

ParseMessage

```
public static java.lang.String ParseMessage(java.lang.String jsonString)
```

This method parses a message coming from the front-end which supposed to be in JSON format.

Parameters:

jsonString - is a String coming from the front-end including the JSON

Returns:

is the parsed message

---

instance

```
public static SOSServer instance()
```

Returns:

The unique instance of this class.

---

send

```
public static void send(java.lang.String action,
                        java.lang.Object payload)
```

Dispatch an action with the parameters defined in the payload.

Parameters:

action - the action to be dispatched.
payload - the payload of the action.

---

sosInterface

Class SOSSessionManager

```
java.lang.Object
    |
    +--sosInterface.SOSSessionManager
```

< Constructors > < Methods >

---

public class **SOSSessionManager**
extends java.lang.Object

SOSSessionManager keeps track of each session for every user. It can gather information about the current session, update the session, or even destroy it.

Constructors

SOSSessionManager

```
protected  SOSSessionManager()
```

The constructor could be made private to prevent others from instantiating this class. But this would also make it impossible to create instances of SOSSessionManager subclasses.

Methods

DestroySession

```
public static boolean DestroySession(java.lang.String sessionID)
```

This method destroys a given user session

Parameters:

sessionID - is the ID for the session to be destroyed

Returns:

is true if DestroySession is successful and false otherwise.

GetCurrentSession

```
public static java.lang.String GetCurrentSession()
```

This method returns the information regarding a current live session

Returns:

is a String containing current session's ID.

GetSessionInformation

```
public static java.lang.String GetSessionInformation(java.lang.String
sessionID)
```

This method returns the information regarding a given session.

Parameters:

sessionID - is the ID for the desired session

Returns:

is a String containing current session's information.

LogOutUser

```
public static boolean LogOutUser()
```

This method logs out the user from their current session.

Returns:

is true if Logout is successful and false otherwise.

UpdateSessionInformation

```
public static boolean UpdateSessionInformation(java.lang.String data,
                                               java.lang.String sessionID)
```

This method updates the given session with the modified data

Parameters:

sessionID - is the ID of the session to be updated
data - is the modified data to be updated on the given session

Returns:

is true if update information is successfull and false otherwise.

instance

```
public static SOSSessionManager instance()
```

Returns:

The unique instance of this class.

Package user

Class Summary

NewUserBuilder

A Builder which creates new User objects.

User

A run-time representation of a User persistent object.

UserLoader

A class which creates a User object from a database-format User object (e.g., a SQL Table entry for User).

UserManager

A Singleton class which managers all the User functions.

UserUpdater

A class which deals with User modifications.

---

user

Class NewUserBuilder

```
java.lang.Object
   |
   +--user.NewUserBuilder
```

< Constructors > < Methods >

public class **NewUserBuilder**
extends java.lang.Object

A Builder which creates new User objects. It is used to decouple the parts of the process of creating a new User from the actual User class, which is intended to only be a data wrapper class which can be easily parsed into the database format. Namely, this class implements the checks and validations necessary to create a valid User and will reject invalid ones. As part of this validation, it must interact with the SOS Security System classes that implement the password and access policies.

Constructors

NewUserBuilder

```
public  NewUserBuilder()
```

Creates a new NewUserBuilder to instantiate the new User.

Methods

BuildUser

`public User BuildUser()`

Finalizes the User creation and returns it.

Returns:

returns the final User.

ValidateCredentials

`protected boolean ValidateCredentials()`

Checks the current User, returning True if it is valid so far and False otherwise.

Returns:

True, if the Event is valid so far. False if otherwise.

---

setEmail

`public NewUserBuilder setEmail(java.lang.String email)`

Adds a email value to the current User.

Parameters:

email - the value to be added.

Returns:

the current builder.

---

setName

`public NewUserBuilder setName(java.lang.String name)`

Adds a name value to the current User.

Parameters:

name - the value to be added.

Returns:

the current builder.

---

setPassword

```
public NewUserBuilder setPassword(java.lang.String password)
```

Adds a password value to the current User.

Parameters:

password - the value to be added.

Returns:

the current builder.

---

setPrivacy

```
public NewUserBuilder setPrivacy(java.lang.String privacy)
```

Adds a privacy value to the current User.

Parameters:

privacy - the value to be added.

Returns:

the current builder.

---

setUsername

```
public NewUserBuilder setUsername(java.lang.String username)
```

Adds a username value to the current User.

Parameters:

username - the value to be added.

Returns:

the current builder.

---

user

Class User

```
java.lang.Object
    |
    +--user.User
```

< Constructors >

---

public class **User**
extends java.lang.Object

A run-time representation of a User persistent object. This class is used as an intermediary for creation, retrieval, and modification of User data within the Java code (and the JVM). It is encodable (or serializable) to a database format (e.g., SQL Entry).

Constructors

User

```
protected  User()
```

Constructs a new User class. Called through the UserBuilder class. Attribute assignations are done through protected scope.

---

user

Class UserLoader

```
java.lang.Object
   |
   +--user.UserLoader
```

< Constructors > < Methods >

---

public class **UserLoader**
extends java.lang.Object

A class which creates a User object from a database-format User object (e.g., a SQL Table entry for User). This class decouples the parsing from the database to the system logic from the UserManager class and can be extended to include internal checks for data integrity purposes.

Constructors

UserLoader

```
public  UserLoader()
```

Methods

LoadOrganization

```
public static User LoadOrganization(java.lang.String sqlEntry)
```

Creates a User from a database-format entry.

Parameters:

sqlEntry - a sql entry for the given organization.

Returns:

a User object with the given attributes.

---

user

Class UserManager

```
java.lang.Object
    |
    +--user.UserManager
```

< Constructors > < Methods >

---

public class **UserManager**
extends java.lang.Object

A Singleton class which managers all the User functions. This class receives dispatched actions from the SOS Dispatcher and completes that action using objects internal to its subsystem. It also is in charge of interacting with the SOS Data Store Façade directly. Part of the role of this class is to parse front-end format user data (e.g., JSON-String defining a new User) and calling the appropriate functions on the other classes according to that data. It also is in charge of encoding a User object into database format objects (e.g., SQL Table entry for User).

Constructors

UserManager

```
protected  UserManager()
```

A protected or private constructor ensures that no other class has access to the Singleton.

Methods

ChangeUserDetails

```
public void ChangeUserDetails(User user,
                              java.util.Map change)
```

Updates a User object with the given changes. Done through the UserUpdater class.

Parameters:

user - the User that will be updated.
change - a Map where the key is the variable name and the value is the update.

CreateNewProfile

```
public User CreateNewProfile(java.lang.String jsonString)
```

Creates a new User from a json User description. Done by calling the NewUserBuilder class.

Parameters:

jsonString - the JSON object describing the new User.

Returns:

a User object with the given attributes.

LoadUser

```
public User LoadUser(java.lang.String sqlEntry)
```

Creates a User from a database-format entry. Done by calling the UserLoader class.

Parameters:

sqlEntry - a sql entry for the given user.

Returns:

a User object with the given attributes.

---

instance

```
public static UserManager instance()
```

Gives the instance of the UserManager, or creates one if none exists.

Returns:

the unique instance of this class.

---

user

Class UserUpdater

```
java.lang.Object
  |
  +--user.UserUpdater
```

< Constructors > < Methods >

---

public class **UserUpdater**
extends java.lang.Object

A class which deals with User modifications. User modifications are done on the system logic-level User object first and are only finalized once they are stored to the database. The UserUpdater decouples these modifications from the UserManager class and from the User class itself and implements checks and validations in the same way that NewUserBuilder does. It also ensures that every modification to the User class is saved to the SOS Data Store.

Constructors

UserUpdater

```
public  UserUpdater()
```

Methods

ChangeUser

```
public void ChangeUser(User user,
                       java.util.Map update)
```

Updates a User object with the given changes.

Parameters:

user - the User that will be updated.
change - a Map where the key is the variable name and the value is the update.

INDEX