Review

- Mid-Sem topics:
  - Coding Design Patterns
  - Dependency Injection
  - Database Integration
  - Architectural Design Patterns
  - Git
  - APIs

Coding design Patterns

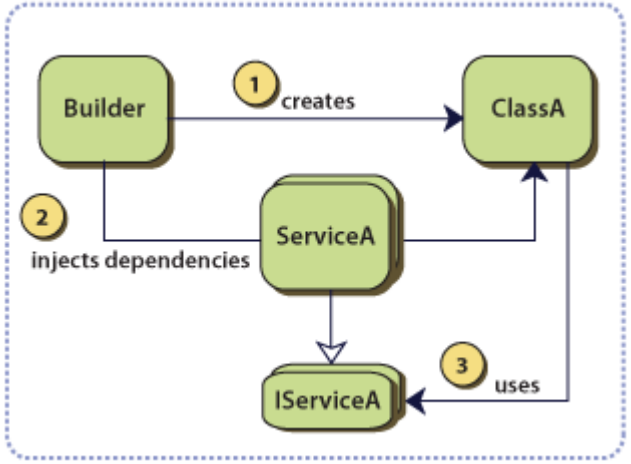
- What are they/ Why do we use them?
  - Description of communicating objects and classes to solve a problem.
  - Re-use
- What classes of coding design patterns are there?
  - Creational vs structural vs behavioural

- Can you give an example of a design pattern you have used?

```
package design.test;
import design.factory.ComputerFactory; import design.model.Computer;

public class TestFactory {

public static void main(String[] args) {

  System.out.print( “Do you want a Server? y/n" );
  String input = scanner.nextLine();
  if (input.equalsIgnoreCase(“y”))
    Computer comp = ComputerFactory.getComputer("server")
  else {
    System.out.print( “Enter HDD for custom PC?" );
    String my_hhd = scanner.nextLine();
    System.out.print( “Enter RAM for custom PC?" );
    String my_RAM = scanner.nextLine();
    System.out.print( “Enter CPU speed for custom PC?" );
    String my_cpu = scanner.nextLine();
    Computer comp = ComputerFactory.getComputer("pc“, my_hhd, my_RAM,
                                                my_cpu);
  }

  System.out.println(“chosen_computer::"+ comp.toString());
  }
}
```

Dependency Injection

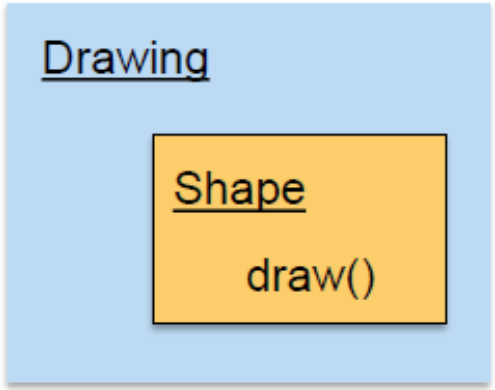
- Why is it valuable?
  - Avoid change class source code.
  - Avoid compile time specification of dependencies
  - Re-use

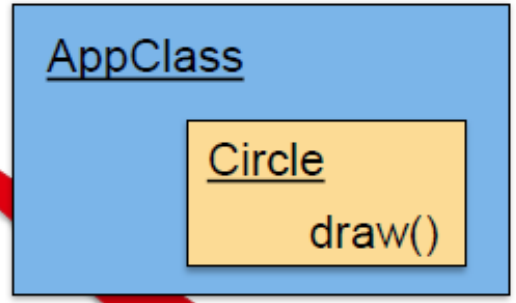


What we want

```
protected class Drawing {
        private Shape shape;
        public setShape(Shape shape) {
                this.shape = shape;
        }
        public drawShape() {
                this.shape.draw();
        }
}
```

```
Circle myCircle = new Circle():
drawing.setShape(myCircle);
drawing.drawShape();
```



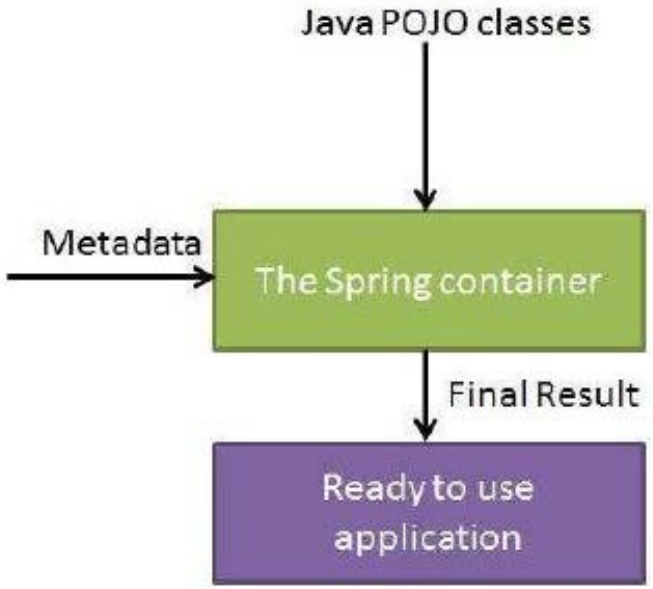


here the dependency is injected

different classes: the dependency has been separated

- Drawing class does not know about shape, only application class.
- However, it can accept any shape.
- Drawing class assumes something else will initialise it
- Application class is what creates the circle (not drawing)
- Can easily add new shapes, don't have to modify drawing class, just pass in new shape
  - Reason: dependency of drawing class to shape object not owned by drawing class
    - Instead dependency is injected by external entity

Java Containers



- Java containers manage Java objects: instantiation and lifecycle
- In this course, you will use :
  - Tomcat servlet container: hosts and processes web pages, such as HTML, JSP,etc.
  - Spring beans container: A bean is any *Plain Old Java Object* (POJO) , which can be used for:
    - Business components/services, such as AccountBean, BookingBeanetc.; or
    - Data Objects, such as Product, Contract etc.

Example

```
package com.tutorialspoint;

public class TextEditor {
   private SpellChecker spellChecker;

   // a setter method to inject the dependency.
   public void setSpellChecker(SpellChecker spellChecker) {
      System.out.println("Inside setSpellChecker." );
      this.spellChecker = spellChecker;
   }

   // a getter method to return spellChecker
   public SpellChecker getSpellChecker() {
      return spellChecker;
   }
   public void spellCheck() {
      spellChecker.checkSpelling();
   }
}
```

```
package com.tutorialspoint;

public class SpellChecker {
   public SpellChecker(){
      System.out.println("Inside SpellChecker constructor." );
   }
   public void checkSpelling(){
      System.out.println("Inside checkSpelling." );
   }
}
```

```
package com.tutorialspoint;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationCon

public class MainApp {
   public static void main(String[] args) {
      ApplicationContext context = new ClassPathXmlApplicationContext
      TextEditor te = (TextEditor) context.getBean("textEditor");
      te.spellCheck();
   }
}
```

Example

```
<?xml version = "1.0" encoding = "UTF-8"?>

<beans xmlns = "http://www.springframework.org/schema/beans"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation = "http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <!-- Definition for textEditor bean using inner bean -->
   <bean id = "textEditor" class = "com.tutorialspoint.TextEditor">
      <property name = "spellChecker">
         <bean id = "spellChecker" class = "com.tutorialspoint.SpellChecker"/>
      </property>
   </bean>

</beans>
```

```
Inside SpellChecker constructor.

Inside setSpellChecker.

Inside checkSpelling.
```

Dependency Injection

- What types of dependency injection are there?
  - Constructor-based dependency injection
  - Setter-based dependency injection
  - Interface injection

Database Integration

- Why do you need databases?
- How do you integrate databases with your application?
- What is the value of a DAO?
- How can you add to improve re-use?

```java
@Autowired
private SessionFactory sessionFactory;

@RequestMapping(value = "/hibernateAdd", method = RequestMethod.GET)
public String hibernateAdd(Locale locale, Model model) {

    Person p = new Person();
    p.setAge(20);
    p.setFirst("FirstName");
    p.setLast("lastName");

    sessionFactory.getCurrentSession().save(p);
    return "home";
}
```

ORM translates this to SQL

Database Integration

```
@Autowired
private PersonDao personDao;

@RequestMapping(value = "/hibernateDaoAdd", method = RequestMethod.GET)
public String hibernateDaoAdd(Locale locale, Model model) {
    Person p = new Person();
    p.setAge(20);
    p.setFirst("FirstName");
    p.setLast("lastName");

    personDao.savePerson(p);

    return "home";
}
```

```java
package au.edu.sydney.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import au.edu.sydney.dao.PersonDao;
import au.edu.sydney.domain.Person;

@Service(value="personService")
// @Transactional
public class PersonService {

    @Autowired
    private PersonDao personDao;

    // business logic of registering a Person into the database
    public void registerPerson(Person person) {

        // Step 1: check whether this person is already in the database

        // Step 2: if not, save this person into the database
        personDao.savePerson(person);
    }

}
```

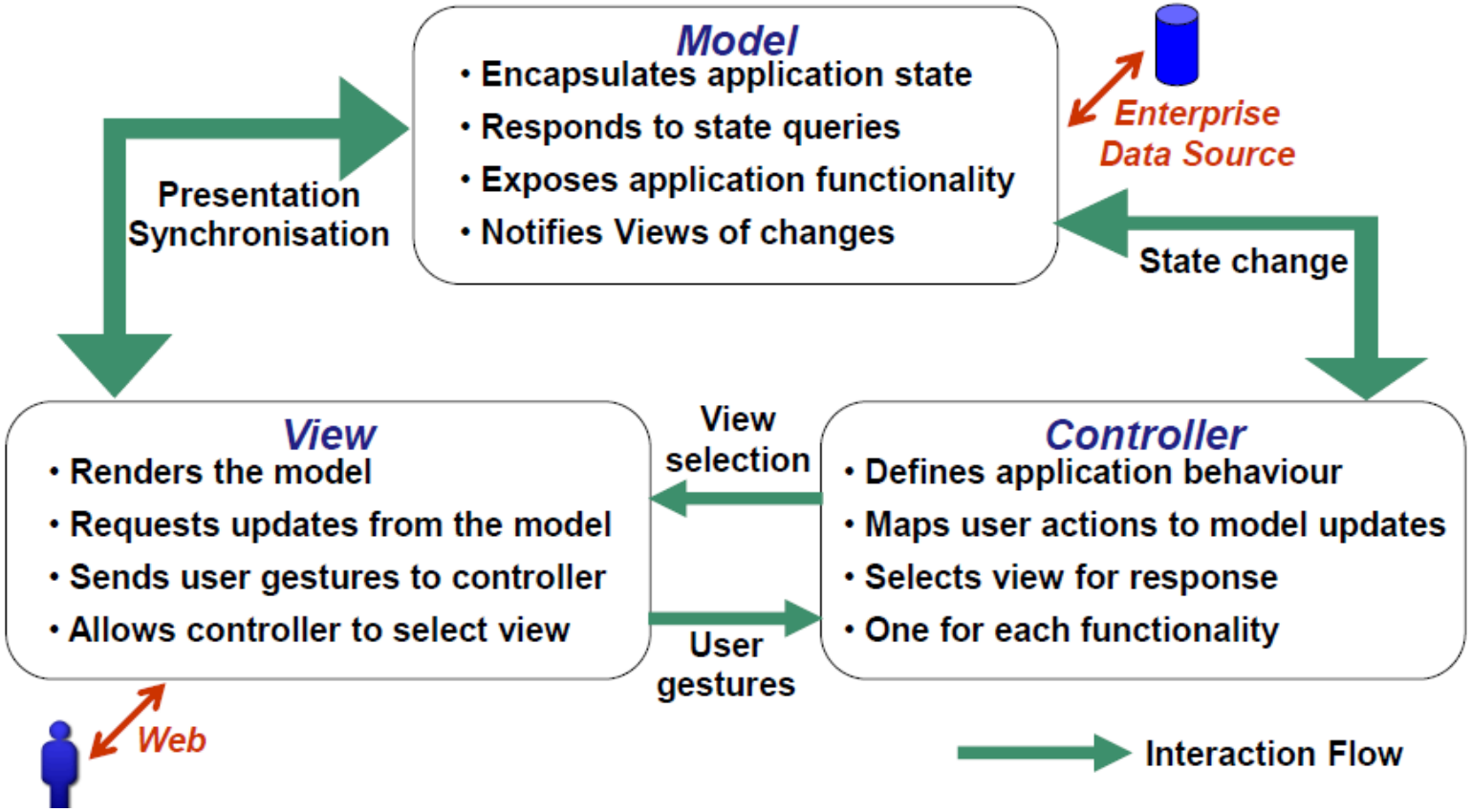
```java
@Autowired
private PersonService personService;

@RequestMapping(value = "/hibernateDaoServiceAdd", method = RequestMethod.GET)
public String hibernateDaoServiceAdd(Locale locale, Model model) {
    Person p = new Person();
    p.setAge(20);
    p.setFirst("FirstName");
    p.setLast("lastName");

    personService.registerPerson(p);

    return "home";
}
```

Architectural Design Patterns

- Can you describe some patterns?
  - E.g. Laying Pattern/Horizontal/Vertical/MVC
- What pattern have you used for your project? Explain what it is and how you have used it

- What are their advantages/disadvantages

MVC Pattern


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies Views of changes
Enterprise Data Source
Presentation Synchronisation
State change
View
• Renders the model
• Requests updates from the model
• Sends user gestures to controller
• Allows controller to select view
View selection
User gestures
Controller
• Defines application behaviour
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Web
Interaction Flow

Architectural Design Patterns

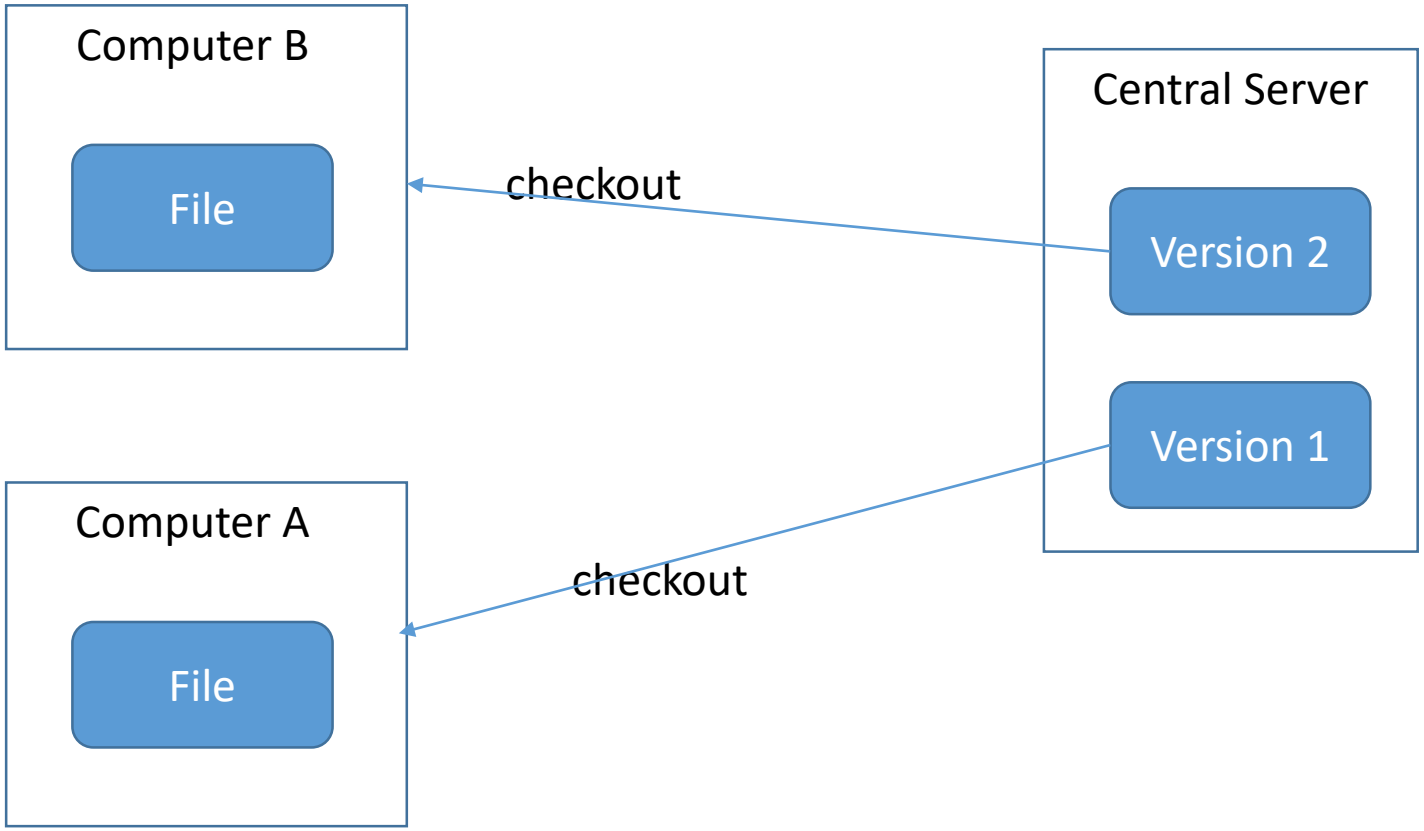
- What are sessions?
- What are types of session state?
  - Client Session state – Data on client e.g. URL for a web presentation/cookies/hidden field on web form/store on rich client
  - Server Session State – Data in server memory between requests
  - Database Session State – Data in tables and fields in database
- What have you used?
- What communication patterns have you used?

GIT

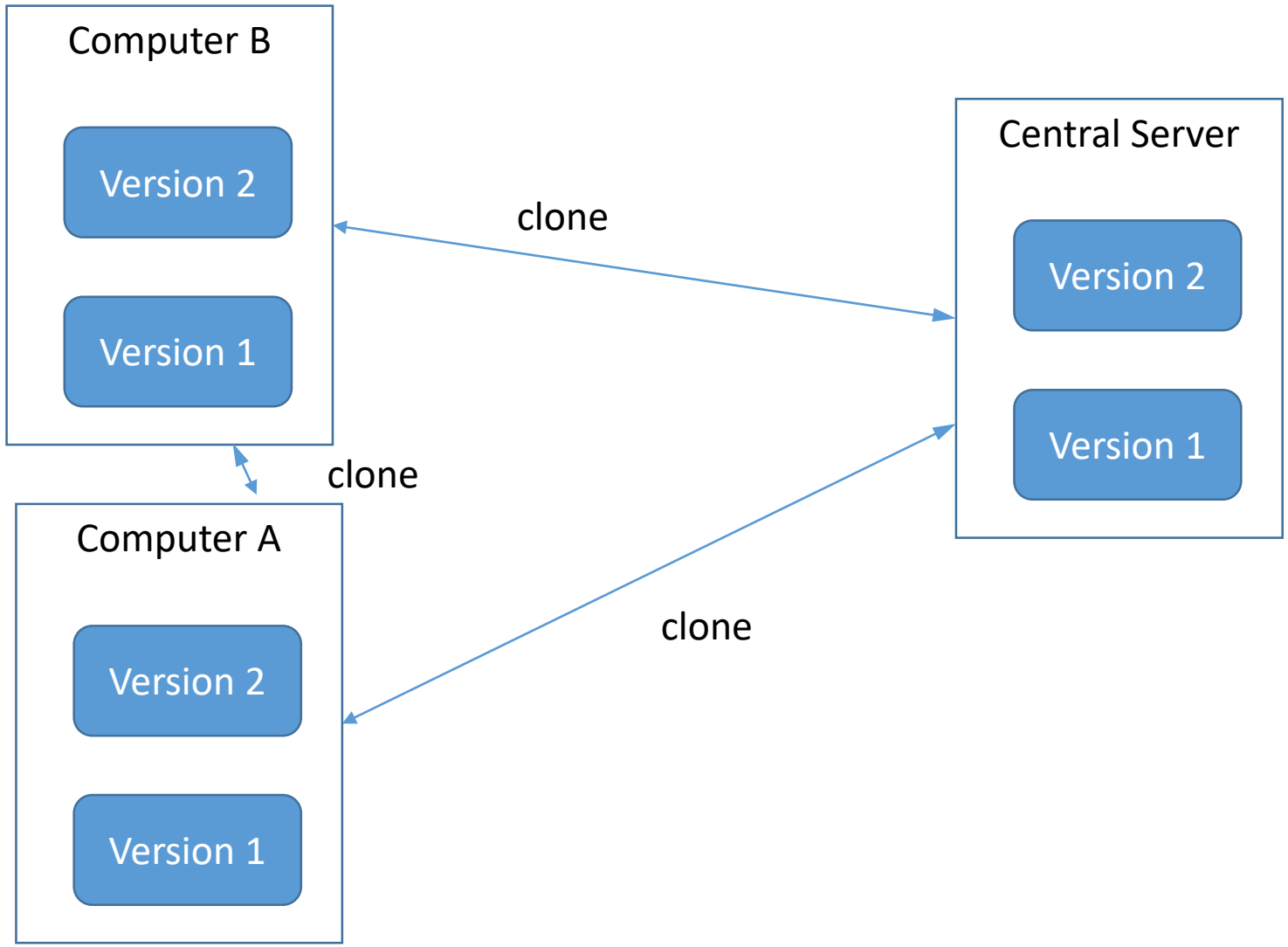
- What is GIT/ Why is GIT valuable?
- How does Git differ from other VCS?

- Not so much low-level details.

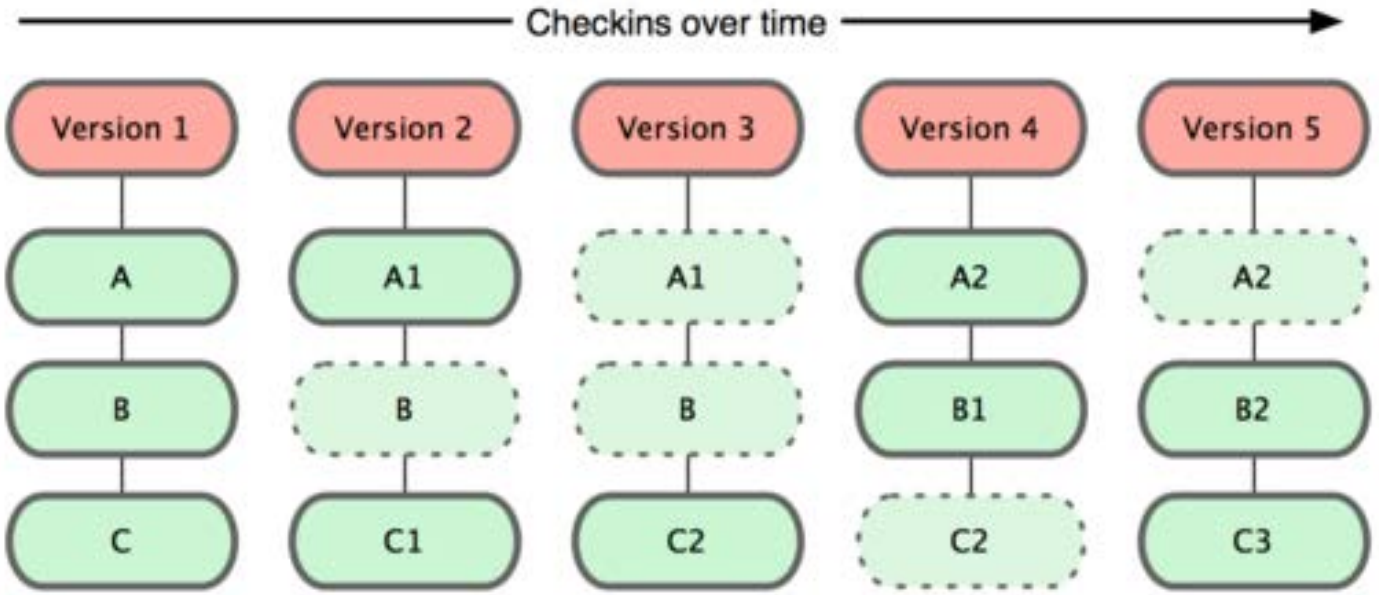
How traditional version control system works



How git works



How git works



- *Internally, all copies of similar files are not stored, instead ‘deltas’, or changes between the files
  - This is only to save on storage space
  - It is easiest to think of different copies of similar files

https://stackoverflow.com/questions/8198105/how-does-git-store-files

APIs

- What are APIs?
- What is REST?
- What are typical REST behaviours?
- Can you explain what OAuth is for/how it works

Using REST APIs

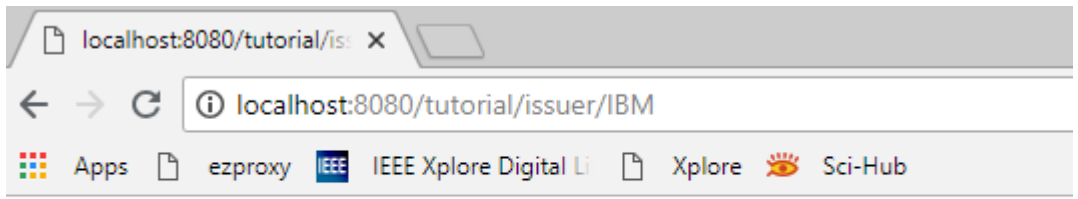
```java
public class Issuer {
    private String ticker;
    private String issuerName;
    private String issuerType;
    private String country;

    public Issuer() {
    }

    public Issuer(String ticker, String issuerName, String issuerType, String country) {
        setTicker(ticker);
        setIssuerName(issuerName);
        setIssuerType(issuerType);
        setCountry(country);
    }

    public String getTicker() {
        return ticker;
    }

    public void setTicker(String ticker) {
        this.ticker = ticker;
    }

    public String getIssuerName() {
        return issuerName;
    }

    public void setIssuerName(String issuerName) {
        this.issuerName = issuerName;
    }

    public String getIssuerType() {
        return issuerType;
    }

    public void setIssuerType(String issuerType) {
        this.issuerType = issuerType;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public String toString() {
        return "[" + getTicker()
                + ", " + getIssuerName()
                + ", " + getIssuerType()
                + ", " + getCountry()
                + "]";
    }
}
```

```java
@Controller
public class RestController {

    private static final Logger logger = LoggerFactory.getLogger(RestController.class);
    private Map<String, Issuer> issuers = new HashMap<String, Issuer>();

    public RestController() {
        // pre-initialize the list of issuers available ...

        issuers.put("ATEN", new Issuer("ATEN", "A10 Networks Inc", "corp", "USA"));
        issuers.put("AAPL", new Issuer("AAPL", "Apple Inc", "corp", "USA"));
        issuers.put("T", new Issuer("T", "AT&T", "corp", "USA"));
        issuers.put("CSCO", new Issuer("CSCO", "Cisco Systems, Inc.", "corp", "USA"));
        issuers.put("CTXS", new Issuer("CTXS", "Citrix Systems, Inc.", "corp", "USA"));
        issuers.put("GOOGL", new Issuer("GOOGL", "Google Inc", "corp", "USA"));
        issuers.put("IBM", new Issuer("IBM", "IBM", "corp", "USA"));
        issuers.put("JNPR", new Issuer("JNPR", "Juniper Networks, Inc.", "corp", "USA"));
        issuers.put("MSFT", new Issuer("MSFT", "Microsoft Corporation", "corp", "USA"));
        issuers.put("ORCL", new Issuer("ORCL", "Oracle Corporation", "corp", "USA"));
    }
```

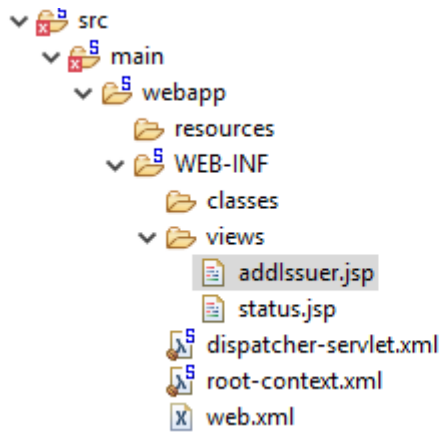
```java
@RequestMapping(value = "/", method = RequestMethod.GET)
public String home(Locale locale, Model model) {
    logger.info("Welcome home! The client locale is asdsad {}.", locale);

    Date date = new Date();
    DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);

    String formattedDate = dateFormat.format(date);

    model.addAttribute("serverTime", formattedDate );

    return "status";
}

@RequestMapping(value="/issuers", method=RequestMethod.GET)
@ResponseBody
public Map<String, Issuer> getAllIssuers() {
    logger.info("Inside getAllIssuers() method...");
    return issuers;
}
```

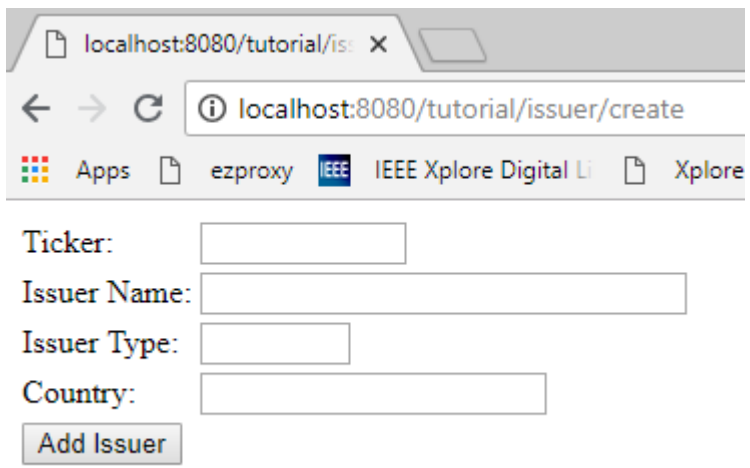
```java
@RequestMapping(value="/issuer/{ticker}", method=RequestMethod.GET)
@ResponseBody
public Issuer getIssuerByTicker(@PathVariable("ticker") String ticker) {
    Issuer myIssuer = issuers.get(ticker);

    if (myIssuer != null) {
        logger.info("Inside getIssuerByTicker, returned: " + myIssuer.toString());
    } else {
        logger.info("Inside getIssuerByTicker, ticker: " + ticker + ", NOT FOUND!");
    }
    return myIssuer;
}

@RequestMapping(value="/issuer/delete/{ticker}", method=RequestMethod.GET)
@ResponseBody
public Issuer deleteIssuerByTicker(@PathVariable("ticker") String ticker) {
    Issuer myIssuer = issuers.remove(ticker);

    if (myIssuer != null) {
        logger.info("Inside deleteIssuerByTicker, deleted: " + myIssuer.toString());
    } else {
        logger.info("Inside deleteIssuerByTicker, ticker: " + ticker + ", NOT FOUND!");
    }
    return myIssuer;
}

@RequestMapping(value="/issuer/create", method=RequestMethod.GET)
public ModelAndView addIssuer() {

    return new ModelAndView("addIssuer", "command", new Issuer());
}

@RequestMapping(value="/issuer/addIssuer", method=RequestMethod.POST)
@ResponseBody
public Issuer addIssuer(@ModelAttribute("issuer") Issuer issuer) {

    if (issuer != null) {
        logger.info("Inside addIssuer, adding: " + issuer.toString());
    } else {
        logger.info("Inside addIssuer...");
    }
    issuers.put(issuer.getTicker(), issuer);
    return issuer;
```


```
{"MSFT":{"ticker":"MSFT","issuerName":"Microsoft Corporation","issuerType":"corp","country":"USA"},"GOOGL":
{"ticker":"GOOGL","issuerName":"Google Inc","issuerType":"corp","country":"USA"},"ATEN":{"ticker":"ATEN","issuerName":"A10 Networks
Inc","issuerType":"corp","country":"USA"},"AAPL":{"ticker":"AAPL","issuerName":"Apple Inc","issuerType":"corp","country":"USA"},"CSCO":
{"ticker":"CSCO","issuerName":"Cisco Systems, Inc.","issuerType":"corp","country":"USA"},"CTXS":{"ticker":"CTXS","issuerName":"Citrix
Systems, Inc.","issuerType":"corp","country":"USA"},"T":{"ticker":"T","issuerName":"AT&T","issuerType":"corp","country":"USA"},"JNPR":
{"ticker":"JNPR","issuerName":"Juniper Networks, Inc.","issuerType":"corp","country":"USA"},"IBM":
{"ticker":"IBM","issuerName":"IBM","issuerType":"corp","country":"USA"},"ORCL":{"ticker":"ORCL","issuerName":"Oracle
Corporation","issuerType":"corp","country":"USA"}}
```


```
{"ticker":"IBM","issuerName":"IBM","issuerType":"corp","country":"USA"}
```






```jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="f" %>
<%@ page session="false" %>
<html>
    <head>
        <title>${message}</title>
    </head>
    <body>
        <h1>${message}</h1>
        <f:form method="POST" action="addIssuer">
        <table>
            <tbody>
                <tr>
                  <td>Ticker:</td>
                  <td><f:input path="ticker" size="10" maxlength="10"></f:input></td>
                </tr>
                <tr>
                  <td>Issuer Name:</td>
                  <td><f:input path="issuerName" size="30"></f:input></td>
                </tr>
                <tr>
                  <td>Issuer Type:</td>
                  <td><f:input path="issuerType" size="6"></f:input></td>
                </tr>
                <tr>
                  <td>Country:</td>
                  <td><f:input path="country" size="20"></f:input></td>
                </tr>
                <tr>
                    <td colspan="2"><input type="submit" value="Add Issuer"></td>
                </tr>
            </tbody>
        </table>
        </f:form>
    </body>
</html>
```

Access external/internal REST APIs

```
JavaHttpUrlConnectionReader test = new JavaHttpUrlConnectionReader();

public class JavaHttpUrlConnectionReader
{
  public static void main(String[] args)
  throws Exception
  {
    new JavaHttpUrlConnectionReader();
  }

  public JavaHttpUrlConnectionReader()
  {
    try
    {
      String myUrl = "https://api.github.com/users/mojombo";
      // if your url can contain weird characters you will want to
      // encode it here, something like this:
      // myUrl = URLEncoder.encode(myUrl, "UTF-8");

      String results = doHttpUrlConnectionAction(myUrl);
      System.out.println(results);
    }
    catch (Exception e)
    {
      // deal with the exception in your "controller"
    }
  }
```

```
private String doHttpUrlConnectionAction(String desiredUrl)
throws Exception
{
  URL url = null;
  BufferedReader reader = null;
  StringBuilder stringBuilder;

  try
  {
    // create the HttpURLConnection
    url = new URL(desiredUrl);
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();

    // just want to do an HTTP GET here
    connection.setRequestMethod("GET");

    // uncomment this if you want to write output to this url
    //connection.setDoOutput(true);

    // give it 15 seconds to respond
    connection.setReadTimeout(15*1000);
    connection.connect();

    // read the output from the server
    reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
    stringBuilder = new StringBuilder();

    String line = null;
    while ((line = reader.readLine()) != null)
    {
      stringBuilder.append(line + "\n");
    }
    return stringBuilder.toString();
  }
  catch (Exception e)
  {
    e.printStackTrace();
    throw e;
  }
  finally
  {
    // close the reader; this can throw an exception too, so
    // wrap it in another try/catch block.
    if (reader != null)
    {
      try
      {
        reader.close();
      }
      catch (IOException ioe)
      {
```

```java
System.out.println("Testing getUser API----------");
RestTemplate restTemplate = new RestTemplate();
Issuer isuser = restTemplate.getForObject(REST_SERVICE_URI+"/issuer/IBM", Issuer.class);
System.out.println(isuser);

RestTemplate restTemplate2 = new RestTemplate();
String response = restTemplate2.getForObject("https://api.github.com/users/defunkt", String.class);
System.out.println(response);
```

```
[IBM, IBM, corp, USA]
{"login":"defunkt","id":2,"node_id":"MDQ6VXNlcjI=","avatar_url":"https://avatars0.githubusercontent.com/u/2?v=4","gravatar_id":"","url":"https://api.g
```