1. **Resource Bundle:**

The ResourceBundle class provides a mechanism to globalize the messages. The hardcoded message is not considered good in terms of programming, because it differs from one country to another. So we use the ResourceBundle class to globalize the massages. The ResourceBundle class loads this information from the properties file that contains the messages.

* **Commonly used methods of ResourceBundle class**

There are many methods in the ResourceBundle class. Let's see the commonly used methods of the ResourceBundle class.

* **public static ResourceBundle getBundle (String basename)** returns the instance of the ResourceBundle class for the default locale.
* **public static ResourceBundle getBundle (String basename, Locale locale)** returns the instance of the ResourceBundle class for the specified locale.
* **public String getString (String key)** returns the value for the corresponding key from this resource bundle.

1. **Lightweight vs heavy weight:**

Light weights frameworks are very less complex when compared to heavy weight frameworks. They provide an easy way to use them. The light weight frameworks do not require more code changes to incorporate as it does. Whereas the heavy weight requires more interfaces, and sometimes the code may be tightly coupled.

Light weight refers to less code where as heavy code refers to more code

1. **Mutable vs immutable objects:**

Mutable objects are the objects whose values could be altered after the construction. Whereas the immutable objects are the objects which could not be altered, once constructed.

Example,

String s=new String(“immutable string”);

System.out.println(s);// prints “immutable string”

s.replaceAll(“immutable”,”mutable”);

System.out.println(s);// prints immutable string”

Here the string value does not change even we alter the value. This is known as immutable object.

Example,

Int a=10;

a=a+10;

System.out.println(a);// prints 20 as output.

1. **Wrapper classes in java:**

Wrapper classes in java provides a mechanism of converting primitive type to object type. The conversion of primitive to object is known as autoboxing and vice versa is known as unboxing.

|  |  |
| --- | --- |
| **Primitive type** | **Wrapper class** |
| boolean | Boolean |
| char | Character |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |

**Primitive to wrapper example:**

public class PrimitiveToWrapper

{

public static void main (String args [])

{

int a=10;

Integer i=Integer.valueOf (a);

System.out.println(a+” “+i);

}

}

**Wrapper to primitive example:**

public class WrapperToPrimitive

{

public static void main (String args [])

{

Integer i=new Integer (3);

int a=i. intValue ();

System.out.println(i+” “+a);

}

}

1. **Thread safety:**

Thread safety in java is a process to make our program safe to use in multithreaded program. A piece of code is thread safe if it only manipulates shared data structure in a manner that guarantees safe execution by multiple threads at a same time.

1. **System Properties:**

Java maintains a set of system properties for its operations. Each java system property is a key value pair. We can retrieve all the system properties as System.getProperties() or System.getProperties(key).

mail.smtp.host is a system property that tells the smtp server that is to be connected.

1. **Java mail session:**

The main runtime interface between a java application and hibernate. The main function of session is to offer create, read and delete operations for instances of mapped entity classes.

The session class represents a mail session. It collects together properties and defaults used by the mail API’s. The session class provides access to the protocol providers that implements the store, transport and related classes.

Session.getDefaultInstance(Properties file) – reads the default instance

Session.getDefaultInstance(Properties file, Authenticator) –

The authenticator represents an object that knows how to obtain authentication for a network connection. It authenticates the password by the following method.

Protected passwordAuthentication getPasswordAuthentication () – this is called when password authentication is used. It returns the parameterized constructor that take the values as sender and password.

1. **MIME message:**

MIME stands for Multipurpose Internet Mail Extensions. It allows documents other than plain text, to be attached to an email message. The MimeMessage class represents a mime style email message. We can create a mime style messaging that will instantiate an empty MimeMessage object and then fill it with appropriate attributes and content. Some of the important mime message methods are as follows.

|  |  |
| --- | --- |
| **Method** | **Description** |
| MimeMessage.setFrom(Address address) | Set RFC 822 ‘FROM’ header field |
| addReceipnt(Message.receipntType type, Address address) | Add the given address to the specified recipient type. |
| setSubject(String subject) | Set the subject header field. |

The RFC 822 represents a standard format for electronic messages which consistes of set of header fields such as sender, recipient etc.

1. **Transport:**

There are many methods in the transport api. One of them is transport.send(message) that will send the message to the specified recipient.

1. **MessagingException:**

This exception is thrown when the connect method on a store or transport object fails due to an authentication failure.

1. **Properties class:**

The properties object contains key and value pair both as a string. It can be used to get property value based on the property key. The properties class provides methods to get data from properties file and store data into properties file. Some of the properties are as below.

|  |  |
| --- | --- |
| **Method** | **Description** |
| Mail.smtp.host | The smtp server to connect |
| mail.smtp.socketFactory.port | Specifies the port to connect when using the specified socket factory. If not set the default port will be used. |
| mail.smtp.sockectFactory.class | Specifies the class name that implements socketFactory. The class will be used to create the smtp sockets. |
| mail.smtp.auth | If true is set as value, then attempts to authenticate the user using AUTH command. The default value is false. |
| mail.smtp.port | The smtp port to connect to. If not specified connects to 25 port. |

1. **Hibernate annotations:**

**@Entity:**

Typically, an entity represents a table in relational database. When a pojo class is marked with @Entity, it tells that the class is an entity bean.

@Entity(name=”name”) used to name the Entity

@Table(name=”name”) used to name the table in database.

**@Id:**

This annotation represents the primary key of the entity. Generally, the primary key is auto generated using the annotation @GeneratedValue.

**@Column:**

The column annotation is used to specify the details of the column to which a field or property will be mapped. The attributes of column annotations are

Name- name of the column

Length – specifies the size of the column

Nullable – specifies the columns to mark as not null

Unique – specifies to be unique.

**@OneToMany:**

This annotation is used to create the one to many relationships between the pojo’s. for example, a student has number of phone numbers. It takes a parameter that maps to the pojo.

**@ManyToOne**

This annotation is used to create the many to one relationships between the pojo’s. for example, many students have the same address.

**@JoinTable(name=”name”):**

This represents that OneToMany is mapped using a join column named specified in the name attribute. It represents the foreign key in the current table.

1. **HttpClient:**

The HttpClient library for java is designed to work with any Http services on the web. HttpClient is a library that allows to handle Http requests. We receive and send request and response through HttpClient class by creating instance of this class with the following statement.

CloseableHttpClients clients=HttpClient.createDefault();

**HttpGet:** The Get method means retrieve whatever information is identified by request URL. It has several methods such as HttpGet (), HttpGet (URI url),

**HttpPost:** post method is used to send data to the server.

**Put:** put is also used to send data to a server. The main differences are that, **put** produces the same responses even though it is called multiple times. Whereas **post** produces side effects of creating same responses multiple times when called multiple times.

**HTTP response code:** All http response codes are separated into five classes. They are as follows.

1xx (informational) – the request was received, continuing process.

2xx (Successful) – the request was successfully received, understood and accepted.

3xx (Redirection) – further action needs to be taken in order to complete the request.

4xx (Client Error) – the request contains bad syntax or cannot be fulfilled.

5xx (server error) – the server failed to fulfill an apparently valid request.

1. **Json**

**ObjectMapper:**

Object mapper provides functionality for reading and writing Json either to and from pojo class or to and from a general purpose json tree model.

ObjectMapper.readTree (new BasicResponseHandler (). handleResponse (response))

This method will deserialize the JSON content as a tree expressed as JsonNode instances.

**JsonNode:**

Base class for all the JsonNodes. JsonNode represents both the Json object and json array. It facilitates us in easy parsing of json data.

1. **SimpleDateFormat:**

Simple date format is a concrete class for formatting and parsing date.

1. **Hibernate concepts:**

An instance of configuration allows the application to specify the properties and mapping documents to be used when creating a session factory. Usually an application will create a single configuration, build a single instance of SessionFactory.