**FINAL REPORT**

**INTERSHIP IN CS**

Hello Professor,

In this Report, I want to explain about the remaining topics, which I gained knowledge in the internship Program. Well I am not going to cover the topics, which I mentioned in my previous reports.

**AGENDA:**

* **Spring AOP**
* **SOAP**
* **Cloud Computing**
* **Amazon Web services**

**Spring AOP:**

Before going to learn about Spring AOP, what was the necessity of this module in the spring framework? AOP stands for Aspect Oriented Programming. Before AOP we have, Functional Programming and then we got Object Oriented Programming After these two here comes the AOP.

**Functional Programming:**

If we write code in some of the older programming languages like C, we would think of the problem and break it down into different functions. Each function accomplishes a particular unit of work and then a function can call other functions and that is how they talk to each other and a single program consists of all these different functions then they call each other when the last function ends then the program is completed.

**Problem:** If a complex function is involved, there is a whole lot of interdependencies function. Therefore, we go for Object Oriented Programming.

**Object Oriented Programming:**

In this, we will not think of functions when we are trying to solve the problem by writing code. We would think of individual entities as objects and we write objects that mirror different entities in our problem space. Each object would contain the member variables as well as methods. Therefore, we have encapsulated entities where we can design code that is more complex because we have a cleaner design and separation of concerns.

**Problem:** Not all of the tasks that we would want program want to do can classified into objects.

* Too many relationships to the crosscutting objects.
* Code is still required in all methods.
* Not all can change at once.

**Cross Cutting Concerns:**

Some functionalities that need to be used by different objects and they may not be a part of the problem domain. It could be Infrastructure related, Security.

Common Cross cutting Concerns:

* Logging
* Transactions
* Security

To overcome these problems In AOP, we will identify what are the cross cutting concerns and write as aspects and then configure where the aspects apply.

**Web Services:**

**Application Programming Interface (API):**

Defines everything we need to communicate with a Web Service.

**Elements of a Web Service API:**

**Message Format:**

* SOAP, XML, JSON (Java Script Object Notation), Other

**Request Syntax:**

* Named Methods: They use syntax similar to client side.
* Uniform Resource Identifier (URI): They sent us http request.
* Parameter names and data types.

**Requesting an action:**

* Named Methods
* HTTP Verbs (POST, GET, PUT, PATCH, DELETE).

**Authentication:**

* Username and Password
* Authentication tokens

**Receiving the data:**

* Formats: SOAP, XML, JSON
* Metadata: Describes data structure, including field/ property names and data types.

We have two kinds of Web services.

1. **SOAP** (Simple Object Access Protocol) which implements **JAX-WS**.
2. **REST** (Representational State Transfer) which implements **JAX-RS**.

The classes can access the method and if we want to provide any feature to user then we will have an MVC module that makes the web application to the users. MVC module will have a call to the method and then it can show the list of products to user in a nice HTML format. The following are some of the Key components of Web Service.

* **WSDL:**

It describes the web service in an XML format. So, those different technologies can understand.

* **UDDI:**

It is a directory where any publisher can publish their web services and consumer can query this directory and get access to all the different web services.

* **SOAP:**

It is a Protocol language in XML format, which used to encode and decode different messages. If we are making a call to Web Service, it ends up as a SOAP message that transmitted over the network.

* **SEI:**

Interface to a Web Service that provides a way for the client application irrespective of the technology to call the Web service.

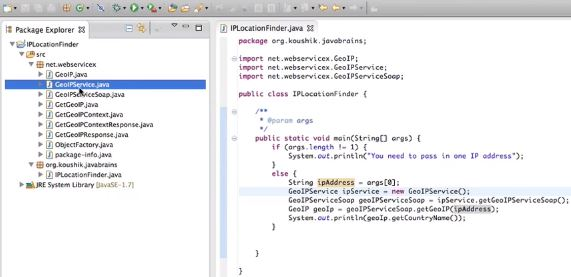
**Steps to Generate Stubs from WSDL:**

* Open the Command prompt
* Create a new directory i.e. Mkdir sei
* Now change the directory to sei i.e. Cd sei
* Import all the files i.e. wsimport –keep –s src “url”

In the SEI folder, create a SRC folder after in which the execution of wsimport command it generates the Java class in SRC folder and class type in net folder.

After writing the code, project name: IPLocation Finder. For this, we need to pass input arguments like IP Address of Google. To get Google IP address type ping Google.com in command window. Copy that IP address and paste in Run Configurations-> Arguments->Run.

Fig shown below is the Output.



The important concepts in the SOAP based Web Service is:

* **JAX-B Annotations**

@xmlRootElement (), @xmlType (proporder= {}), @xmlElement (), @webmethod, @webResult ()

We need to have a noArg Constructor because we need a way for JAX-B to initialize our new instance of the class i.e. it will need to instantiate the object first and if we have only one constructor with some arguments. JAX-B does not know what arguments to pass the things. JAX-B does XML to Java Object Conversion tool.

* **Handling Faults**

In a method if something should not return the return type as mentioned, this can done when the method throws an exception.

* **SOAP UI**

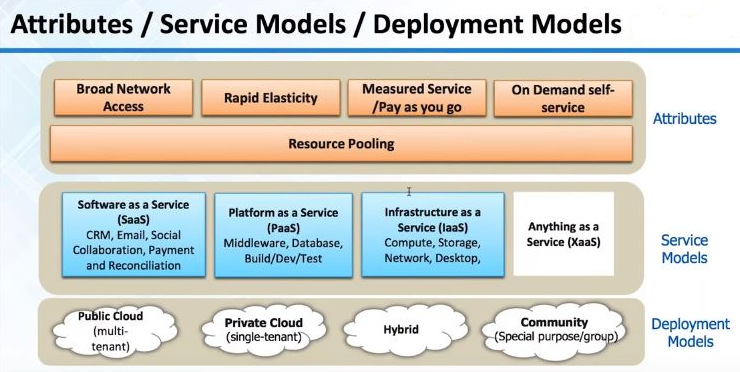
-It is one of the tools to test the SOAP Web Services.

-It is widely used by lot of QA professionals.

-It is an Open Source.

**Cloud Computing:**

Before Cloud Computing, it was very difficult for start-ups. All the innovation we see in the world is in large part because of Cloud Computing. It made so cheap and easy to experiment in the ways that was possible before and lot more activity in the technology space.



**Attributes:** Key characteristics of cloud computing

* **Broad Network Access**:

We can essentially connect to infrastructure elements from any type of client. It is readily available on Internet.

* **Rapid Elasticity:**

Key characteristic, Stretchable

* **Measured Service:**

It is charged on hourly based or per GB on monthly based.

**Service Models:**

* **SAAS:**

It is for end users and they use only software.

* **PAAS:**

It is a service for developers. Platforms make it very easy to deploy the application and they hide the entire complexity infrastructure from us. It is one layer below Saas.

* **IAAS:**

It is a Raw Infrastructure. A server with 1GB Ram or storage with 1TB can request on hourly bases or on monthly bases.

Most Saas offerings will be probably using a Paas or an Iaas.

In software, if we have an application we can deploy it on infrastructure as a service like AWS. So then our offering become like Saas. They do rather depend on each other. Saas can run on Iaas or Paas. Paas offering can run on Infrastructure as a service.

**Deployment Models:**

* **Public cloud(Multi-tenant):**

Originally Cloud computing was public cloud. It is available on the Internet and Infrastructure shared across many companies and developers. If we are running an application most lightly, hardware shared with other developers or companies in a virtualized manner.

* **Private cloud(Single-tenant):**

Cloud computing industry is maturing right and they are trying to go and make larger companies like Financial, Health care companies as customers. Based on the demand in these industries there is a concept of Private cloud. Where essentially a single-tenant, the infrastructure is only for one company so there is no chance of data, being stole or leaked but we still have cloud characteristics like Elasticity, pay as you go.

* **Hybrid:**

In some cases, there is a possibility of hybrid. Let me take an example: If we are a bank, our core systems like accounting and code banking system, lot of sensitive information, credit scores, day-to-day payment information those has to be secured and these can remain in private cloud. However, certain other applications like learning management system, internet data and public website those can be in public cloud. Therefore, there are some systems and software is which can make it possible for us to see cloud in hybrid fashion.

**Amazon Web services (AWS):**

* Amazon Web Services (AWS) is a collection of remote computing services also called web services that make up a cloud-computing platform.
* These services are operated from 11 geographical regions across the world and operated using zones and edge locations.

**Overview of AWS Products:**

Compute, Storage, Database, Application services, Deployment and Management, Networking.

**AWS Free Tier:**

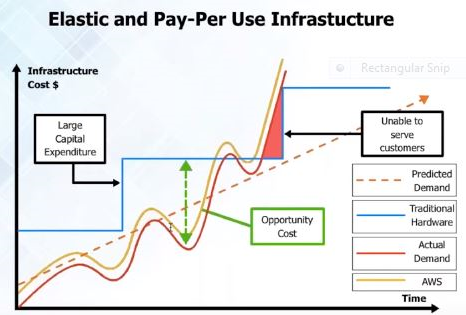
1 year free tier with all services.

Physical resources like processor memory shared. Virtualization was the original big innovation in the infrastructure side because it allowed sharing our hardware across multiple application or multiple operating systems. AWS works in a similar way that is why AWS is cheap because we do share infrastructure across many companies and developers. In AWS, it is possible to have infrastructure that which is dedicated to our computer.

**Elastic and pay per use Infrastructure:**

Blue line is the traditional hardware. Normally we over estimate what we need, we spend too much upfront. As the application sees some growth, we again have jump in investment like stepladder. However, sometime if we are slow to procure usage may go up faster than the capacity rate. Quality of service suffers i.e. pink shaded patch.

If we see AWS it follows the actual demand very closely because of Elasticity feature we can start servers, change the size of servers at the click of button. It is possible to save a lot of money.



**Regions, Zones and Edge locations:**

* AWS has lot of Regions, Zones and Edge locations.
* AWS is essentially an isolated set of Infrastructure. There is no connection between one region and other. Within a region, there are availability zones that are isolated pieces of infrastructure.
* Edge locations, which are for, content Delivery. Let us take an example: If we have YouTube like application where we need to deliver heavy media video, audio images then we need edge locations. Depending on where the user is the closest edge location used to send information to the user.
* Primary Infrastructure is for Regions and Zones.

**Difference between Open stack and AWS:**

Open stack is an open source cloud-computing platform especially it is the software layer. It is a platform that allows creating own infrastructure. Whereas, AWS is not cloud management platform but it is also underlying infrastructure.

Open stack does not give the server capacity or Database capacity, Storage, Networking. With open stack if we have our own server or own datacenter we can create cloud like capabilities with open stack. Especially, good for creating Private cloud. However, AWS is cloud management plus the actual infrastructure.

IBM purchased a company called soft layer, which is also a very good offering.

**Secure:**

AWS has ISO 27000 certification, SAAS 270 certification. We can encrypt data at Rest and secure data over the network. Now AWS has private cloud offerings as well. In Public cloud, Infrastructure shared across customers. However, in Private cloud we can choose that infrastructure used for us.

**Various Offerings from Amazon Web Services (AWS):**

**Compute:**

These things require CPU and RAM i.e. typically virtual type of server offerings.

* **EC2**: Virtual servers in the cloud.
* **EC2 Container Service:** Run and Manage Docker Containers.
* **Elastic Beanstalk**: Run and Manage Webapps.
* **Lambda:** Run code in Response to Events.

**Storage and content Delivery:**

Similar to SAN or NAS and Tape drive from which we can take back up.

* **S3:** Scalable Storage in the cloud.
* **Elastic File System:** Fully Managed File system for EC2.
* **Storage Gateway:** Integrate On-premises IT Environment with cloud storage.
* **Cloud Front:** Global content Delivery Network.
* **Glacier:** Archive Storage in cloud.
* **Import/Export Snowball:** Large-scale data transport.

**Database:**

Offerings whether it is Relational Database or NOSQL Database also things like Data ware house, Cache servers.

* **RDS:** Managed Relational Database Service.
* **ElastiCache:** In-memory cache.
* **Redshift:** Managed petabyte-scale Data ware house services.
* **DynamoDB:** Predictable and scalable NOSQL Data store.
* **DMS:** Managed Database Migration Service.

**Security and Identity:**

We can create User ID’s control that has to access what.

* **Identity and Access Management:** Manage User Access and Encryption Keys.
* **Inspector:** Analyze Application Security.
* **Directory Service:** Host and Manage Active Directory.
* **WAF:** Filter Malicious Web Traffic.

**Developer Tools:**

* **Code Commit:**  Store code in private GIT Repositories.
* **Code Pipeline:** Release software using Continuous Delivery.
* **Code Deploy:** Automate Code Deployments.

**Networking:**

In this, we have TNS servers, Virtual Private cloud. These are secured isolated cloud resources.

* **VPC:** Isolated Cloud Resources.
* **Direct Connect:** Dedicated Network Connection to AWS.
* **Route53:** Scalable DNS and Domain Name Registration.

**Management Tools:**

Stuff for management things like cloud watch which is used to monitor Infrastructure like Storage running out, is the server alive or not.

* **Cloud Watch:** Monitor Resources and Applications.
* **Cloud Trial:** Track User Activity and API Usage.
* **Service Catalog:** Create and use Standardized Products.
* **CONFIG:** Track Resource Inventory and Changes.
* **Cloud Formation:** Create and Manage Resources with Templates.
* **OpsWorks:** Automate Operations with chef.
* **Trusted Advisor:** Optimize Performance and security.

**Analytics:**

In this phase, we have Hadoop offerings so that we can run 50 or 100 nodes by using **EMR** (Elastic Map Reduce).

* **Data pipeline:** For Data-Driven workflows.
* **EMR:** Managed Hadoop Framework.
* **Elastic Search Service:** Run and scale Elastic Search Clusters.
* **Machine Learning:** Build smart Applications quickly and easily.
* **Kinesis:** Work with Real-time Streaming data.

**Mobile Services:**

* **Cognito:** User Identity and Application Data Synchronization.
* **Mobile Analytics:** Collect, View and Export Application Analysis.
* **Mobile Hub:** Build, Test and Monitor Mobile application.
* **Device Farm:** Test Android, Fire OS and ios applications on real devices in the cloud.
* **SNS:** Push Notification Service.

**Internet of Things:**

* **AWS LOT:** Connect devices to the cloud.

**Enterprise Applications:**

* **Workspaces:** Desktops in the cloud.
* **WorkMail:** Secure Email and Calendaring Service.
* **WorkDocs:** Secure Enterprise Storage and Sharing Service.

**Application Services:**

These are Helpers, which can send Email and Notification.

* **App Stream:** Low Latency application streaming.
* **Elastic Transcoder:** Easy-to-use Scalable Media Transcoding.
* **API Gateway:** Build, Deploy and Manage API’s.
* **CloudSearch:** Managed search service.
* **SQS:** Message Queue Service.
* **SES:** Email Sending Service.
* **SWF:** Workflow Service for coordinating application components.

**COMPUTE EC2:**

* Web service that enables to laugh and manage server instance.
* Instance available in different sizes and configuration.
* Use only the capacity you need and pay only what you use.
* Increase the capacity dynamically when the demand raises and vice versa.
* Supports both Vertical and Horizontal Scaling.
* Pre-built AMI available from market place.
* Have various pricing models like On-Demand, Spot and Reserved Instances.

**EBS-Elastic Block Storage:**

* Provides high availability and highly reliable storage volumes.
* Suitable for database, file system or raw block storage.
* Created volume can attached to any EC2 instance.
* Multiple volumes can attach to one instance.
* Snapshot of the volume has to be taking for backup and sharing.

**Steps:**

The following are the steps for how to create, launch and connect to a server in AWS.

1. **Choose an Amazon Machine Image (AMI)**

An AMI is a template that contains the software configuration (operating system, application server and applications) required to launch your instance. We can select an AMI provided by AWS, our user community, or the AWS Marketplace, or we can select one of our own AMIs.

1. **Choose an Instance Type**

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage and networking capacity that give you the flexibility to choose the appropriate mix of resources for our applications.

1. **Configure Instance Details**

Configure the instance to suit our requirements and can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign and access management role to the instance.

1. **Add Storage**

Instance will launched with the storage device settings. We can attach additional EBS volumes and instance store volumes to our instance.

1. **Tag Instance**

A tag consists of a case-sensitive key-value pair.

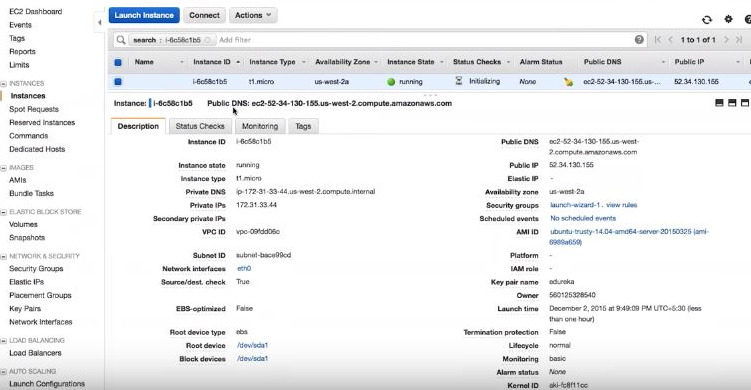
1. **Configure Security Group**

A security group is a set of firewall rules that control the traffic of our instance. We can add rules to allow specific traffic to reach our instance.

1. **Review Instance Launch**

**Output:**

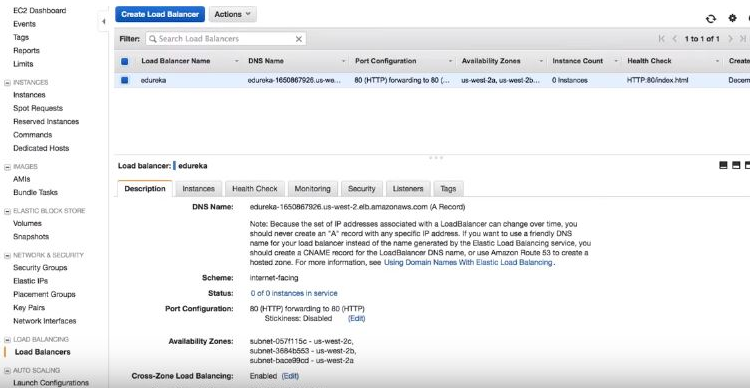
Server is created in AWS using EC2 and EBS is shown below.

****

**Steps:**

The following are the steps involved in creating a load balancer.

1. Define Load Balancer
2. Assign Security Groups
3. Configure Security Settings
4. Configure Health Check
5. Add EC2 Instances
6. Add Tags
7. Review



**Auto Scaling:**

Two steps to create Auto Scaling:

1. **Create launch configuration:**

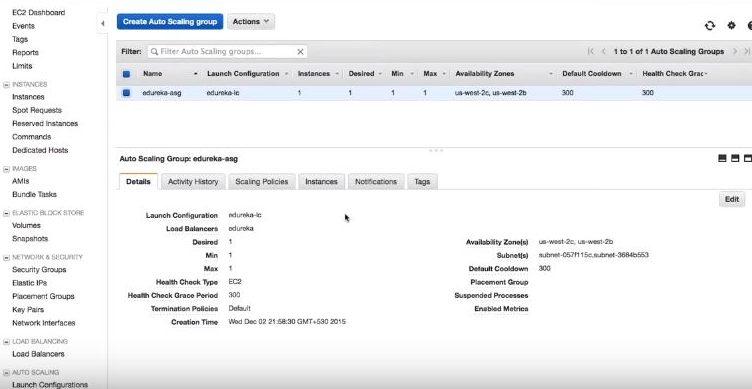
In the launch configuration, we tell to Auto Scaling group what the machine needs to use whether to launch or not. Size of the server it may be Micro server or large server.

Choose AMI->Choose Instance Type->Configure details->Add Storage->Configure Security Group-> Review.

1. **Create Auto Scaling Group**

It is a group where you define the minimum or maximum and what servers will be added on what conditions and when it has to be removed.

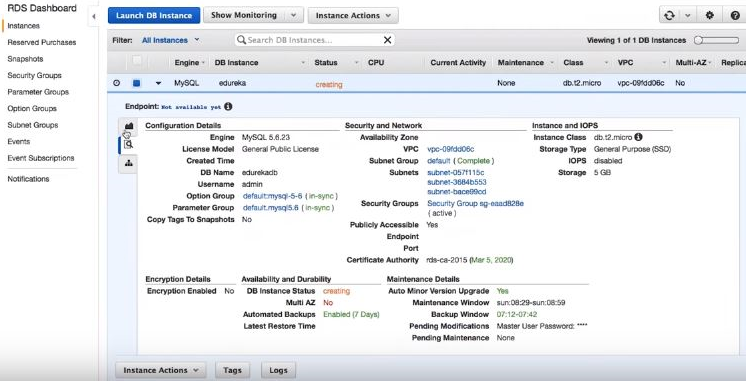
Configure Auto Scaling group details-Configure scaling policies-> Configure Notifications-> Configure Tags-> Review.



**RDS – Relational Database Service**

* Web Service that makes it easy to set up, operate and scale a relational database in cloud.
* Low-level database admin work handled automatically by AWS.
* Supports PostgreSQL, MySQL, MSSQL and Oracle database.
* Amazon RDS automatically patches the database software.
* Backup the database for a user defined retention period.
* Enables point in time recovery.

**Output:**



In this report I did not shared any code part because it may look to messy and space consuming concern. Still I shared some of the output snapshots to show what work I have done. I did not cover all minor topics I just concentrated on the main purpose i.e. on my Agenda and I do not wanted to drag it too much. Therefore, I finish my report with these topics. I hope you like this report professor.

Overall, it was nice of time to spend so much of time in learning new things. Internship program went very well and I am sure that the knowledge I acquired in such a short period will be very useful in my career.

Thank you,

**SREEKAR GOUD ABBAGOUNI**

**ID: 700630631**

**SXA06310@UCMO.EDU**