

সূচিপত্র

পরিচিতি	1.1
পূর্বশর্ত	1.1.1
ওয়েবসাইট যেভাবে কাজ করে	1.1.2
ওয়েব ফ্রেমওয়ার্ক পরিচিতি	1.1.3
জ্যাক্সো পরিচিতি	1.1.4
এমটিভি প্যাটার্ন পরিচিতি	1.1.5
ভার্চুয়াল এনভায়নমেন্ট ইন্সটল	1.1.6
জ্যাংগো ইন্সটল	1.1.7
প্রথম প্রজেক্ট শুরু	1.1.8
ডেভেলপমেন্ট সার্ভার রান করা	1.1.9
অ্যাপস কি? প্রোজেক্ট এবং অ্যাপস এর পার্থক্য!	1.1.10
প্রথম অ্যাপস তৈরি	1.1.11
সেটিংস পরিচিতি	1.1.12
সিম্পল ভিউ ফাংশন	1.1.13
ইউআরএল কনফিগারেশন	1.1.14
টেমপ্লেট ব্যবহার	1.1.15
ও আর এম পরিচিতি	1.1.16
ডাটাবেইজ কনফিগারেশন	1.1.17
মডেল তৈরি	1.1.18
জ্যাক্সো শেল পরিচিতি	1.1.19
ডাটাবেইজে ডাটা ইনসার্ট, আপডেট, রিড, ডিলেট	1.1.20
ভিউ থেকে ডাটা রিড করা	1.1.21
টেমপ্লেট ট্যাগ এবং ফিল্টার	1.1.22
ইউআরএল কনফিগারেশন ইন ডেপথ	1.1.23
ডায়নামিক কনটেন্ট	1.1.24
এডমিন পরিচিতি	1.1.25
প্রথম প্রজেক্টঃ সিম্পল ইউআরএল শর্টনার	1.1.26
শীঘ্রই আসছে	1.1.26.1
দ্বিতীয় প্রজেক্টঃ সিম্পল ব্লগ সাইট	1.1.27
শীঘ্রই আসছে	1.1.27.1

তৃতীয় প্রজেক্টঃ স্কুলের বেজান্ট ম্যানেজমেন্ট সিস্টেম	1.1.28
শীঘ্রই আসছে	1.1.28.1
স্বতন্ত্র টিউটোরিয়াল ১	2.1
শুরুর আগে	2.1.1
ইনস্টলেশন	2.1.2
প্রথম অধ্যায়	2.1.3
দ্বিতীয় অধ্যায়	2.1.4
তৃতীয় অধ্যায়	2.1.5
চতুর্থ অধ্যায়	2.1.6
স্বতন্ত্র টিউটোরিয়াল ২	3.1
হ্যালো ওয়ার্ল্ড	3.1.1

বাংলায় জ্যাঙ্গো টিউটোরিয়াল



Like



Share

11K people like this. Sign Up to see what your friends like.

স্বয়ংক্রিয় কন্ট্রিবিউটরের তালিকা

(প্রথম ৫ জন)

[004] [Nuhil Mehdy](#)

[001] [Abu Ashraf Masnun](#)

[001] [Md. Al-Amin](#)

প্রারম্ভিকা

সহজ এবং সংক্ষেপে বলতে গেলে, django হচ্ছে পাইথনে লেখা একটি ফ্রি এবং ওপেনসোর্স ওয়েব অ্যাপ্লিকেশন ফ্রেমওয়ার্ক যেটা 'মডেল - ডিউ - কন্ট্রোলার' আর্কিটেকচারে প্যাটার্ন ফলো করে। অন্যান্য ফ্রেমওয়ার্ক এর মতই জ্যাঙ্গো দিয়ে খুব দ্রুত এবং তুলনামূলক কম কোড লিখে ভালো মানের ওয়েব অ্যাপ ডেভেলপ করা যায়।

ওপেন সোর্স

এই বইটি মূলত স্বেচ্ছাশ্রমে লেখা এবং বইটি সম্পূর্ণ ওপেন সোর্স। এখানে তাই আপনিও অবদান রাখতে পারেন লেখক হিসেবে। আপনার কন্ট্রিবিউশান গৃহীত হলে অবদানকারীদের তালিকায় আপনার নাম যোগ করে দেওয়া হবে।

এটি মূলত একটি [গিটহাব রিপোজিটরি](#) যেখানে এই বইয়ের আর্টিকেল গুলো মার্কডাউন ফরম্যাটে লেখা হচ্ছে। রিপোজিটরিটি ফর্ক করে পুল রিকুয়েস্ট পাঠানোর মাধ্যমে আপনারাও অবদান রাখতে পারেন।

আপনি যদি শুধুই পাঠক হন অথবা গিট সম্পর্কে ভালো আইডিয়া না থাকে, সেক্ষেত্রে পরিবর্তন বা পরিবর্ধন এর জন্য ইমেইল করুন - [masnun\[at\]transcendio.net](mailto:masnun[at]transcendio.net) - এই ঠিকানায়। মূল লেখক আপনার সাজেশন বিবেচনা করে প্রয়োজনীয় পরিবর্তন করে দিবেন।



Like 21



Share

gitter

join chat



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](#).

এই কোর্সটি করতে আপনার যা (করা/জানা/থাকা) লাগবে!

- পাইথন প্রোগ্রামিং ল্যান্ডস্কেপ জানা থাকা লাগবে! একদম ভালো না হলেও মোটামুটি লেভেলের পাইথন জ্ঞান দরকার, ওওপি সম্পর্কে ভালো ধারণা থাকতে হবে।
- এইচটিএমএল জানা থাকা লাগবে, সিএসএস মোটামুটি জানতে হবে, জাভাস্ক্রিপ্ট সম্পর্কে ধারণা থাকলে ভালো!
- কম্পিউটার থাকতে হবে, ইন্টারনেট থাকলে খুবই ভালো! পাইথন ও ইন্সটল করা থাকতে হবে। pip ইন্সটল করা থাকতে হবে, pip দিয়ে প্যাকেজ ইন্সটল করা জানতে হবে। উইন্ডোজের কমান্ড প্রম্পট বা লিনাক্সের টার্মিনাল এর বেসিক কমান্ডগুলো জানতে হবে, যেমন cd, ls, source ইত্যাদি। (এগুলো পাইথন সম্পর্কিত বিষয়, তাই এই কোর্সে এগুলো কভার করা হবে না।)
- ধৈর্য, ইচ্ছা এবং মনোযোগ!
- পাইথনের অন্য কোন ওয়েব ফ্রেমওয়ার্ক (যেমন ফ্ল্যাস্ক) জানা থাকলে খুবই ভালো! অন্যথায় প্রথম প্রথম জ্যাকো কঠিন লাগতে পারে!
- অন্য কোন ল্যান্ডস্কেপের ওয়েব ফ্রেমওয়ার্ক (যেমন লারাভেল, রেইলস) জানা থাকলে ভালো!
- গুগলে সার্চ করা জানতে হবে। খুব ভালো না, মোটামুটি ভাবে সার্চ করে কোন বিষয় জানতে পারার মত হলেও চলবে!
- প্রতিটি চ্যাপ্টারে কোড নিজে করতে হবে (নিজে মানে নিজ হাতে “কি বোর্ডে” টাইপ করতে হবে, কপি পেস্ট করা বা শুধু কোডের দিকে চোখ বুলিয়ে যাওয়া একদম নিষেধ)
- প্রতিটি চ্যাপ্টার বুঝতে হবে, একবার পড়ে না বুঝলে দুই/তিন বার পড়তে হবে। তাও না বুঝলে চ্যাপ্টারের শেষে দেয়া বিস্তারিত জানার লিংকগুলোতে যেয়ে পড়তে হবে। তাও না বুঝলে গুগলে সে বিষয় সার্চ করে পড়তে হবে, তাও না বুঝলে পাইথন সম্পর্কিত কোন গ্রুপে না বুঝা বিষয়টা নির্দিষ্ট করে উল্লেখ করে প্রশ্ন করতে হবে। (আর হ্যা! ইংরেজি কিছুটা জানা থাকলেও স্ট্যাকওভারফ্লো তে সে বিষয়টা সার্চ করে বের করার চেষ্টা করতে হবে!)
- ডাটাবেইজ সম্পর্কে ধারণা থাকতে হবে

আমরা কোর্সেটতে পাইথন ৩ এর সাথে জ্যাকো ১.১১ ব্যবহার করব। প্রোজেক্ট এর কোড গুলো এবং স্ক্রিনশটগুলো উইন্ডোজ ওএস তে করব। কারণ আমরা ধরে নিচ্ছি যে এই কোর্স যে করবে সে জ্যাকোতে একেবারে নতুন (কিন্তু অবশ্যই পাইথন সম্পর্কে ভালো ধারণা রাখে!) এবং সে উইন্ডোজ ব্যবহারকারি। (কেননা বাংলাদেশের অধিকাংশ বিগিনার প্রোগ্রামার উইন্ডোজ ওএস ব্যবহার করে!) তবে অন্য অপারেটিং সিস্টেম গুলোর সাথে এক্ষেত্রে খুব একটা পার্থক্য থাকবে না বিধায় লিনাক্স বা ম্যাক ব্যবহারকারীদের কোন সমস্যা হবার কথা না!

ওয়েবসাইট যেভাবে কাজ করে!

সহজ ভাষায় বলতে গেলে, যখন আমরা আমাদের মোবাইল বা কম্পিউটারে কোন ওয়েবসাইটের ঠিকানা লিখে এন্টার চাপি তখন ব্রাউজারটি সেই ওয়েবসাইটে সার্ভারে একটা রিকুয়েস্ট পাঠা, রিকুয়েস্ট পেয়ে সেই সার্ভার প্রথমে রিকুয়েস্টটিকে পর্যবেক্ষন করে, অতঃপর সেই রিকুয়েস্ট এর প্রেক্ষিতে একতা উত্তর বা রেসপন্স আমাদের ব্রাউজারে পাঠিয়ে দেয় এবং আমাদের ব্রাউজার সে রেসপন্সটি আমাদের কাছে প্রদর্শন করে।

যদি আমরা <http://www.django.howtocode.com.bd> আমাদের ব্রাউজারের এড্রেসবারে লিখে এন্টার চাপি তাহলে যে হবেঃ

1. আমাদের ব্রাউজার থেকে হাউটুকোড এর সার্ভারে একটা রিকুয়েস্ট যাবে, যেখানে বলা হবে যে আমি www.django.howtocode.com.bd ঠিকানাতে থাকা কনটেন্ট গুলো দেখতে চাচ্ছি!
2. হাউটুকোড এর সার্ভার উক্ত ঠিকানা পর্যবেক্ষন করে আমাদেরকে জ্যাঙ্গো কোর্সের মূল পাতা টি শো করার সিদ্ধান্ত নিবে!
3. অতঃপর সার্ভার তার ডাটাবেইজ থেকে জ্যাঙ্গো কোর্সের মূল পাতার কনটেন্টগুলো নিয়ে আসবে এবং
4. সেটাকে এইচটিএমএল ফরমেটে সাজিয়ে আমাদের কাছে পাঠিয়ে দিবে।
5. আমাদের ব্রাউজার সেই এইচটিএমএল রেসপন্সটিকে সঠিক ভাবে সাজিয়ে আমাদের সামনে উপস্থিত করবে। যখনই আমরা কোন ওয়েবসাইটের লিংকে ক্লিক করি তখনই উপরের কাজগুলো হয়ে থাকে। এটা একটা রিকুয়েস্ট-রেসপন্স সাইকেল! ব্রাউজার থেকে রিকুয়েস্ট যাবে, সার্ভার থেকে রেসপন্স আসবে!

ওয়েব ফ্রেমওয়ার্ক পরিচিতি

ডেভেলপার হিসেবে আপনি আপনার সারাজীবনে একটিমাত্র ওয়েবসাইট/ওয়েবঅ্যাপ তৈরি করে অবসর নিবেন বলে মনে হয়না! অবশ্যই অনেক অনেক ওয়েবসাইট/ওয়েবঅ্যাপ তৈরি করবেন! সমস্যা হল, প্রতিটি ওয়েবসাইট বা ওয়েবঅ্যাপ এর মধ্যে কিছু জিনিস সব সময়ই কমন থাকে, এবং প্রতিটি ওয়েবসাইট তৈরি করার সময় বার বার কিছু বিষয়ে একই কোড লিখা লাগে! যেমন রেজিস্ট্রেশন/লগিন সিস্টেম তৈরি, ডাটাবেইজ এর সাথে সংযোগ, কমন সিকিউরিটি ফিচার, এডমিন প্যানেল তৈরি করা ইত্যাদি। প্রতিবারই এগুলো করা একেতো খুবই বিরক্তিকর কাজ তার উপর একজন সফল ‘অলস’ ডেভেলপারের কর্মপন্থার সাথে সাংঘর্ষিক!

ওয়েবফ্রেমওয়ার্ক হল এমন সফওয়ার বা অ্যাপ্লিকেশন যেটাতে এ সকল কমন বিষয়েগুলো বিস্টাইন থাকে, তাই ফ্রেমওয়ার্ক ব্যবহার করলে একই কোড বার বার লিখতে হয়না, সময় এবং মাথার ঘাম দুটোই বেঁচে যায়! আমাদের কাজ শুধু প্রয়োজনীয় ফ্রেমওয়ার্কটা নিজেদের প্রোজেক্টে ব্যবহার করা।

জ্যাস্পো পরিচিতি

জ্যাস্পো খুবই শক্তিশালি, রোবাস্ট, প্রোডাক্টিভ একটি ফ্রী এবং ওপেনসোর্স ওয়েব ফ্রেমওয়ার্ক। জ্যাস্পো দিয়ে খুব সহজে, দ্রুত বড় ধরনের ওয়েবসাইট/ওয়েব অ্যাপ্লিকেশন তৈরি করা যায়, এটি পাইথন প্রোগ্রামিং ল্যান্গুয়েজ ব্যবহার করে তৈরি করা হয়েছে, তাই জ্যাস্পো শেখার আগে অবশ্যই আপনাকে পাইথন শিখতে হবে!

কয়েকজন ডেভেলপার, যারা নিউজ সাইট তৈরি করত। তারা খেয়াল করে দেখল যে তাদের তৈরি করা প্রতিটি নিউজ সাইটেরই প্রায় ৯০ ভাগ কোড একই রকম, তারা আসলে প্রতিটা ওয়েবসাইটে একই কোড বার বার লিখছে! তখন তারা সকল কমন কোডগুলো, যা সবগুলো সাইটেই দরকার হয়, সেগুলো একসাথে করে একটা ফ্রেমওয়ার্ক তৈরি করল, জ্যাস্পো! এখন একই কোড বার বার লিখা লাগেনা, জ্যাস্পো ব্যবহার করেই কমন বিষয়গুলো কোড করা থেকে মুক্তি পাওয়া যায়! এটা জ্যাস্পোর শুরুর গল্প... জ্যাস্পোর ইতিহাস সম্পর্কে আরো জানতে পারবেন এখানেঃ

[https://bn.wikipedia.org/wiki/%E0%A6%9C%E0%A7%8D%E0%A6%AF%E0%A6%BE%E0%A6%99%E0%A7%8D%E0%A6%97%E0%A7%8B_\(%E0%A6%93%E0%A6%AF%E0%A6%BC%E0%A7%87%E0%A6%AC_%E0%A6%AB%E0%A7%8D%E0%A6%B0%E0%A7%87%E0%A6%AE%E0%A6%93%E0%A6%AF%E0%A6%BC%E0%A6%BE%E0%A6%B0%E0%A7%8D%E0%A6%95\)](https://bn.wikipedia.org/wiki/%E0%A6%9C%E0%A7%8D%E0%A6%AF%E0%A6%BE%E0%A6%99%E0%A7%8D%E0%A6%97%E0%A7%8B_(%E0%A6%93%E0%A6%AF%E0%A6%BC%E0%A7%87%E0%A6%AC_%E0%A6%AB%E0%A7%8D%E0%A6%B0%E0%A7%87%E0%A6%AE%E0%A6%93%E0%A6%AF%E0%A6%BC%E0%A6%BE%E0%A6%B0%E0%A7%8D%E0%A6%95))

জ্যাস্পো দিয়ে তৈরি এরকম কিছু পরিচিত ওয়েবসাইট হলঃ <http://instagram.com/>

<https://www.pinterest.com/> <https://www.nasa.gov/>

<https://www.spotify.com/> <https://support.mozilla.org/> <https://disqus.com/>

<http://www.theguardian.com/>

<http://www.nationalgeographic.com/>

<https://bitbucket.org/>

উপরের লিস্ট দেখে নিশ্চয় এটা বুঝতে পারছেন, আপনি যত বড় আর যত কমপ্লেক্স ওয়েব অ্যাপ্লিকেশন তৈরি করতে চান না কেন... জ্যাস্পো অলটাইম পারফেক্ট চয়েস!

এমটিভি প্যাটার্ন পরিচিতি

এম,টি,ভি (MTV) এর পূর্ণরূপ হল মডেল, টেমপ্লেট, ভিউ (Model, Template, View)। এটা একটা ডেভেলপমেন্ট ডিজাইন প্যাটার্ন। প্রোজেক্ট যখন আস্তে আস্তে বড় হয়ে যায়, অনেক অনেক কোড লেখা হয়, তখন সেগুলো মেইনটেন করতে ডেভেলপারের হিমশিম খেতে হয়। এছাড়াও একটা প্রোজেক্টে অনেক ধরনের টিম কাজ করে, ফ্রন্টএন্ড, ব্যাকএন্ড, ডাটাবেইজ ইত্যাদি ক্ষেত্রে আলাদা আলাদা টিম কাজ করে, তো একটা প্রোজেক্টে এত টিম বা ডেভেলপার কাজ করলে তাদের মধ্যে বিভিন্ন কনফিউশন তৈরি হয়! কার কোড কেমন হবে, একজনের কোডের কারনে অন্যের কোডে কোন সমস্যা হবে কিনা ইত্যাদি বিষয় নিয়ে সমস্যা তৈরি হয়।

এধরনের সমস্যার সমাধান হল এই ডেভেলপমেন্ট ডিজাইন প্যাটার্ন! এটা প্রোজেক্টকে সুন্দর ভাবে অর্গানাইজ করে রাখে, তাই প্রোজেক্ট বড় হলেও সেটা মেইনটেন করতে সমস্যা হয়না। এবং কোডগুলোও বিভিন্ন ভাগে বিভক্ত থাকে, তাই একেক টিম/ডেভেলপার কোডের একেকটা পার্ট নিয়ে নিশ্চিত কাজ করতে পারে, অন্যের কোডের কোন ধরনের সমস্যা করা ছাড়াই! একারণেই প্রায় সকল ধরনের সফটওয়্যার বা ওয়েবএপ ডিজাইন প্যাটার্ন ফলো করেই তৈরি করা হয়!

বিভিন্ন ধরনের ডেভেলপমেন্ট ডিজাইন প্যাটার্ন রয়েছে, জ্যাক্সো ওয়েবফ্রেমওয়ার্ক যে ডিজাইন ফলো করে সেটার নাম হল এমটিভি প্যাটার্ন, অর্থাৎ মডেল, টেমপ্লেট, ভিউ প্যাটার্ন! নাম শুনেই বোঝা যাচ্ছে, এই প্যাটার্নের মূল পার্ট তিনটিঃ

১) মডেল (Model), আপনার প্রোজেক্ট অবশ্যই ডাটাবেইজ ব্যবহার করবে এবং আপনার সাইটের সব কনটেন্ট ডাটাবেইজে সেভ থাকবে। ডাটাবেইজের ডিজাইন, ডাটাবেইজের সঙ্গে আপনার সকল ধরনের যোগাযোগ, ডাটা বেইজের সকল অপারেশন গুলো মডেল হিসেবে লেখা হয়! অর্থাৎ আপনার সরাসরি ডাটাবেইজ ব্যবহার করার দরকার হবেনা, বরং মডেল ব্যবহার করেই ডাটাবেইজের কাজগুলো করতে হবে। মডেল ডাটাবেইজের উপর একটা লেয়ার হিসেবে থাকে, আপনি একটা ডাটা মডেল তৈরি করে সেটা যেকোন ডাটাবেইজের জন্য ব্যবহার করতে পারবেন। মডেল এর এই ব্যবহার খুবই সহজ করে দেয় জ্যাক্সোর ওআরএম! এ নিয়ে আমরা পরে আরো বিস্তারিত জানব।

২) টেমপ্লেট (Template), ফ্রন্টএন্ড এর কোডগুলো, যা ইউজার বা ভিউয়ার এর সামনে প্রদর্শিত হবে সে সকল কোড এই ট্যামপ্লেট পার্টে থাকবে। সাধারণত এইচটিএমএল, সিএসএস বা অন্যান্য স্ট্যাটিক ফাইলগুলোই ট্যামপ্লেট এর মাধ্যমে প্রদর্শিত হয়।

৩) ভিউ (View), অত্যন্ত গুরুত্বপূর্ণ পার্ট, ক্লায়েন্ট বা ইউজার কোন ইউআরএল এ রিকুয়েস্ট করলে তাকে কোন জিনিস দেখানো হবে বা দেখানো হবেনা সেটা ঠিক করা ভিউ এর কাজ। ভিউ ক্লায়েন্টের রিকুয়েস্ট পর্যবেক্ষন করে মডেল থেকে প্রয়োজন অনুযায়ী ডাটা নিয়ে সেটা ট্যামপ্লেট এর মাধ্যমে সাজিয়ে ইউজারের কাছে পাঠিয়ে দেয়া। অর্থাৎ ক্লায়েন্ট, মডেল ও টেমপ্লেট এর মধ্যকার যোগাযোগ এবং ক্লায়েন্টকে তার আকাঙ্ক্ষিত বস্তু ঠিকমত প্রদর্শন করার ব্যবস্থা করা হল ভিউ এর দায়িত্ব!

ভিউ ঠিক করে দেয় ব্যবহারকারী কোন জিনিসটা দেখবে! আর টেমপ্লেট ঠিক করে দেয় ভিউ এর ঠিক করা জিনিসটা ইউজারের সামনে কিভাবে (কোন ডিজাইনে) প্রদর্শিত হবে!

আস্তে আস্তে এমটিভি প্যাটার্ন বিষয়টা আরো ক্লিয়ার হয়ে যাবে, এখনই খুব বেশি মাথা ঘামানোর দরকার নেই।

ভার্চুয়াল এনভায়রনমেন্ট ইন্সটল

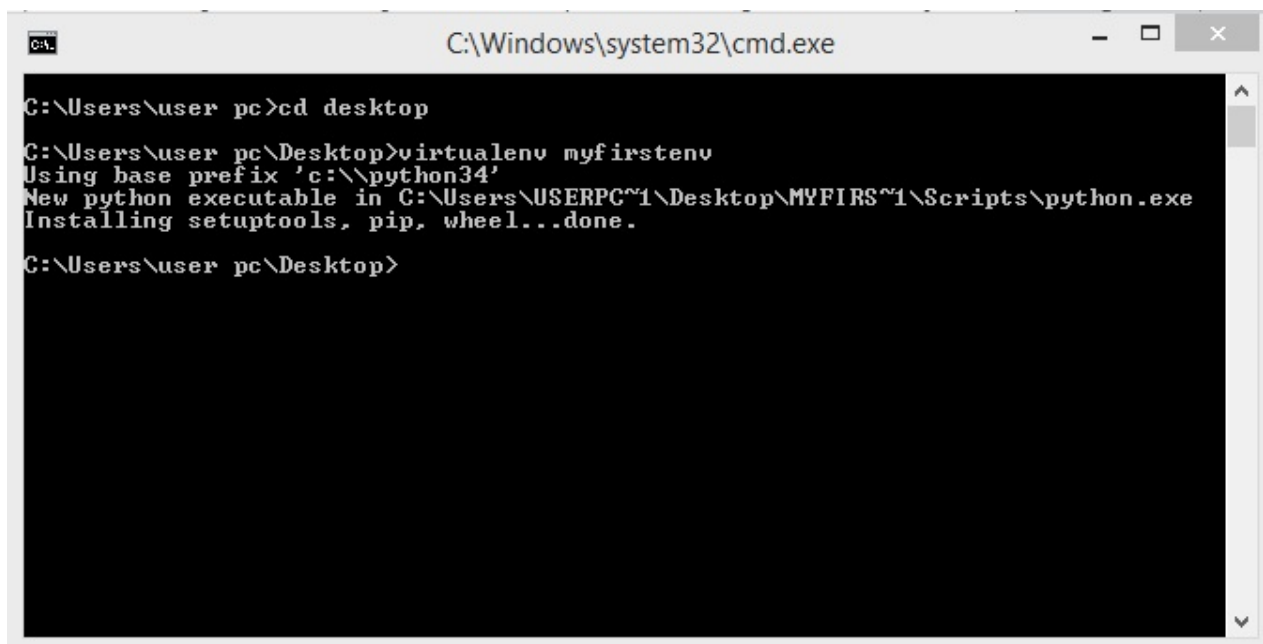
পাইথন প্রোগ্রামার হিসেবে ভার্চুয়াল এনভায়রনমেন্ট এর সাথে পরিচিত আছেন আশা করি। তাও সংক্ষেপে বলি, ভার্চুয়াল এনভায়রনমেন্ট হল একটা পাইথন ইন্টারপ্রেটার ইন্সটল! কঠিন হয়ে গেল!?! ধরি আপনার কম্পিউটারে পাইথন ৩ ইন্টারপ্রেটার ইন্সটল করা আছে, এবং সেখানে জ্যাকসো ১.১১ ভার্সন ইন্সটল করা আছে। এখন আপনি এমন একটা প্রোজেক্ট করতে চান যেটাতে জ্যাকসো ১.৮ ব্যবহার করা লাগবে তাহলে কি করবেন!?! আগের জ্যাকসো ১.১১ রিমোভ করে জ্যাকসো ১.৮ ইন্সটল করবেন!?! আবার যদি অন্য কোন প্রজেক্টে জ্যাকসোর লেটেস্ট ভার্সন ২.০ দরকার হয় তখন কি করবেন?!

বিভিন্ন ধরনের প্রজেক্টে বিভিন্ন ভার্সনের প্যাকেজ/মডিউল ব্যবহার করতে হয়, কিন্তু একটা মাত্র পাইথন ইন্টারপ্রেটারে একই প্যাকেজের একাধিক ভার্সন ইন্সটল করা যাবেনা! তাই সবচাইতে ভালো হবে যদি আপনি পাইথনের ইন্টারপ্রেটারটাই কপি করে ফেলেন! প্রতিটি প্রজেক্টের জন্য আলাদা ভাবে একটা পাইথন ইন্টারপ্রেটার থাকবে, সেখানে শুধুমাত্র সেই প্রজেক্টের জন্য দরকারি মডিউল/প্যাকেজগুলোই ইন্সটল করা থাকবে! এতে করে প্যাকেজের ভার্সন নিয়ে কোন সমস্যা হবেনা, এবং প্রোজেক্ট লাইভ সার্ভারে ডেপ্লয় করতে সুবিধা হবে।

ভার্চুয়াল এনভায়রনমেন্ট এই কাজটিই করে, আপনার কম্পিউটারের মেন পাইথন ইন্টারপ্রেটার এর একটা কপি করে দেয়। আপনি সেটা নিজের মত করে ব্যবহার করতে পারেন। আবার দরকার শেষ হয়ে গেলে সেটা ডিলেট করে দিতে পারেন। মেন পাইথন ইন্টারপ্রেটারে এর কোন প্রভাব পরবেনা!

ভার্চুয়াল এনভায়রনমেন্ট ব্যবহার করতে প্রথমে কমান্ড প্রম্পট (cmd) ওপেন করুন (লিনাক্সে টার্মিনাল ওপেন করুন)। এর পর `pip install virtualenv` কমান্ড দিয়ে ভার্চুয়াল এনভায়রনমেন্ট প্যাকেজটি ইন্সটল করুন। ইন্সটল হয়ে গেলে (বা আগে থেকেই ইন্সটল করা থাকলে) আপনি ভার্চুয়াল এনভায়রনমেন্ট ব্যবহার করার জন্য প্রস্তুত!

আমাদের জ্যাকসো প্রজেক্ট শুরু করার আগে আমরা ভার্চুয়াল এনভায়রনমেন্ট তৈরি করে নিব। কমান্ড প্রম্পট ওপেন করে এই কমান্ড দিন! `cd desktop` এতে উইন্ডোজের ডেস্কটপটি কারেন্ট ওয়ার্কিং ডিরেক্টরি হিসেবে সেট হবে। তারপর এই কমান্ডঃ `virtualenv myfirstenv`



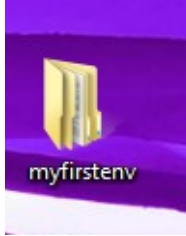
```

C:\Windows\system32\cmd.exe

C:\Users\user pc>cd desktop
C:\Users\user pc\Desktop>virtualenv myfirstenv
Using base prefix 'c:\python34'
New python executable in C:\Users\USERPC~1\Desktop\MYFIRS~1\Scripts\python.exe
Installing setuptools, pip, wheel...done.
C:\Users\user pc\Desktop>
  
```

ভার্চুয়াল এনভায়রনমেন্ট তৈরি হয়ে গেছে, এখন এটা একটিভ করতে হবে। একটিভ করা মানে হল এখন থেকে পাইথনের মাইন ইন্টারপ্রেটারের পরিবর্তে ভার্চুয়াল এনভায়রনমেন্ট এর মধ্যকার ইন্টারপ্রেটারটি কাজ করবে! আবার কাজ শেষে ডিএকটিভ করে দিতে হবে, তখন ভার্চুয়াল এনভায়রনমেন্ট এর ইন্টারপ্রেটার আর কাজ করবেনা।

লক্ষ্য করে দেখবেন ডেস্কটপে (বা ওয়ার্কিং ডিরেক্টরিতে) myfirstenv নামে নতুন একটা ফোল্ডার তৈরি হয়েছে



সেটাতে ডাবল ক্লিক করে ভিতরে যান, এরকম কয়েকটি ফোল্ডার দেখবেনঃ

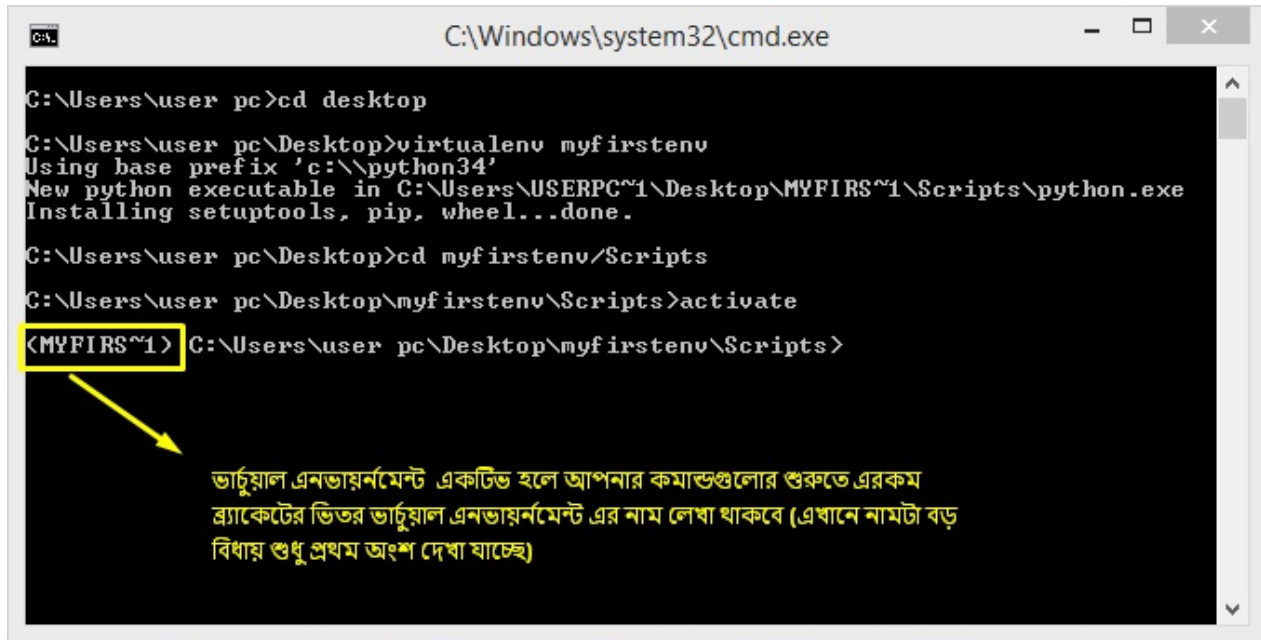
myfirstenv			
Name	Date modified	Type	
Include	3/23/2017 5:16 PM	File folder	
Lib	10/4/2017 12:02 PM	File folder	
Scripts	10/4/2017 12:03 PM	File folder	
tcl	10/4/2017 12:02 PM	File folder	

সেখানের Scripts ফোল্ডারটিতে (লিনাক্সের ক্ষেত্রে bin ডিরেক্টরিতে) গেলে এই ফাইলগুলো দেখা যাবেঃ

myfirstenv > Scripts				
Name	Date modified	Type	Size	
activate	10/4/2017 12:03 PM	File	3 KB	
activate.bat	10/4/2017 12:03 PM	Windows Batch File	1 KB	
activate.ps1	10/4/2017 12:03 PM	Windows PowerS...	9 KB	
activate_this.py	10/4/2017 12:03 PM	PY File	2 KB	
deactivate.bat	10/4/2017 12:03 PM	Windows Batch File	1 KB	
easy_install.exe	10/4/2017 12:03 PM	Application	88 KB	
easy_install-3.4.exe	10/4/2017 12:03 PM	Application	88 KB	
pip.exe	10/4/2017 12:03 PM	Application	88 KB	
pip3.4.exe	10/4/2017 12:03 PM	Application	88 KB	
pip3.exe	10/4/2017 12:03 PM	Application	88 KB	
python.exe	10/4/2017 12:02 PM	Application	27 KB	
pythonw.exe	10/4/2017 12:02 PM	Application	27 KB	
wheel.exe	10/4/2017 12:03 PM	Application	88 KB	

;))

ফাইলগুলোর নাম দেখে কি বুঝতে পারছেন এগুলো কি? না বুঝলেও সমস্যা নেই! এখানের একটা মাত্র ফাইল আমাদের লাগবে সেটা হল activate কমান্ড প্রম্পট যদি ওপেন করাই থাকে তাহলে সেখানে cd কমান্ড দিয়ে উপরের script ফোল্ডারে আসুনঃ Cd myfirstenv/Scripts এর পর activate কমান্ডটি লিখে এন্টার চাপুনঃ (লিনাক্সের ক্ষেত্রে source activate বা . activate) একটিভ হয়ে গেলে কমান্ড প্রম্পট বা টার্মিনালের লাইনগুলোর শুরুতে প্রথম ব্র্যাকেটের ভিতর ভার্চুয়াল এনভায়নমেন্টের নাম দেখে যাবে! কাজ শেষে ভার্চুয়াল এনভায়নমেন্ট ডিএকটিভেট করে দিতে হবে deactivate কমান্ড দিলেই কাজ হয়ে যাবে।



```

C:\Windows\system32\cmd.exe

C:\Users\user pc>cd desktop
C:\Users\user pc\Desktop>virtualenv myfirstenv
Using base prefix 'c:\python34'
New python executable in C:\Users\USERPC~1\Desktop\MYFIRS~1\Scripts\python.exe
Installing setuptools, pip, wheel...done.
C:\Users\user pc\Desktop>cd myfirstenv\Scripts
C:\Users\user pc\Desktop\myfirstenv\Scripts>activate
<MYFIRS~1> C:\Users\user pc\Desktop\myfirstenv\Scripts>

```

ভার্চুয়াল এনভায়নমেন্ট একটিভ হলে আপনার কমান্ডগুলোর শুরুতে এরকম ব্র্যাকেটের ভিতর ভার্চুয়াল এনভায়নমেন্ট এর নাম লেখা থাকবে (এখানে নামটা বড় বিধায় শুধু প্রথম অংশ দেখা যাচ্ছে)

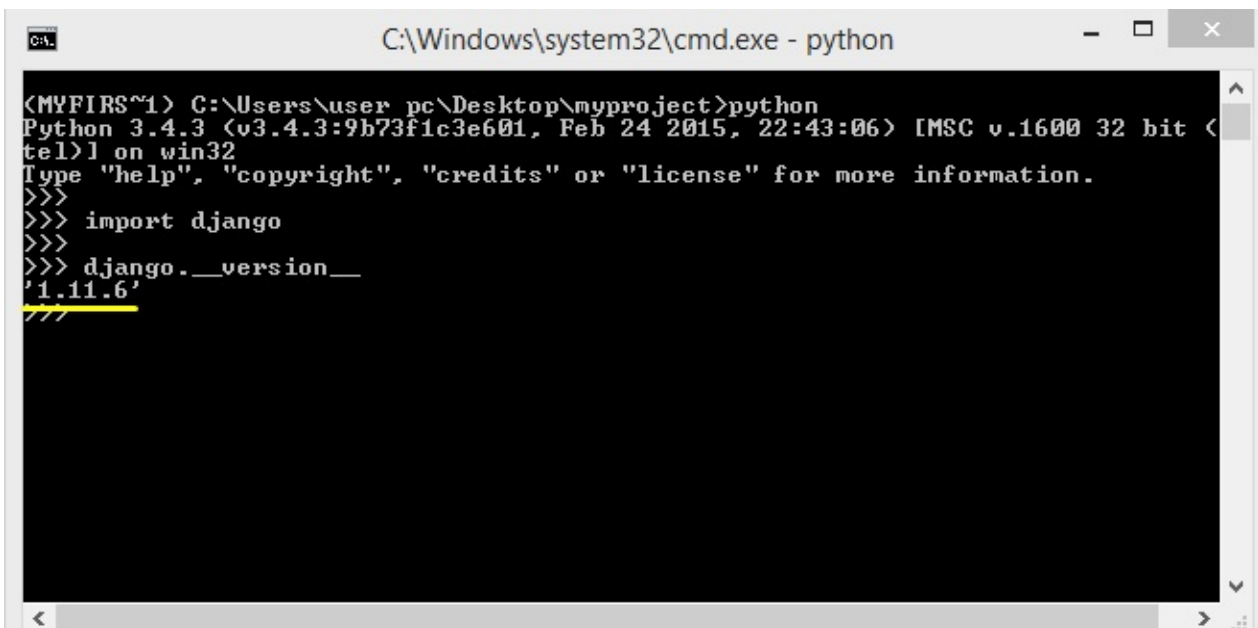
ভার্চুয়াল এনভায়নমেন্ট ইন্সটল, একটিভ করা হল! এখন জ্যাক্সো ইন্সটল করতে হবে! মনে রাখবেন, জ্যাক্সো কিন্তু ভার্চুয়াল এনভায়নমেন্ট এর মধ্যে ইন্সটল হবে/করতে হবে, আপনার মেইন পাইথন ইন্টারপ্রেটারে জ্যাক্সো ইন্সটল করা থাকলেও! কারন সেটা আমরা ব্যবহার করবনা, আমরা ব্যবহার করব ভার্চুয়াল এনভায়নমেন্ট এর জ্যাক্সো!

জ্যাংগো ইন্সটল

জ্যাংগো ইন্সটল করা অন্যান্য পাইথন প্যাকেজ ইন্সটল করার মতই সহজ, কমান্ড প্রম্পট বা টার্মিনালে ডাচুয়াল এনভায়নমেন্ট একটিভ করে নিন। তারপর `pip install django` কমান্ড এন্টার করুন! জ্যাংগোর স্ট্যাবল ভার্সন আপনার ডাচুয়াল এনভায়নমেন্টে ইন্সটল হয়ে যাবে।

ইন্টারনেট কানেকশন ঠিক আছে কিনা চেক করে নিন!

ভার্সন চেক করতে প্রথম কমান্ড প্রম্পট বা টার্মিনালে পাইথন চালু করুন, এর পর জ্যাংগো ইম্পোর্ট করুন `import django` যদি কোন ইরর দেখায় তাহলে বুঝতে হবে জ্যাংগো ঠিকভাবে ইন্সটল হয়নি। ইরর না দেখালে এটা লিখে এন্টার চাপুন `django.__version__` আপনার জ্যাংগো ভার্সন নাম্বার দেখতে পারবেন।



```
C:\Windows\system32\cmd.exe - python

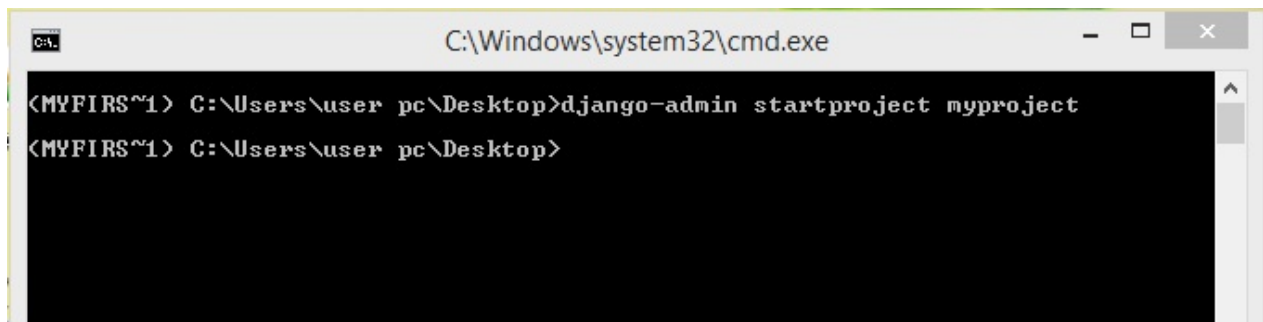
<MYFIRS~1> C:\Users\user pc\Desktop\myproject>python
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit <
tel>] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import django
>>>
>>> django.__version__
'1.11.6'
>>>
```

জ্যাংগো সঠিকভাবে ইন্সটল হয়ে গেলে আমরা আমাদের প্রথম প্রোজেক্ট শুরু করতে পারি...

প্রথম প্রজেক্ট শুরু

জ্যাক্সের ব্যাপারে বলা হয় যে এই ফ্রেমওয়ার্কটি দ্বারা খুবই দ্রুত হাই লেভেলের ওয়েবসাইট/ওয়েব অ্যাপ তৈরি করা যায়! আসলেও তুলনামূলক ভাবে খুব দ্রুত তৈরি করা যায় যদি সেটা হাই লেভেলের বা বড়সড় কোন ওয়েবসাইট হয়! এর মানে হল, ছোটখাটো হ্যাণ্ডো ওয়ার্ল্ড টাইপের ওয়েবসাইট/ওয়েবপেইজ তৈরি করতে গেলে জ্যাক্সের দ্রুততা চোখে পড়বেনা, এমনকি এটাকে খুবই ঝামেলার মনে হবে। হ্যাণ্ডোওয়ার্ল্ড টাইপের ওয়েবসাইট তৈরি করার জন্য জ্যাক্সো ব্যবহার করা মশা মারার জন্য কামান দাগানোর মত!!

যাইহোক, আমাদের প্রথম প্রজেক্ট আসলে হ্যাণ্ডো ওয়ার্ল্ড টাইপের ওয়েবসাইট তৈরি করা! অর্থাৎ এমন ওয়েবসাইট যেটাতে ভিজিট করলে আমরা Hello World! লেখাটি দেখতে পাবো! প্রথমে কমান্ড প্রম্পট (টার্মিনাল) ওপেন করুন, ভার্সুয়াল এনভায়নমেন্ট একটিভ করুন, কারেন্টওয়ার্কিং ডিরেক্টরি ঠিক করুন যেখানে আপনার প্রজেক্টটি বানাতে চান (উইন্ডোজের ডেস্কটপে প্রজেক্ট করতে চাইলে `cd desktop` লিখে ডেস্কটপকে কারেন্ট ওয়ার্কিং ডিরেক্টরি বানিয়ে নিন) এরপর নিচের কমান্ডটি লিখে এন্টার চাপুনঃ `Django-admin startproject myproject`



```
C:\Windows\system32\cmd.exe
<MYFIRS~1> C:\Users\user pc\Desktop>django-admin startproject myproject
<MYFIRS~1> C:\Users\user pc\Desktop>
```

Myproject হল প্রজেক্ট এর নাম, এটা আপনার ইচ্ছা অনুযায়ী যেকোন কিছু দিতে পারেন। আপনার কারেন্ট ডিরেক্টরিতে প্রজেক্টের নামে ফোল্ডার তৈরি হয়ে যাবেঃ



সেটার ভিতরে একটি ফাইল (manage.py) এবং একটি ফোল্ডার (myproject) থাকবে:

myproject		Search myproject		
Name	Date modified	Type	Size	
myproject	10/4/2017 7:39 PM	File folder		
manage.py	10/4/2017 7:39 PM	PY File	1 KB	

লক্ষ্য করুন, প্রোজেক্টের নামে কিন্তু ফোল্ডার মোট দুইটা, একটার ভিতর আরেকটা! বাইরের ফোল্ডার এর নাম রিনেম করে যা ইচ্ছা দিতে পারবেন সমস্যা নেই! তবে ভিতরের Myproject ফোল্ডারটিই আসলে মূল প্রজেক্ট! এটার নাম চেঞ্জ করবেননা! ভিতরের myproject ফোল্ডারে চারটা পাইথন মডিউল আছে, সেগুলোর সাথে একটু পরিচিত হয়ে নেই!

myproject > myproject		Search myproject		
Name	Date modified	Type	Size	
__init__.py	10/4/2017 7:39 PM	PY File	0 KB	
settings.py	10/4/2017 7:39 PM	PY File	4 KB	
urls.py	10/4/2017 7:39 PM	PY File	1 KB	
wsgi.py	10/4/2017 7:39 PM	PY File	1 KB	

init.py খালি ফাইল, ভিতরে কোড নেই! কোন ফোল্ডারকে পাইথন প্যাকেজ বানানোর জন্য এটা থাকে! **Settings.py** নাম দেখেই বোঝা যাচ্ছে, প্রোজেক্টের সব সেটিংসগুলো এই ফাইলে থাকবে। **Urls.py** ওয়েবসাইটের ইউআরএল গুলো এখানে ডিফাইন করা থাকবে! এসম্পর্কে আমরা পরে বিস্তারিত জানব। **Wsgi.py** সার্ভার সেটিংস, মাথা ঘামানোর দরকার নেই। প্রোজেক্ট ডেপ্লয় করার সময় লাগবে।

এছাড়াও বাইরের **manage.py** মডিউলটি খুবই গুরুত্বপূর্ণ! এখন থেকে আমাদের প্রায় সকল কমান্ডগুলই এই মডিউলের মাধ্যমে করতে হবে! কিভাবে করব সেটা আস্তে আস্তে জানব!

প্রোজেক্ট তৈরি করার কাজ শেষ, এখন দেখতে হবে যে প্রজেক্ট ঠিকমত কাজ করে কিনা !

ডেভেলপমেন্ট সার্ভার রান করা

জ্যাক্সো ওয়েবফ্রেমওয়ার্ক হল সার্ভার সাইড ফ্রেমওয়ার্ক! অর্থাৎ জ্যাক্সো দিয়ে ওয়েবসাইটের ব্যাকএন্ড তৈরি করা হয়, এবং তা সার্ভারের মাধ্যমে পরিচালিত হয়। আমরা যখন আমাদের প্রোজেক্টটি লাইভ/প্রোডাকশন সার্ভারে/হোস্টিংএ ডেপ্লয়/আপলোড করব তখন সেই সার্ভারটাই আমাদের জ্যাক্সো প্রোজেক্ট রান করবে! কিন্তু প্রোজেক্ট তৈরি করার সময় আমাদের নিজেদের কম্পিউটারে জ্যাক্সো কিভাবে রান করব?

এর সহজ সমাধান জ্যাক্সোর মধ্যেই রয়েছে, একটি লাইটওয়েট ডেভেলপমেন্ট সার্ভার! যেটা জ্যাক্সোর সাথে বিল্টইন ভাবে থাকে। আমরা সেটাই ব্যবহার করব!

কমান্ড প্রম্পটে (টার্মিনালে) ডারুয়াল এনভায়নমেন্ট একটিভ করুন। cd কমান্ড দিয়ে প্রোজেক্ট ফোল্ডারটিকে (যেটার ভিতর manage.py মডিউল রয়েছে) কারেন্ট ওয়ার্কিং ডিরেক্টরি হিসেবে সেট করুন! এর পর নিচের কমান্ড টি দিনঃ

```
Python manage.py runserver
```

সব ঠিক থাকলে আপনার ডেভেলপমেন্ট সার্ভার চালু হবে এবং কমান্ড প্রম্পটে নিচের মত তথ্য দেখা যাবেঃ

```
C:\Users\user pc\Desktop\myproject>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 13 unapplied migration(s). Your project may not work properly until you
apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
October 05, 2017 - 15:35:18
Django version 1.11.4, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

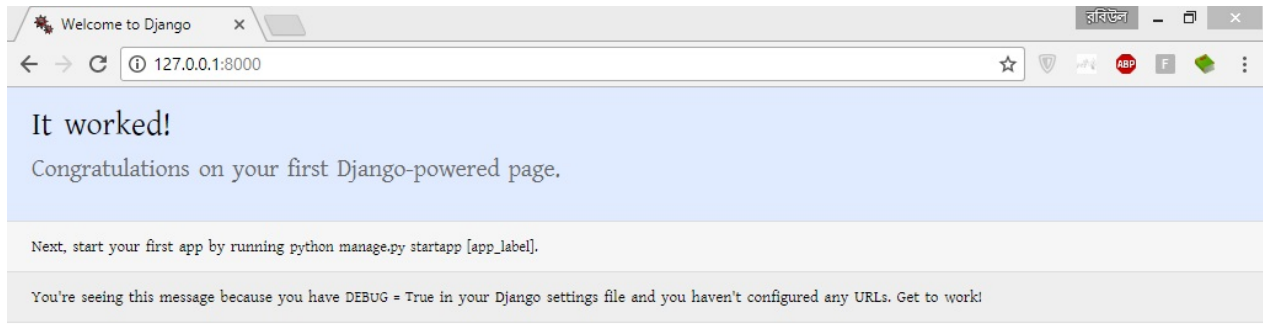
ডেভেলপমেন্ট সার্ভার চালু হয়ে গেল। এখন কোন একটি ওয়েবব্রাউজার ওপেন করুন এবং এই ঠিকানায় যানঃ

<http://127.0.0.1:8000/>

এটাই হল লোকাল ডেভেলপমেন্ট সার্ভারের আইপি+পোর্ট! (যখন প্রোজেক্টটি লাইভ সার্ভারে ডেপ্লয় করব তখন এটা কোন রিয়েল ডোমেইন হবে, যেমন <http://www.myfirtsdjangoproject.com> বা আপনার পছন্দের ডোমেইন!)

উপরের ঠিকানায় গেলে আপনি এরকম একটা পেইজ দেখতে পাবেন যেখানে লেখা থাকবে

It worked! Congratulations on your first Django-powered page.



যদি এরকম দেখেন তাহলে নিজেকে কংগ্রেটস দিন! আপনি সফলভাবে আপনার প্রথম জ্যান্সো পাওয়ার্ড ওয়েবসাইট তৈরি করে ফেলেছেন... যদিও এই সাইট (এখনো) কোনো কাজের না!

প্রতিবার আপনি আপনার প্রোজেক্টে কাজ করতে চাইলে এবং ওয়েবসাইট ভিজিট করতে চাইলে প্রথমেই কমান্ড প্রম্পট বা টার্মিনালে ডাচুয়াল এনভায়নমেন্ট একটিভ করবেন! এবং প্রোজেক্ট ফোল্ডারটিকে (যেখানে `manage.py` ফাইলটি আছে) ওয়ার্কিং ডিরেক্টরি হিসেবে সেট করবেন! অতপর (সার্ভার চালু করতে চাইলে) `python manage.py runserver` কমান্ড দিয়ে সার্ভার চালু করবেন! আপনার লোকাল সার্ভারের আইপি (ঠিকানা) হবে <http://127.0.0.1:8000/> ব্রাউজারের এড্রেসবারে এই ঠিকানা লিখে এন্টার চাপলেই আপনার সাইটের কনটেন্ট দেখতে পাবেন!

আমাদের এই প্রজেক্টটিকে কাজের হিসেবে তৈরি করতে হলে আমাদেরকে পরিচিত হতে হবে জ্যান্সো অ্যাপস এর সাথে... ভয় নেই, আমরাতো আছিই...

অ্যাপস কি? প্রজেক্ট ও অ্যাপস এর পার্থক্য!

জ্যাস্টোতে বিগিনারদের কনফিউশনের কয়েকটি বিষয়ের একটি হল জ্যাস্টো প্রোজেক্ট এবং অ্যাপস এর পার্থক্য! যদিও বিষয়টি খুব সহজ, কিন্তু নতুন অনেকের কাছে এটা অদ্ভুত লাগে বিধায় এ নিয়ে একটু আলোচনা করছি।

প্রোজেক্ট কি?

স্বাভাবিক ভাবে বলতে গেলে আমরা যে ওয়েব বেজ্‌ড সফটওয়্যার বা ওয়েবসাইট তৈরি করি সেটাই হল প্রজেক্ট। যেমন আমরা যদি জ্যাস্টো দিয়ে একটা ব্লগ তৈরি করি তাহলে সেটা একটা জ্যাস্টো প্রোজেক্ট, আবার যদি একটা ইআরপি সিস্টেম বা সোশ্যাল নেটওয়ার্ক তৈরি করি তাহলে সেগুলোও হবে একেকটা প্রোজেক্ট। অর্থাৎ আমাদের তৈরি করা সম্পূর্ণ সাইট বা সিস্টেমকেই জ্যাস্টোর পরিভাষায় প্রোজেক্ট বলা হবে।

অ্যাপস কি?

অ্যাপ্‌ডেভ এর অ্যাপ বা অ্যাপলিকেশন এর সাথে আমরা সবাই পরিচিত, কম্পিউটার বা মোবাইলের সফটওয়্যার গুলোকেই সাধারণত আমরা অ্যাপ হিসেবে চিনি। জ্যাস্টোতে অ্যাপ আসলে সেরকম কিছু নয়! জাস্ট প্রোজেক্টের বিভিন্ন অংশ!! অর্থাৎ বিভিন্ন অ্যাপস মিলে একটা প্রোজেক্ট তৈরি হয় বা বলা যায় একটা প্রোজেক্ট আসলে এক বা একাধিক প্রোজেক্টের সমষ্টি!!!

একটা উদাহরণ দেই, গাড়ি তৈরির কোম্পানিগুলো গাড়ির সম্পূর্ণ অংশ কিন্তু এক জায়গাতে, একসাথে তৈরি করেনা! তাদের ওয়ার্কশপে বিভিন্ন টিম থাকে, একেক টিম গাড়ির একেক অংশ ভিন্ন ভিন্ন ভাবে তৈরি করে, কোন টিম ইঞ্জিন তৈরি করে, কোন টিম বডি তৈরি করে, আবার কোন টিম গাড়ির পেইন্টিং করে। গাড়ির বিভিন্ন অংশ তৈরি হয়ে গেলে সেটা আবার অন্য কোন টিম একত্রে জোড়া দিয়ে সম্পূর্ণ গাড়ি তৈরি করে ফেলে!

এখানে সম্পূর্ণ গাড়ি হল একটি প্রজেক্ট, আর সেটার বিভিন্ন অংশ হল একেকটি অ্যাপ! বডি একটি অ্যাপ, ইঞ্জিন একটি অ্যাপ, চাকা একটি অ্যাপ! এই সকল অ্যাপস যুক্ত হয়ে গাড়ি প্রজেক্টটি তৈরি হয়!

আরেকটা উদাহরণ দেই, আমরা আমাদের ডেস্কটপ কম্পিউটারের দিকে লক্ষ্য করলে দেখব যে শুধু মনিটরকে বা শুধু সিপিইউ কে কিন্তু কম্পিউটার বলা হয় না। মনিটর, সিপিউ, কিবোর্ড, মাউস, স্পিকার ইত্যাদি মিলেই কিন্তু কম্পিউটার। তাই জ্যাস্টোর পরিভাষায় এখানে আমরা কম্পিউটারকে প্রোজেক্ট বলতে পারি এবং এটা বলতে পারি যে কম্পিউটার প্রোজেক্টটি তৈরি হয়েছে মনিটর অ্যাপ, সিপিউ অ্যাপ, কিবোর্ড অ্যাপ বা মাউস, স্পিকার, রাউটার অ্যাপ এর মত অ্যাপস গুলোকে একত্রিত করে!

জ্যাস্টোতে তৈরি করা ওয়েবসাইট গুলোর ব্যপারটাও এরকম। ধরি আমরা একটা ব্লগ তৈরি করব, তাহলে সে সম্পূর্ণ ব্লগটা হবে একটা প্রোজেক্ট, আর সেই ব্লগের বিভিন্ন অংশ যেমন সেটার পোস্ট করার সিস্টেম, কमेंট সিস্টেম, এডমিন ইন্টারফেইস, অ্যানালিটিক্স সিস্টেম ইত্যাদি ছোট ছোট অংশগুলো হল একেকটা অ্যাপ। এই সকল অ্যাপস নিয়েই তৈরি হয় আমাদের ব্লগ প্রোজেক্ট!

প্রোজেক্টকে বিভিন্ন অ্যাপ এ ভাগ করার প্রয়োজন কি?

এর অনেকগুলো সুবিধা রয়েছে। যেমন সম্পূর্ণ প্রোজেক্ট একসাথে মেনেটেন করার চেয়ে ছোট ছোট অ্যাপ আলাদা আলাদা ভাবে মেনেটেন করা সহজ। গাড়ি বা কম্পিটারের কোন অংশ নষ্ট হলে যেমন শুধু সে অংশটা ঠিক করিয়ে নিলেই হয়, সম্পূর্ণ গাড়ি বা কম্পিটার পরিবর্তন বা মেরামত করাতে হয়না। এখানেও কোন অ্যাপে সমস্যা হলে শুধু মাত্র সেই অ্যাপ নিয়েই মাথা ঘামাতে হয়, সেই অ্যাপটা ঠিক করলেই হয়, সম্পূর্ণ প্রোজেক্ট নিয়ে মাথা ঘামানোর প্রয়োজন পরেনা। রিইউজঅ্যাবিলিটি বাড়ে, অর্থাৎ আপনি চাইলে এক প্রোজেক্টে তৈরি করা অ্যাপ অন্য প্রোজেক্টে ব্যবহার করতে পারবেন, এমনকি অন্যের তৈরি করা অ্যাপও নিজের প্রোজেক্টে ব্যবহার করতে পারবেন... কম্পিটার বা গাড়ির পার্টসগুলোর মত!

প্রথম অ্যাপ তৈরি

আমরা প্রোজেক্ট তৈরি করেছি, ডেভেলপমেন্ট সার্ভার রান করেছি, এখন অ্যাপ তৈরি করব! প্রথমেই ডারুয়াল এনভায়রনমেন্ট একটিভ করুন এবং প্রোজেক্ট ফোল্ডারকে কারেন্ট ডিরেক্টরি করুন। এরপর এই কমান্ড দিনঃ

```
python manage.py startapp myapp
```

প্রথম প্রজেক্ট তৈরি করার সময়ও আমরা এরকম একটা কমান্ড দিয়েছিলাম, তবে সেটা django-admin টুল ব্যবহার করে এবং এটা manage.py মডিউল ব্যবহার করে। এখানে অ্যাপ এর নাম দিয়েছি myapp আপনি যেকোন নাম দিতে পারেন। এখন আপনার প্রোজেক্ট ফোল্ডারে myapp নামে নতুন একটি ফোল্ডার তৈরি হবে, সেটাই অ্যাপ ফোল্ডার। ফোল্ডারটির ভিতরে কয়েকটি পাইথন ফাইল দেখতে পাবেন, সেগুলোঃ

myproject > myapp					Search myapp
	Name	Date modified	Type	Size	
	migrations	10/6/2017 9:30 PM	File folder		
	__init__.py	10/6/2017 9:30 PM	PY File	0 KB	
	admin.py	10/6/2017 9:30 PM	PY File	1 KB	
	apps.py	10/6/2017 9:30 PM	PY File	1 KB	
	models.py	10/6/2017 9:30 PM	PY File	1 KB	
	tests.py	10/6/2017 9:30 PM	PY File	1 KB	
	views.py	10/6/2017 9:30 PM	PY File	1 KB	

১) migrations নামে একটি ফোল্ডার দেখবেন, ইগনোর করুন! ২) **init.py** ফাকা ফাইল! এই ফোল্ডারটিকে পাইথন প্যাকেজ বানানোর জন্য। ৩) **admin.py** এডমিন অ্যাপ সম্পর্কিত সেটিংসগুলো এখানে থাকবে, এডমিন বিষয়ে সামনে জানতে পারব! ৪) **apps.py** এই অ্যাপ স্পেসিফিক সেটিংস গুলো এই ফাইলে থাকবে। ৫) **models.py** মডেল এর কথা মনে আছেতো!? এমটিভি প্যাটার্নের মডেল। আপনার অ্যাপ এর সকল মডেল এই ফাইলে থাকবে। ৬) **tests.py** প্রাথমিক ভাবে জ্যাক্সকে যতই কঠিন আর গোজামিল টাইপের মনে হোক না কেন, জ্যাক্সে আসলে খুবই অর্গানাইজড এবং বেস্ট প্র্যাকটিস ফলো করে তৈরি একটা ফ্রেমওয়ার্ক! তাই জ্যাক্সে সব সময় চায় যে তার (ব্যবহারকারী) ডেভেলপারগণও সবসময় বেস্ট প্র্যাকটিস ফলো করবে... আর কোড টেস্টিং বা ইউনিট টেস্টিং হল বেস্ট প্র্যাকটিসের অন্যতম একটা অংশ! ইউনিট টেস্ট বা টেস্টিং কোডগুলো লেখার জন্য জ্যাক্সে অটোমেটিক আপনার জন্য এই ফাইল তৈরি করে দেয়!!! ৭) **views.py** ভিউ, আপনার ওয়েবসাইটের কেন্দ্রবিন্দু... এই ফাইলেই সকল ভিউ তৈরি করা হবে!

আমাদের প্রথম অ্যাপ তৈরি হয়ে গেছে, কিন্তু এটা এখনো আমাদের প্রোজেক্টে ব্যবহারযোগ্য হয়ে উঠেনি! আপনি কি বলতে পারবেন কেন?

সেটিংস পরিচিতি

আমরা [অ্যাপ/প্রোজেক্ট] আগের চ্যাপ্টারে জেনেছি যে জ্যান্সোর অ্যাপগুলো রিইউজঅ্যাবল! অর্থাৎ আপনি একটা অ্যাপ তৈরি করে সেটা যেকোন প্রোজেক্টে ব্যবহার করতে পারবেন। এমনকি অন্যের তৈরি অ্যাপ নিজের প্রোজেক্টে ব্যবহার করতে পারবেন! প্রতিটি অ্যাপই স্বয়ংসম্পূর্ণ। একারণে যখন আপনি কোন অ্যাপ তৈরি করেন, অথবা অন্য কোথাও থেকে নিয়ে আসেন সেটা অটোমেটিক জ্যান্সোতে এড হয়ে যায়না! ম্যানুয়ালি এড করে নিতে হয়। খুব সহজেই আপনি অ্যাপ এড করতে পারেন, সেটার জন্য যা করতে হবেঃ প্রথমে প্রোজেক্ট ফোল্ডারে থাকা settings.py ফাইলটি কোন টেক্সট এডিটরে ওপেন করুন!

myproject/settings.py

ফাইলের মধ্যে INSTALLED_APPS নামে একটা লিস্ট দেখতে পাবেন! যেটার ভিতরে এরকম কয়েকটা আইটেম থাকবেঃ

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

আমাদের কাজ হল আমাদের তৈরি করা অ্যাপ এর নামটা এই লিস্টে ঢুকিয়ে দেয়া! লিস্টটি এডিট করে এরকম করে ফেলুনঃ

```
INSTALLED_APPS = [  
    'myapp',  
  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

লক্ষ্য করুন, আমরা লিস্টের ফার্স্ট আইটেম হিসেবে আমাদের অ্যাপ এর নাম স্ক্রিং হিসেবে দিয়ে দিয়েছি! আপনি হয়তো বুঝে গেছেন যে লিস্টের বাকি আইটেমগুলোও আসলে বিভিন্ন অ্যাপ এর নাম, যেগুলো আসলে খুবই কাজের, কমন কিছু বিষয় নিয়ে অহেতুক কষ্ট করে কোড লেখা থেকে আমাদের বাঁচিয়ে দেয়ার জন্য এগুলো জ্যান্সোতে বিল্ডিন ভাবেই থাকে, আমরা এ অ্যাপগুলো নিয়ে আস্তে আস্তে জানতে পারব!

সেটিংস ফাইলে এই লিস্ট ছাড়াও আরো কিছু লিস্ট/ভেরিয়ারবল দেখতে পাবেন, সবগুলো সম্পর্কেই জানব ইনশাআল্লাহ... ধিবে ধিবে...

সিম্পল ডিউ ফাংশন

[ওয়েব যেভাবে কাজ করে] থেকে আমরা জেনেছি যে ক্লায়েন্ট (ব্রাউজার) সার্ভারে (ওয়েবসাইটে) রিকুয়েস্ট করার পর সার্ভার সেই রিকুয়েস্ট অনুযায়ী রেসপন্স সেন্ড করে! আর [এমডিটি প্যাটার্ন] ও [জ্যাক্সো ফ্রেমওয়ার্ক] থেকে আমরা জেনেছি যে ওয়েব ফ্রেমওয়ার্কে ডিউ এর কাজ হচ্ছে ক্লায়েন্ট এর রিকুয়েস্ট অনুযায়ী রেসপন্স তৈরি করে সেন্ড করা।

তো এখন আমরা একটা ডিউ তৈরি করব যেটা আমাদেরকে Hello User, Welcome! লেখাটি সেন্ড করবে! অর্থাৎ ব্রাউজারে যদি আমরা আমাদের সাইটের ঠিকানা (127.0.0.0:8000) লিখে এন্টার চাপি তাহলে সেখানে Hello User, Welcome! লেখাটি দেখতে পারব!

ডিউ তৈরি করা হয় ফাংশন লিখে, এবং এই ফাংশনগুলো লেখা হয় অ্যাপ এর views.py ফাইলে! যেটা অ্যাপ তৈরি করার সময় জ্যাক্সো অটোমেটিক আমাদের জন্য তৈরি করে দেয়! আমাদের তৈরি অ্যাপ myapp এর ভিতরের views.py ফাইলটা ওপেন করুন। সেখানে এরকম দুটি লাইন দেখতে পাবেনঃ

```
from django.shortcuts import render

# Create your views here.
```

নিচের লাইনটি পাইথন কমেন্ট, আর উপরের লাইনটি একটি ইম্পোর্ট স্টেটমেন্ট যেটা render ফাংশন ইম্পোর্ট করছে! আপাতত এটা নিয়ে মাথা না ঘামাই! কমেন্ট এর নিচ থেকে একটা ফাংশন লেখা শুরু করুন এরকম ভাবেঃ

```
def index(request):
    return HttpResponse("Hello World!")
```

মাত্র দুই লাইন! যেহেতু আপনি পাইথন জানেন তাই বুঝতেই পারছেন যে ফাংশনটা একটা request নামে প্যারামিটার গ্রহণ করবে এবং HttpResponse("Hello World!") এই অবজেক্টটাকে রিটার্ন করবে!

আমরা জানি যে ব্রাউজারের রিকুয়েস্ট গ্রহন করা এবং সে অনুযায়ী রেসপন্স রিটার্ন করা ডিউ এর দায়িত্ব! তাই জ্যাক্সোর প্রতিটি ডিউ (উপরের ডিউ ফাংশনটি সহ) সব সময় একটা request গ্রহন করবে (যেটা আসলে ব্রাউজারের রিকুয়েস্ট!) এবং একটা রেসপন্স রিটার্ন করবে!

উপরের ডিউ ফাংশনটি যে রেসপন্স অবজেক্ট রিটার্ন করছে সেটা হল এইচটিটিপি রেসপন্স! যা HttpResponse ক্লাসটির মাধ্যমে তৈরি হচ্ছে, এবং সেটা "Hello User, Welcome" স্ট্রিং টিকে আর্গুমেন্ট হিসেবে গ্রহন করছে। মূলত এই স্ট্রিং আর্গুমেন্টটাই রেসপন্স হিসেবে সেন্ড হবে এবং ক্লায়েন্ট এর ব্রাউজারে প্রদর্শিত হবে!

পাইথনিস্টা হিসেবে এতক্ষণে আপনার মনে এই প্রশ্ন উদ্ভূত হবার কথা যে আমরা এই HttpResponse ক্লাসটি কোথায় পেলাম!? আসলে এটা এখনো কোথাও পাইনি, বরং এটাকে ইম্পোর্ট করে নিতে হবে! Views.py ফাইলের একদম উপরের ইম্পোর্ট স্টেটমেন্টটির সাথে এটাও যোগ করে দিন এরকম করেঃ

```
from django.shortcuts import render, HttpResponse
```

আমাদের কাছে এখন একটা ডিউ ফাংশন আছে যেটা ব্রাউজারের রিকুয়েস্ট পেলে এইচটিটিপি রেসপন্স হিসেবে “Hello User, Welcome” স্ট্রিং টি রিটার্ন করবে। আমাদের সম্পূর্ণ ডিউ মডিউলের কোডঃ

```
from django.shortcuts import render, HttpResponseRedirect

# Create your views here.
def index(request):
    return HttpResponseRedirect("Hello World!")
```

ডিউ এর কাজ শেষ, এখন এটা ঠিক করতে হবে যে ওয়েবসাইটের কোন ইউআরএল (URL) এ ক্লায়েন্ট রিকুয়েস্ট করলে এই ডিউ ফাংশনটি তার কাজ করবে!

ইউআরএল কনফিগারেশন

ইউআরএল (URL) হল ওয়েবসাইটের ঠিকানা, যেমন <http://google.com/> এটা একটা ইউআরএল, আবার <http://django.howtocode.com.bd/> এটাও একটা ইউআরএল, আবার <http://django.howtocode.com.bd/tutorial/2.html> এটাও একটা ইউআরএল!

একটা ওয়েবসাইটের অনেক/অগণিত ইউআরএল থাকতে পারে, একেক ইউআরএলে একেকটা বিষয় থাকতে পারে, আপনি ইউআরএল গুলোকে কম্পিউটারে ফোল্ডার/ডিরেক্টরি হিসেবে কল্পনা করতে পারেন। যেমন ধরুন কম্পিউটারে যদি আপনি এই `mycomputer/user/songs/` ডিরেক্টরিতে যান তাহলে সে ডিরেক্টরিতে থাকা সকল গান এর লিস্ট দেখতে পাবেন। আবার যদি এই `mycomputer/user/images/` ডিরেক্টরিতে যান তাহলে সে ডিরেক্টরিতে থাকা সকল ইমেজের লিস্ট দেখতে পাবেন।

ওয়েবসাইটের ইউআরএল এর ব্যপারটাও এমন, যেমন ধরুন আপনি যদি এই <http://django.howtocode.com.bd/> ইউআরএল এ যান তাহলে আপনার সামনে জ্যাক্সো টিউটোরিয়ালের হোম পেইজ প্রদর্শিত হবে, আবার যদি <http://django.howtocode.com.bd/tutorial/2.html> ইউআরএল এ যান তাহলে জ্যাক্সো টিউটোরিয়ালের একটা স্পেসিফিক টিউটোরিয়াল পেইজ আপনার সামনে প্রদর্শিত হবে।

আমরা জেনেছি যে ব্রাউজারে যখন আমরা কোন ইউআরএল লিখে এন্টার চাপি তখন সেখান থেকে একটা রিকুয়েস্ট সেই ইউআরএল বা ওয়েবসাইটের সার্ভারে চলে যায়! তারপর সেই ওয়েবসাইটের সার্ভার সিদ্ধান্ত নেয় যে রিকুয়েস্ট করা ব্রাউজারের কাছে কোন বিষয়গুলো রেসপন্স হিসেবে পাঠানো হবে, আমরা এটাও জেনেছি যে জ্যাক্সোতে এই সিদ্ধান্ত নেয়ার কাজটা আসলে 'ভিউ' করে থাকে!।

আমরা যেটা এখনো জানি না সেটা হল, ইউজার ঠিক কোন ইউআরএল এ রিকুয়েস্ট করলে আমাদের প্রোজেক্টের কোন ভিউটা এক্সিকিউট হবে এবং ইউজারকে রেসপন্স পাঠাবে!

আগের চ্যাপ্টারে আমরা একটা ভিউ তৈরি করেছি `index` নামে। এখন আমাদেরকে ঠিক করে দিতে হবে যে কোন ইউআরএল এ রিকুয়েস্ট আসলে এই ভিউটা তার 'Hello World!' লেখাটি প্রদর্শন করবে! ইউআরএল গুলো নিয়ে কাজ করার জন্য আমাদের প্রোজেক্ট ফোল্ডারে `urls.py` নামে একটা ফাইল পাবেন, সেটা টেক্সট এডিটরে ওপেন করুন।

প্রথম দেখায় মনে হতে পারে যে সেখানে হাজার হাজার লাইন কোড লেখা রয়েছে, কিন্তু একটু খেয়াল করলেই দেখবেন যে উপরের বিশাল অংশটি আসলে কমেস্ট! ইগনোর করুন। মূল কোড হল নিচের দিকের কোড গুলো, যা দেখতে এরকমঃ

```
from django.conf.urls import url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
]
```

প্রথম লাইনে url ফাংশনটি ইম্পোর্ট করা হয়েছে, সেটা একটু পরই কাজে লাগবে। দ্বিতীয় লাইনে এডমিন অ্যাপ ইম্পোর্ট করা হয়েছে। এডমিন কি তা নিয়ে আমরা পরে আলোচনা করব, এখানে এতটুকু বুঝুন যে এডমিন অ্যাপটি এখানে ইম্পোর্ট করা হয়েছে এটা ঠিক করে দেয়ার জন্য যে সেই অ্যাপটা ঠিক কোন ইউআরএল এ গেলে পাওয়া যাবে!

এর পর urlpatterns নামে একটা লিস্ট দেখা যাচ্ছে, যার মধ্যে ইউআরএল ম্যাপারগুলো থাকবে, এবং কোন ইউআরএল টা কোন ভিউকে কল করবে সেটা থাকবে। আপাতত লিস্টে একটা মাত্র আইটেম রয়েছে

```
url(r'^admin/', admin.site.urls)
```

url() এর ভিতরে সাধারণত দুটি প্যারামিটার থাকবে। একটা হল ইউআরএল ম্যাপার, যা রেগুলার এক্সপ্রেশন হিসেবে লেখা হবে। আরেকটা প্যারামিটার হল ভিউ। ব্রাউজারের রিকুয়েস্ট করা কোন ইউআরএল যদি এই লিস্টের কোন ইউআরএল ম্যাপারের সাথে ম্যাচ করে তাহলে সেই ম্যাপারের সাথে সংশ্লিষ্ট ভিউটি কল হবে।

কঠিন লাগছে? আমাদের ভিউটি এড করলে বুঝতে আরো সহজ হবে। প্রথমেই আমাদের index ভিউটি ইম্পোর্ট করে নিতে হবেঃ

```
from myapp.views import index
```

এখন urlpatterns লিস্টে একটা আইটেম যোগ করে দিতে হবেঃ

```
url(r'^myview/$', index)
```

url() এর দ্বিতীয় প্যারামিটার হিসেবে আমাদের ভিউটিকে পাস করলাম, আর প্রথম প্যারামিটার হিসেবে একটা ইউআরএল ম্যাপার দিলাম। r'^myview/\$' ম্যাপারটি শুধুমাত্র এই ইউআরএল এর সাথে ম্যাচ করবেঃ

www.mysite.com/myview/

(যেটা আমাদের লোকাল সার্ভারের ক্ষেত্রে এরকমঃ <http://127.0.0.1:8000/myview/>)

উক্ত ইউআরএল এ কোন ব্রাউজার রিকুয়েস্ট করলে আমাদের index ভিউটি কল হবে এবং সেটা “Hello World!” স্ট্রিং টি রেসপন্স হিসেবে পাঠিয়ে দিবে, আর ব্রাউজার সেই স্ট্রিংটি সুন্দর ভাবে আমাদের সামনে প্রদর্শন করবে!

লক্ষ্য করুন, ইউআরএল ম্যাপার কিন্তু আপনার ওয়েবসাইটের ডোমেইন নেম এর সাথে ম্যাচ করবেনা, ডোমেইন এর পর থেকে ম্যাচ করবে। অর্থাৎ সাইটের ইউআরএল যদি হয় www.mysite.com/mypy তাহলে এখানে www.mysite.com টিকে কোন ইউআরএল ম্যাপারের সাথে ম্যাচ করা হবেনা, বরং চেক করে দেখা হবে যে ডোমেইনের পরের অংশ ম্যাচ করে কিনা, এক্ষেত্রে শুধু mypy কোন ইউআরএল ম্যাপারের সাথে ম্যাচ করে কিনা দেখা হবে!

আমাদের urls.py সাইটের মূল কোডগুলো একসাথে এরকম হবেঃ

```
from django.conf.urls import url
from django.contrib import admin

from myapp.views import index

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^myview/', index),
]
```

এক নজরে: জ্যাঙ্গোতে কোন ব্রাউজার থেকে রিকুয়েস্ট আসলে জ্যাঙ্গো যেভাবে তা হ্যান্ডেল করে: ১) রিকুয়েস্ট আসার পর জ্যাঙ্গো তার মেইন ইউআরএল কনফিগারেশন সেটিংস লোড করে, অর্থাৎ urls.py ফাইল লোড করে। ২) রিকুয়েস্ট করা ইউআরএল টিকে urls.py এর urlpatterns লিস্টে থাকা ম্যাপারগুলোর সাথে সিরিয়ালে একটা একটা করে চেক করে দেখে কোনটা মিলে কিনা, মিললে সেই ম্যাপার সংশ্লিষ্ট ভিউকে কল করে। ২) ভিউটি ইউআরএল দেখে সে অনুযায়ী জিনিস পত্র তৈরি কর সেটা রেসপন্স হিসেবে ব্রাউজারকে পাঠিয়ে দেয়!

ইউআরএল কনফিগারেশন সঠিক ভাবে সম্পন্ন হল কিনা সেটা চেক করে দেখা দরকার! ডেভ সার্ভার চালু করুন:

```
python manage.py runserver
```

অতঃপর ব্রাউজারে ইউ ইউআরএল লিখে এন্টার চাপুন: <http://127.0.0.1:8000/myview> যদি Hello World! লেখাটি দেখা যায় তাহলে আপনি সফল, না দেখা গেলে চ্যাপ্টারটি পুনরায় পড়ুন এবং আবার চেষ্টা করুন!!

টেমপ্লেট ব্যবহার

আমাদের index ভিউটি “Hello World!” লেখাটিকে রেসপন্স হিসেবে পাঠাচ্ছে, যদিও এটা একটা স্ট্রিং কিন্তু ব্রাউজার এটাকে এইচটিএমএল (HTML) হিসেবেই রেন্ডার বা প্রদর্শন করে! এর মানে হল, ভিউটি যদি ‘শুধু স্ট্রিং’ এর বদলে এইচটিএমএল ট্যাগ সহ স্ট্রিং রিটার্ন করত তাহলে সেটাকেও ব্রাউজার সুন্দর ভাবে প্রদর্শন করতে পারত।

বিষয়টা হাতে কলমে করে দেখা যাক, ইন্ডেক্স ভিউটিকে আপডেট করে স্ট্রিং এর মধ্যে একটা এইচটিএমএল ট্যাগ বসিয়ে দিন এভাবেঃ

```
def index(request):  
    return HttpResponse("<h1>Hello World!</h1>")
```

আমরা হ্যালো ওয়ার্ল্ড লেখাটিকে এইচটিএমএল এর H1 ট্যাগ এর ভিতরে রাখলাম, খেয়াল করুন, ট্যাগসহ পুরো লাইনটাই কিন্তু স্ট্রিং হিসেবে আছে!

এখন আবার ব্রাউজারে এইঠিকানায় যায় <http://127.0.0.1:8000/myview/>

(ডেভ সার্ভার চালু করতে ভুলবেননা যান) । কোন পরিবর্তন দেখতে পেলেন? আমাদের Hello World! লেখাটি এখন আগের চেয়ে বড় দেখা যাচ্ছে!

ব্রাউজার প্রতিটা রেসপন্সকেই আসলে এইচটিএমএল ওয়েবপেইজ হিসেবে প্রদর্শন করে! সমস্যা হল, আমাদের রেসপন্স কিন্তু সব সময় এরকম হ্যালো ওয়ার্ল্ড এর মধ্যে সীমাবদ্ধ থাকবেনা, আরো অনেক কনটেন্ট যুক্ত হবে, অনেক এইচটিএমএল, সিএসএস, জাভাস্ক্রিপ্ট কোড যুক্ত হবে... এমনকি রেসপন্সটা কয়েক হাজার লাইনের এইচটিএমএল/সিএসএস যুক্ত কনটেন্ট হতে পারে!

আমাদের ইন্ডেক্স ভিউ ফাংশনটিতেই যদি আমরা সম্পূর্ণ এইচটিএমএল কোড যুক্ত করি তাহলে সেটা এরকম হবেঃ

```
def index(request):  
    response_string = """<!DOCTYPE html>  
<html>  
  <head>  
    <title>My view</title>  
  <head>  
  <body>  
    <h1>Hello World!</h1>  
  </body>  
</html>"""  
  
    return HttpResponse(response_string)
```

এটা এখনই বিদ্যুটে লাগছে! এরপর যদি সিএসএস যুক্ত ফুল ফিচার্ড একটা এইচটিএমএল পেইজ করতে চাই তাহলে ভিউ ফাংশনকে আর খুঁজেই পাওয়া যাবেনা !

এই সমস্যার সহজ সমাধান হল জ্যাকো ট্যামপ্লেট ল্যান্ডুয়েজ! আপনি ডিউ ফাংশন থেকে রেসপন্স এর কনটেন্ট (এইচটিএমএল/সিএসএস/জাভাস্ক্রিপ্ট) আলাদা করে ফেলবেন, জাস্ট অন্য একটা ফাইলে সকল কনটেন্ট থাকবে, ডিউ ফাংশনটা শুধু সেই ফাইলকে রেন্ডার করে রেসপন্স হিসেবে রিটার্ন করবে!

টেমপ্লেট ব্যবহার করতে প্রথমে myapp ফোল্ডারের ভিতরে templates নামে একটা ফোল্ডার তৈরি করুন! তারপর সে টেমপ্লেট ফোল্ডারে index.html নামে একটা ফাইল তৈরি করুন! ডিরেক্টরি দেখতে এরকম হবে myproject/myapp/templates/index.html

তারপর index.html ফাইলটিকে টেক্সট এডিটরে ওপেন করে এই কনটেন্টগুলো যুক্ত করে দিনঃ

```
<!DOCTYPE html>
<html>
  <head>
    <title>My view</title>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

এখন ডিউ ফাংশনটিকে এডিট করে এরকম করুনঃ

```
def index(request):
    return render(request, 'index.html')
```

ডিউ থেকে এইচটিএমএল কনটেন্টগুলো আলাদা করা গেল, তবে ডিউতে আমরা এখন HttpResponse এর পরিবর্তে render অবজেক্ট রিটার্ন করছি, render দুটি প্যারামিটার গ্রহন করছে, একটি হল request অবজেক্ট, আপাতত এটা ইগনোর করুন (তবে render এর প্রথম আর্গুমেন্ট হিসেবে দিতে ভুলবেননা)! দ্বিতীয় আর্গুমেন্ট হিসেবে আছে আমাদের টেমপ্লেট ফাইল (যেটা একটু আগে তৈরি করলাম) এর নাম, স্ট্রিং হিসেবে।

views.py ফাইল এর প্রথম লাইনে দেখবেন render ফাংশন ইম্পোর্ট করাই আছে, তবে না থাকলে ইম্পোর্ট করে নিনঃ

```
from django.shortcuts import render
```

এখন আবার ডেভ সার্ভার চালু করে ব্রাউজারে <http://127.0.0.1:8000/myview/> ঠিকানায় যান, কোন পরিবর্তন দেখতে পাবেননা আশা করি, আগের মতই Hello World! লেখাটে বড় করে দেখা যাবে! যদি এরকমটি না হয় তাহলে এই চ্যাপ্টারটি আবার শুরু থেকে পড়ুন, খুঁজে বের করুন কোথাও কোন ভুল হয়েছে কিনা!

আপনার কাছে এখন হয়ত এটা মনে হতে পারে যে আলাদা ভাবে ট্যামপ্লেট ব্যবহার করাটা আরো বেশি ঝামেলার, অথবা অপয়োজনীয়! হতাশ হবেননা... টেমপ্লেট এর কার্যকারিতা এবং সৌন্দর্য আস্তে আস্তে আমাদের সামনে উন্মোচিত হবে !!

“ও আর এম” পরিচিতি

আমাদেরকে প্রায়ই ডাটাবেইজ নিয়ে কাজ করতে হয়, ডাটাবেইজে টেবিল তৈরি, কলাম তৈরি, ডাটা ইনসার্ট, রিড, আপডেট বা ডিলেট করতে হয়! এ সকল কাজগুলো সাধারণত সিকুয়েল (SQL) ব্যবহার করে করতে হয়।

কিন্তু সরাসরি ‘র’ সিকুয়েল দিয়ে ডাটাবেইজ অপারেশনগুলো করতে আসলে অনেক কোড লিখতে হয়, এবং সিকুয়েল খুব সুবিধাজনক বা সুন্দর কোন ল্যঙ্গুয়েজও নয় যে সেটা দিয়ে কোড করতে খুব ভালো লাগবে! (সবাইকে অবশ্য এ কাতারে ফেলা যাবেনা!)

এই বিরক্তিকর ‘র’ সিকুয়েল লেখা থেকে মুক্তি দিতে ও আর এম (O,R,M) বা অবজেক্ট রিলেশনাল ম্যাপিং (Object Relational Mapping) এর আবির্ভাব! ওআরএম এর মাধ্যমে আপনি অবজেক্ট অরিয়েন্টেড স্টাইলে কোড লিখেই ডাটা বেইজের সকল কাজ করতে পারবেন, ডাটাবেইজে ‘র’ সিকুয়েল তৈরি করার কাজটা ওআরএম ই করে দিবে, আপনার কোন সিকুয়েল কোড লিখার দরকার হবেনা।

ওআরএম কে আপনি একটা কনভার্টার হিসেবে কল্পনা করতে পারেন, যেটা আপনার পাইথন কোডগুলো সিকুয়েল কোডে রূপান্তর করে দিবে, এমনকি ভিন্ন ভিন্ন ‘রিলেশনাল ডাটাবেইজ ম্যানেজমেন্ট সিস্টেম’ (RDBMS) এর জন্য সেগুলোর প্রয়োজনীয় স্টাইলে সিকুয়েল লিখতে পারবে!

ডাটাবেইজ কনফিগারেশন

জ্যাঙ্গো মূলত ডাটাবেজ ড্রিভেন ফ্রেমওয়ার্ক। অর্থাৎ জ্যাঙ্গো ব্যবহার করতে হলে আপনাকে ডাটাবেইজ ব্যবহার করতে হবে, এবং জ্যাঙ্গোর সাথে ডাটাবেইজ কানেক্ট করে নিতে হবে। প্রাথমিক ভাবে জ্যাঙ্গো সিকুয়েল লাইট (SQLite) ডাটাবেইজ ব্যবহার করে, এটা লাইটওয়েট এবং কোন ধরনের সেটিংস দরকার হয়না, এবং এটা পাইথনের সাথে ডিফল্ট ডাটাবেই থাকে।

তবে আপনি যদি অন্য কোন ডাটাবেইজ যেমন মাই সিকুয়েল (MySQL), পোস্টগ্রে সিকুয়েল (PostgreSQL) ইত্যাদি ব্যবহার করতে চান তাহলে সেগুলোকে ঠিক মত কনফিগার করে নিতে হবে।

জ্যাঙ্গো প্রোজেক্ট এর সেটিংস ফাইল ওপেন করুনঃ (আমাদের প্রোজেক্টে ফাইলটা হল myproject/settings.py) সেখানে DATABASES নামে একটা ডিকশনারি দেখতে পাবেন, যেটা দেখতে এরকমঃ

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

এখানে ডিফল্ট ডাটাবেইজ হিসেবে sqlite3 ব্যবহার করা হয়েছে। 'ENGINE' হল ডাটাবেইজ ড্রাইভারের নাম। এবং 'NAME' হল ডাটাবেইজ এর নাম। (SQLite এর ক্ষেত্রে ফাইলের নাম, কেননা sqlite ডাটাবেইজ সিঙ্গেল ফাইলের ভিতর তৈরি হয়)

আপনি যদি অন্য কোন ডাটাবেজ ব্যবহার করতে চান তাহলে সেগুলোর কনফিগারেশন এখানে দিতে হবে, যেমন ধরি আমরা sqlite3 এর পরিবর্তে PostgreSQL ব্যবহার করব, তাহলে উক্ত ডিকশনারিটাকে এভাবে লিখতামঃ

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql', # ডাটাবেইজ ড্রাইভার
        'NAME': 'mydatabasename', # ডাটাবেইজের নাম
        'USER': 'database_username', #ডাটাবেইজ ইউজারনেম
        'PASSWORD': 'D@T@B#S3_password', #ডাটাবেইজ পাসওয়ার্ড
        'HOST': '', #ডাটাবেইজ হোস্ট, নোকানহোস্ট ব্যবহার করতে চাইলে খালি রাখুন
        'PORT': '', # ডাটাবেইজ পোর্ট, ডিফল্ট ব্যবহার করতে চাইলে খালি রাখুন
    }
}
```

উল্লেখিত ডিকশনারিতে আপনার ডাটাবেইজ অনুযায়ী 'কি/ভ্যালু' সেট করে দিতে হবে।

ডাটাবেইজ তৈরি হয়ে গেলে আমাদের কাজ হল মডেল তৈরি করা, মডেল এর কথা মনে আছেতো? ডাটাবেইজের সঙ্গে আমাদের যোগাযোগ হবে মডেলের মাধ্যমে...

মডেল তৈরি

আমাদের লক্ষ্য হল, কেউ যদি আমাদের ওয়েবসাইটে আসে তাহলে তাকে শুধু Hello World! না দেখিয়ে এর সাথে একটা মেসেজও দেখাবো। আমরা কয়েকটি টেক্সট মেসেজ ডাটাবেইজে সেভ করে রাখব, আর ভিজিটরকে র‍্যান্ডম ভাবে একেক সময় একেকটা মেসেজ দেখাব। যেমন Hello World! Today is Beautiful Day! অথবা Hello World! Nice To Meet You!

এই মেসেজগুলো লিখতে হলে আমাদেরকে ডাটাবেইজে একটা টেবিল তৈরি করতে হবে, টেবিল তৈরি করতে প্রথমে আপনার অ্যাপ (myapp) ফোল্ডারের models.py ফাইলটি কোন টেক্সট এডিটরে ওপেন করুন। সেখানে এরকম দুটি লাইন দেখা যাবে।

```
from django.db import models

# Create your models here.
```

প্রথম লাইনে models মডিউল ইম্পোর্ট করা হয়েছে। ওআরএম চ্যাপ্টারে আমরা বলেছি যে ডাটাবেইজের টেবিল/কলাম তৈরি করব অবজেক্ট অরিয়েন্টেড কোড দিয়ে, অর্থাৎ ক্লাস লিখে। আমাদের একেকটা টেবিল হবে একেকটা ক্লাস, ক্লাসের এট্রিবিউট গুলো হবে টেবিলের কলাম! আর হ্যাঁ, আমাদের সকল মডেল ক্লাসগুলোই উপরে ইম্পোর্ট করা models.Model এর সাবক্লাস হবে।

ব্যপারটা কঠিন লাগছে? আসুন কোডে চলে যাই, আমাদের সাইটের মেসেজ গুলো রাখার জন্য প্রথমে আমরা Message নামে একটা ক্লাস তৈরি করব, models.py ফাইলে (# Create your models here. কमेंটটার পর) লিখুনঃ

```
class Message(models.Model):
```

আমরা Message নামে ক্লাস তৈরি করলাম, এটাকেই আমাদের ডাটাবেইজ টেবিল হিসেবে ধরে নিতে পারেন, আমাদের ক্লাসটি models.Model এর সাবক্লাস করলাম, এতে আমাদের ক্লাসে প্রয়োজনীয় কিছু এট্রিবিউট/মেথড যুক্ত হবে।

এর পর লিখুনঃ

```
text = models.TextField()
```

আমরা text নামে একটা এট্রিবিউট লিখলাম, যার ড্যালা হল models.TextField অবজেক্ট! TextField দ্বারা বোঝাচ্ছে যে ডাটাবেইজে এই কলামের ফিল্ড/রো গুলো টেক্সট টাইপের হবে, পাইথনের স্ট্রিং গুলোই ডাটাবেইজ টেক্সট হিসেবে থাকবে।

মাথা গরম করার দরকার নেই, এটা ওআরএম এর সুবিধা/বৈশিষ্ট্য! উপরের কোডটির বাংলা অনুবাদটা মোটামোটি এরকমঃ

‘ওহে ডাটাবেইজ, তুমি Message নামক টেবিলে text নামে একটা কলাম তৈরি কর, যে কলামটাতে ব্লগ/ফিল্ড হিসেবে টেক্সট/স্ট্রিং টাইপ ড্যালা থাকবে’

এর পর আমাদেরকে **str()** ম্যাজিক মেথড ওভাররাইট করতে হবে:

```
def __str__(self):  
    return self.text
```

বুঝতেই পারছেন, এই মেথডের কারনে এখন Message ক্লাসের অবজেক্টগুলোকে print করলে সেই অবজেক্টের text ফিল্ডের ড্যালা দেখা যাবে!

আমাদের Message মডেল তৈরি হয়ে গেল। একনজরে models.py ফাইল:

```
from django.db import models  
  
# Create your models here.  
class Message(models.Model):  
    text = models.TextField()  
  
    def __str__(self):  
        return self.text
```

মডেল তৈরি হল, ডাটাবেইজ কিন্তু এখনো তৈরি হয়নি... এই মডেল অনুযায়ী ডাটাবেইজ তৈরি করতে আমাদেরকে manage.py ফাইলের দুটি কমান্ড এর সাথে পরিচিত হতে হবে।

makemigrations

প্রতিবারই আপনি কোন মডেল তৈরি করলে অথবা মডেলের কোন ফিল্ড (এট্রিবিউট) পরিবর্তন করলে আপনাকে কমান্ড প্রম্পট বা টার্মিনালে

```
python manage.py makemigrations
```

কমান্ডটি দিতে হবে। এতে করে আপনার অ্যাপ ফোল্ডারে (আমাদের myapp ফোল্ডারে) থাকা migrations ফোল্ডারে একটি মাইগ্রেশন নাম্বার যুক্ত পাইথন ফাইল তৈরি হবে। সেই ফাইলের ভিতরে যে কোডগুলো থাকবে সেটা আসলে আপনার ডাটাবেইজ এর সিকুয়েল কোড কেমন হবে তারই একটা প্রতিকৃতি! এর উপর ভিত্তি করেই আপনার ডাটাবেইজ টেবিল/কলাম/ডাটা তৈরি হবে। আপনি সিকুয়েল ডালা পারলে এখানে আপনার চাহিদা অনুযায়ী পরিবর্তন করতে পারবেন। মাইগ্রেশনের উল্লেখযোগ্য আরেকটা সুবিধা হল, প্রতিবার মডেল আপডেট করে **makemigrations** কমান্ড দিলে প্রতিবারই migrations ফোল্ডারে নতুন একটা ফাইল তৈরি হয়, যেটা ডাশন কন্ট্রোল করতে খুবই কাজের, অর্থাৎ আপনি যদি ভুলে ডাটাবেইজ আপডেট করেন তাহলে ব্যাক করে আগের অবস্থায় ফিরে যেতে পারবেন।

makemigrations কমান্ড দেয়ার পরও কিন্তু আপনার ডাটাবেইজ তৈরি হয়নি, আসল ডাটাবেইজ তৈরি হবে **migrate** কমান্ড দিলে।

migrate

মাইগ্রেশন তৈরি করার পর তা অনুযায়ী ডাটাবেইজ তৈরি করতে

```
python manage.py migrate
```

কমান্ডটি দিতে হবে। এটাই আপনার ডাটাবেইজ তৈরি/আপডেট করে দিবে।

লক্ষ্য করুন, প্রতিবার models.py তে মডেল যুক্ত বা আপডেট করলে ডাটাবেইজে সেটা ইমপ্লিমেন্ট করতে টার্মিনালে `python manage.py makemigrations` এবং `python manage.py migrate` কমান্ড দুটি দিতে হবে।

`makemigrations` কমান্ড কিন্তু ডাটাবেইজ তৈরি/আপডেট করেনা, শুধু একটা মাইগ্রেশন ফাইল তৈরি করে। এরপর `migrate` কমান্ডটা সেই মাইগ্রেশনের উপর ভিত্তি করে আসল ডাটাবেইজ তৈরি/আপডেট করে।

আমাদের মডেল তৈরি হল এবং ডাটাবেইজও রেডি হয়ে গেলে... এখন ডাটাবেইজে ডাটা রাখতে হবে... এখন পরিচিত হওয়ার সময় এসেছে জ্যাক্সো শেল এর সাথে... জ্যাক্সো শেল... নাম শুনে কি ভয় লাগছে !!??

জ্যাস্কো শেল পরিচিতি

জ্যাস্কোর `manage.py` টুল এর আরেকটি প্রয়োজনীয় কমান্ড হচ্ছে `shell`। এটা আসলে সাধারণ একটা পাইথন ইন্টারপ্রেটিড শেল, কিছু অতিরিক্ত সেটিংস সহ! আমাদেরকে প্রোজেক্টের অনেক ফাইলই ইন্টারপ্রেটিড ভাবে এক্সিকিউট করতে হয়, মানে কমান্ড প্রম্পট বা টার্মিনালে পাইথন শেল ওপেন করে সে ফাইলে কাজ করতে হয়। কিন্তু জ্যাস্কো প্রোজেক্টে কাজ করতে গেলে কিছু অতিরিক্ত সেটিংস লাগে যা সাধারণ পাইথন শেলে ম্যানুয়ালি করার দরকার পরে। বিগিনারদের জন্য একটু কনফিউশনের বিষয়।

একারণে `manage.py` টুলসে `shell` নামে অতিরিক্ত সেটিংস সহ পাইথন ইন্টারপ্রেটিড শেল দেয়া থাকে, প্রোজেক্টের ফাইলগুলো ব্যবহার করতে আমরা পাইথন শেলের বদলে জ্যাস্কোর এই শেল ব্যবহার করব।

শেল ওপেন করতে কমান্ড প্রম্পট বা টার্মিনালে লিখুনঃ

```
python manage.py shell
```

এর পর পাইথন শেলের মতই এটা কাজ করবে।

ডাটাবেইজ কুয়েরি, জ্যাক্সো ওআরএম এর কুয়েরিসেট এপিআই!

মডেল তৈরি (এবং ডাটাবেইজ তৈরি) করার পর আমাদের কাজ হল সেখানে ডাটা ইনসার্ট করা, সেখানের ডাটা রিড, আপডেট বা ডিলেট করা। জ্যাক্সো ওআরএম আমাদেরকে এই কাজগুলি করার জন্য সুন্দর একটা এপিআই দেয়, কুয়েরিসেট! কোডে চলে যাই, প্রথমেই কমান্ড প্রম্পট বা টার্মিনালে জ্যাক্সো শেল ওপেন করুনঃ

```
python manage.py shell
```

পাইথন ইন্টারপ্রেটার শেল চালু হবে, সেখানে `models.py` তে তৈরি করা আমাদের `Message` মডেল (ক্লাস) ইম্পোর্ট করুনঃ

```
from myapp.models import Message
```

আমরা `Message` মডেল দিয়ে ডাটাবেইজ তৈরি করে ফেলেছি, এখন এই মডেল ব্যবহার করেই সেই ডাটাবেইজ এর অপারেশনগুলো করব। আমরা জানি যে এই মডেলটির প্যারেন্ট ক্লাস হল জ্যাক্সোর `models.Model` ক্লাস। এই মডেল ক্লাসটিই ডাটাবেইজ অপারেশনগুলো করার জন্য আমাদেরকে `objects` নামে একটা ম্যানেজার (এপিআই) দেয়। আমরা সেটা আমাদের `Message` ক্লাসে ব্যবহার করব।

প্রথমেই দেখা যাক ডাটাবেইজের সব ডাটা রিড করা যায় কিভাবে, লিখুনঃ

```
Message.objects.all()
```

`Message` ক্লাসের `Obojects` ম্যানেজার এর `all()` মেথডকে কল করলাম, এটা আমাদের ডাটাবেইজ কুয়েরি করে সকল ডাটা কুয়েরিসেট হিসেবে রিটার্ন করবে! উপরের কোড লিখে এন্টার করলে আপনি পরের লাইনে এরকম কিছু দেখতে পাবেনঃ

```
<QuerySet []>
```

কুয়েরিসেট অবজেক্ট, একটা খালি লিস্ট! খালি থাকার কারণ হল আমরা এখনো ডাটাবেইজে কোন ডাটা রাখিনি। আসুন আগে কিছু ডাটা (মেসেজ আরকি) রাখি। ডাটা ক্রিয়েট করতে এই কমান্ডঃ

```
Message.objects.create(text="Hi there! How are You!")
Message.objects.create(text="Have a nice day boy!")
Message.objects.create(text="Are you Crazy !")
```

আমরা তিনটি মেসেজ বা ডাটা তৈরি করলাম, `objects` ম্যানেজারের `create` ফাংশন কল করে ডাটা ক্রিয়েট করেছি, ফাংশনের ভেতর কিওয়ার্ড আর্গুমেন্ট হিসেবে ডাটাবেইজের কলামের নাম এবং ফিল্ড ভ্যালু (সেই কলামের ডাটা যেটা হবে) দিয়েছি। এখন আমাদের ডাটাবেইজের `Message` টেবিলের `text` কলামে তিনটা ফিল্ড/রো যুক্ত হয়েছে, তিনটা মেসেজ!

এবার যদি আমরা `Message.objects.all()` কল করি তাহলে এরকম দেখবঃ

```
<QuerySet [<Message: Hi there! how are you!>, <Message: Have a nice day boy!>, <Message: Are you Crazy!>]>
```

কুয়েরিসেট লিস্ট, ডিতরে তিনটি আইটেম, আমাদের তিনটি মেসেজ!

`objects` এর `all()` ফাংশনটি সকল আইটেম কুয়েরি করে বের করে। আমরা যদি নির্দিষ্ট কোন একটা আইটেম বের করতে চাই তাহলে `get()` ফাংশন ব্যবহার করতে পারিঃ

```
Message.objects.get(id=1)
```

আমাদের মডেলে যদিও আমরা শুধু `text` এটিবিউট বা কলাম রেখেছিলাম, কিন্তু জ্যাকসো আমাদের জন্য ‘প্রাইমারি কি’ হিসেবে অতিরিক্ত আরেকটা এটিবিউট যোগ করে দিয়েছে `id` নামে। এটা করার কারন হল আমরা কোন ফিল্ডকে প্রাইমারি কি হিসেবে সেট করিনি, কিন্তু জ্যাকসো ওআরএম প্রতিটি মডেলের মধ্যেই একটা ‘প্রাইমারি কি’ আশা করে।

যদিও অতিরিক্ত কলামটির নাম `id` তবে সেটাকে `pk` নামেও ডাকা যাবে। `pk = Primary Key` !

`get()` ফাংশনটিতে আমরা `id=1` দিয়ে বলেছি যে সে যেন আমাদেরকে ঐ ফিল্ডটা রিটার্ন করে যার `id` এর ভ্যালু হল ১।

`Message.objects.get(id=1)` লিখলে রিটার্ন আসবেঃ

এবার কোন কুয়েরিসেট/লিস্ট আসেনি, বরং একটাই মাত্র অবজেক্ট! আমরা অবজেক্টটি থেকে আমাদের প্রয়োজন অনুযায়ী ফিল্ড ভ্যালু বের করে নিতে পারিঃ

```
>>> mytext = Message.objects.get(id=1)
<Message: Hi there! how are you!>
>>> mytext.text
"Hi there! How are you!"
>>> mytext.id
1
>>> mytext.pk
1
```

আমাদের মডেলে যদি আরো ফিল্ড থাকত তাহলে আমরা এভাবে সেগুলোর ভ্যালুও বের করতে পারতাম।

`objects` ম্যানেজারের আরেকটা বহুল ব্যবহৃত মেথড হল `filter()`, অনেক সময়ই আমাদেরকে নির্দিষ্ট কোন কন্ডিশনের উপর ভিত্তি করে ডাটাবেইজের ডাটা রিড করতে হয়। ফিল্টার মেথড আমাদেরকে সেই কাজটি করতে সাহায্য করেঃ

```
>>> Message.objects.filter(id=1)
<QuerySet [<Message: Hi there! How are You!>]>
```

ফিল্টারের মাধ্যমে আমরা বলে দিয়েছি যে ঐ ‘সকল’ অবজেক্ট গুলো বের কর যার আইডি হল ১, ফিল্টারটি একটা কুয়েরিসেট রিটার্ন করেছে এবং যেহেতু ডাটাবেইজে শুধুমাত্র একটি অবজেক্টের id এর মান 1 তাই এই কুয়েরিসেটের ভিতর মাত্র একটি ভ্যালু আছে!

এমন যদি হয় যে ফিল্টারের ভিতরের কন্ডিশনটা একাধিক অবজেক্টের সাথে ম্যাচ করে তাহলে সে সেই সবগুলো অবজেক্টই রিটার্ন করবে:

```
>>> Message.objects.filter(text__icontains='you')
<QuerySet [<Message: Hi there! how are you!>, <Message: Are you Crazy!>]>
```

উপরোক্ত কোডে ফিল্টার করা হচ্ছে ঐ ‘সকল’ অবজেক্টকে যেগুলোর টেক্সট ফিল্ডে ‘you’ স্ট্রিং টি আছে। এটা দুটি অবজেক্টের একটি কুয়েরিসেট রিটার্ন করেছে, কেননা দুটি অবজেক্টের টেক্সট ফিল্ডে ‘you’ স্ট্রিং টি আছে!

মডেলের objects মেথড (কোন মডেলের objects মেথডকে আসলে ‘মডেল ম্যানেজার’ বলে) এর মাধ্যমে আমরা যেমন অবজেক্ট রিড করতে পারি, তেমন কোন অবজেক্ট ক্রিয়েট, আপডেট এবং ডিলেটও করতে পারি! এভাবেঃ

```
>>> Message.objects.all().count()
3
```

প্রথমে count() মেথড দিয়ে চেক করে দেখলাম আমাদের মেসেজ মডেলে/টেবিলে কয়টি অবজেক্ট/ফিল্ড আছে! তিনটি আছে!

```
>>> new_message = Message.objects.create(text='today is Friday')
```

Objects ম্যানেজারের create() মেথড দিয়ে নতুন অবজেক্ট ক্রিয়েট করলাম, আর্গুমেন্ট হিসেবে text ফিল্ড এর ভ্যালু দিয়ে দিলাম! এবং অবজেক্টটিকে new_message ভেরিয়েবলে রাখলাম!

```
>>> new_message
<Message: today is Friday>
```

নতুন অবজেক্ট কল করে দেখলাম ক্রিয়েট হয়েছে কিনা! হয়েছে।

```
>>> Message.objects.all().count()
4
```

আবার count() মেথড কল করে চেক করে দেখলাম টেবিলে এখন কয়টি ফিল্ড/অবজেক্ট আছে! এখন আছে চারটি! নতুন অবজেক্ট সহ!

```
>>> new_message.text
'today is Friday'
>>> new_message.id
4
```

নতুন অবজেক্টটির text ফিল্ড এর মান আমরা অবজেক্ট ক্রিয়েট করার সময় দিয়ে দিয়েছিলাম, এবং id ফিল্ডটা প্রাইমারি কি হিসেবে জ্যাস্গো অটোমেটিক তৈরি করে দিয়েছে!

```
>>> new_message.text = 'No, today is Sunday')
>>> new_message.save()
```

আমরা পাইথনের ডেরিয়েবলে ভ্যালু এসাইন করার মতই text ফিল্ড এ নতুন ভ্যালু এসাইন করলাম। এবং এটা ডাটাবেইজে সেভ করার জন্য save() মেথড কল করলাম।

```
>>> new_message.text
'No, today is Sunday'
```

এখন আমার সেই অবজেক্ট এর text ফিল্ড কল করে দেখলাম তার ভ্যালুটা কি! সেখানে আপডেটেড ভ্যালুটা দেখা যাচ্ছে। তার মানে আমাদের ফিল্ড/অবজেক্টটা আপডেট হয়ে গেছে।

```
>>> new_message.delete()
(1, {'myapp.Message': 1})
```

উপ্স! আমরা নতুন ক্রিয়েট করা অবজেক্টটা ডিলেট করে দিলাম!!

```
>>> Message.objects.all().count()
3
```

অতঃপর আবার চেক করে দেখলাম আমাদের টেবিলে চারটা ফিল্ড থেকে একটা ডিলেট হয়ে এখন তিনটা ফিল্ড আছে!

জ্যাস্গো ওআরএম দিয়ে চাইলে আমরা আরো অনেক ধরনের কমপ্লেক্স কুয়েরি করতে পারি, আস্তে আস্তে সব শেখা হবে ইনশাল্লাহ।

মনে রাখবেনঃ জ্যাস্গো ওআরএম এ প্রতিটা মডেল হল ডাটাবেইজের একেকটা টেবিল, মডেলের ফিল্ডগুলো হল ডাটাবেইজের টেবিলের কলাম, মডেলের প্রতিটা ইন্সট্যান্স (মডেল থেকে তৈরি করা প্রতিটা অবজেক্ট) হল টেবিলের ফিল্ড বা ডাটা বা 'রো' !

ভিউ থেকে ডাটাবেইজ এর ডাটা রিড করা

আমরা মডেল তৈরি করেছি, সেই মডেলে কিছু ডাটা/ফিল্ড তৈরি করেছি, এখন আমরা সেই ডাটাগুলো ভিউ এর মাধ্যমে প্রকাশ করব, যাতে কেউ আমাদের ওয়েবসাইটে () গেলে সেই ডাটাগুলো দখতে পায়।

আমরা আগের চ্যাপ্টারে দেখেছি কিভাবে জ্যান্সো শেলের মাধ্যমে মডেল এর ডাটা রিড করা যায়, ভিউ তে আমরা একই ভাবে কাজটা করব। আমাদের myapp এর ভিতরে থাকা views.py ফাইলটা টেক্সট এডিটরে ওপেন করুন, সেটার চেহারা মোটামোটি এরকম থাকার কথাঃ

```
from django.shortcuts import render, HttpResponseRedirect

# Create your views here.
def index(request):
    return render(request, 'index.html')
```

উক্ত ভিউটা রেসপন্স হিসেবে শুধুমাত্র 'index.html' ফাইলটা রেন্ডার করে পাঠিয়ে দিচ্ছে! অন্য কোন কাজ করছেনা। এটাকে আমাদের মন মত কাজ করাতে প্রথমে models.py থেকে Message মডেলটা ইম্পোর্ট করুনঃ

from models import Message তারপর index ভিউ ফাংশন এর ভিতর মডেল ম্যানেজার ব্যবহার করে Message এর সবগুলো ফিল্ড একটা ভ্যারিয়েবলে রাখুনঃ

```
all_messages = Message.objects.all()
```

সবগুলো মেসেজ এখন all_messages ভেরিয়েবলে আছে।

সর্বশেষ যেটা করতে হবে, এই ভেরিয়েবলটাকে 'index.html' টেমপ্লেট এর ভিতর পাঠিয়ে দিতে হবে! এর ফলে আমরা টেমপ্লেট থেকেই all_messages এর ভ্যালু এক্সেস করতে পারব।

এটা করার জন্য render() ফাংশন এর তৃতীয় প্যারামিটার হিসেবে একটা ডিকশনারি পাস করতে হবে, যার 'কী' হবে একটা স্ট্রিং, এবং ভ্যালু হবে all_message ভেরিয়েবল।

```
return render(request, 'index.html', {'messages': all_messages})
```

আমরা ডিকশনারির 'কী' হিসেবে 'messages' স্ট্রিং দিয়েছি। টেমপ্লেটে আমরা all_messages ভেরিয়েবল এক্সেস করার জন্য এই messages 'কী' টাকেই ব্যবহার করব!

আমাদের সম্পূর্ণ views.py এর কোড দেখতে এরকম হবেঃ

```
from django.shortcuts import render, HttpResponseRedirect

from models import Message

# Create your views here.
def index(request):
    all_messages = Message.objects.all()
    return render(request, 'index.html', {'messages': all_messages})
```

ডিউ এর কাজ শেষ! এখন টেমপ্লেট এর কাজ শুরু করতে হবে, মেসেজ এর সবগুলো অবজেক্ট সহ যে ডেরিয়েবলটা টেমপ্লেটে পাঠানো হল তা সুন্দর ভাবে শো করার জন্য আমাদেরকে আরেকটু গভীর ভাবে জ্যাকসো টেমপ্লেট ল্যান্সুয়েজ এর সাথে পরিচিত হতে হবে... চলুন তাহলে!

টেমপ্লেট ট্যাগ এবং ফিল্টার

জ্যাক্সো টেমপ্লেট ল্যাঙ্গুয়েজ (DTL) একদম এইচটিএমএল এর মতই! তবে তাতে অতিরিক্ত কিছু সুবিধা পাওয়া যায়! উল্লেখযোগ্য সুবিধাগুলো হলঃ

১) টেমপ্লেট ইনহেরিটেন্স! পাইথনে আমরা যেভাবে এক ক্লাস থেকে আরেক ক্লাস কে ইনহেরিট করি, টেমপ্লেট ল্যাঙ্গুয়েজেও সেরকমটা করা যায়, একটা বেজ টেমপ্লেট তৈরি করে সেটা যেকোনো চাইল্ড টেমপ্লেটে ইনহেরিট করে নেয়া যায়!

২) টেমপ্লেট ইনক্লুডিং! যদি টেমপ্লেট বড় হয়ে যায় সেটাকে আমরা বিভিন্ন ফাইলে বিভক্ত করে যেকোন একফাইলে ইনক্লুড করে নিতে পারি! উদাহরণস্বরূপ, আমরা টেমপ্লেট এর হেডার, ফুটার, সাইডবার, ন্যাভম্যানু বার, স্লাইডার ইত্যাদি আলাদা আলাদা ফাইলে লিখতে পারি এবং সেগুলোকে হোম পেইজ বা অন্য কোন পেইজে ইনক্লুড করে দিতে পারি। এতে বারবার কোড করা থেকে মুক্তি পাওয়া যাবে এবং কোড ওয়েল অর্গানাইজড থাকবে।

টেমপ্লেট ইনক্লুড করা এবং টেমপ্লেট ইনহেরিট করা নিয়ে প্রথম প্রথম কনফিউশন লাগতে পারে, দুটোকে প্রায় এক মনে হলেও তাদের মধ্যে কিন্তু অনেক পার্থক্য আছে!

৩) টেমপ্লেট ট্যাগ ব্যবহার! ট্যাগ এর মাধ্যমে আমরা পাইথনের কমন কিছু ফাংশনালিটি সরাসরি টেমপ্লেটেই ব্যবহার করতে পারি! যেমন ইফ/এলস কন্ডিশন চেক করা, ফরলুপ চালানো, ফাংশন কল করা, ভেরিয়েবল কল করা ইত্যাদি বিষয়গুলো ট্যাগ দিয়ে করা যায়!

৪) টেমপ্লেট ফিল্টার! এটাকে পাইথনের ফাংশন বা মেথডের সাথে তুলনা করা যায়, ফিল্টার মূলত টেমপ্লেট ট্যাগের উপর ব্যবহার করা হয় সেই ট্যাগের ভ্যালুতে নির্দিষ্ট কিছু পরিবর্তন করার জন্য। যেমন কোনো স্ট্রিংকে আপারকেস করা, ডেটটাইমকে হিউম্যান রিডেবল করা, কোন টেমপ্লেট ট্যাগ এর ডিফল্ট মান দেয়া ইত্যাদি কাজ গুলো ফিল্টার দিয়ে করা যায়।

আগের চ্যাপ্টারে ভিউ থেকে আমরা একটি ভেরিয়েবলকে টেমপ্লেটে পাস করেছিলাম যেটা messages 'কী' এর মাধ্যমে এক্সেস করা যাবে, সে ভেরিয়েবল এর ভ্যালুগুলো এখন আমরা টেমপ্লেট ট্যাগ এর মাধ্যমে শো করব।

Myapp এপ এর ভিতরের templates ফোল্ডারে থাকা index.html ফাইলটি ওপেন করুন (ডিরেক্টরিঃ

myproject\myapp\templates), সেটার কোডগুলো এরকম ছিলঃ

```
<!DOCTYPE html>
<html>
  <head>
    <title>My view</title>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

এখানে <h1>Hello World!</h1> এর নিচে লিখুনঃ

```
{% for message in messages %}
    <p> {{ message.text }} </p>
{% endfor %}
```

খেয়াল করে দেখুন, এটা কিন্তু একটা ফরলুপ! পাইথনের ফরলুপের মতই, শুধু দুটি পার্থক্যঃ ১) ফরলুপটিকে আমরা `{% %}` এরকম বন্ধনির ভিতর রেখেছি! এটা হল ট্যামপ্লেট ট্যাগে এর সিন্ট্যাক্স, ফরলুপ, ফাংশন কল, কন্ডিশনাল চেক ইত্যাদি কাজগুলো করতে হলে সেগুলোকে এই বন্ধনির ভিতর রাখতে হবে। ২) তিন নাম্বার লাইনে দেখুন `{% endfor %}` ট্যাগ দিয়ে আমরা ফরলুপ ব্লক এর শেষ বুঝিয়েছি, পাইথনে যেমন ইন্ডেন্টেশন ব্লক দিয়ে ফরলুপ বা অন্যান্য কোড ব্লক বোঝানো হয়, ট্যামপ্লেট ল্যাঙ্গুয়েজে যেহেতু ইন্ডেন্টেশন বাধ্যতামূলক নয় তাই এখানে স্পষ্ট ভাবে কোডব্লক এর শেষ উল্লেখ করে দিতে হবে!

মধ্যবর্তী লাইনটাতে এইচটিএমএল এর `<p> </p>` ট্যাগে এর ভিতর `{{ message.text }}` লিখে আমরা `message` এর `text` ফিল্ডকে কল করছি। এটা ট্যামপ্লেট ট্যাগ এর আরেক সিন্ট্যাক্স, কোন ধরনের ভেরিয়েবলকে কল করতে চাইলে `{{ }}` এরকম বন্ধনির ভিতর তা লিখতে হবে!

তো আমাদের `index.html` ফাইলের সম্পূর্ণ কোডটি এখন এরকমঃ

```
<!DOCTYPE html>
<html>
    <head>
        <title>My view</title>
    </head>
    <body>
        <h1>Hello World!</h1>

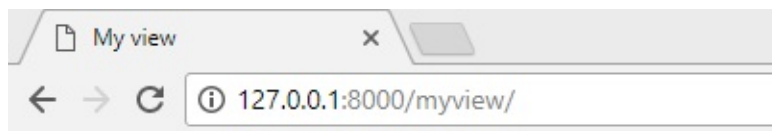
        {% for message in messages %}
            <p> {{message.text}} </p>
        {% endfor %}

    </body>
</html>
```

এখন কমান্ড লাইন থেকে আবার ডেভেলপমেন্ট সার্ভার চালু করুন এবং ব্রাউজারে এই ঠিকানায় যানঃ

```
http://127.0.0.1:8000/myview/
```

দেখবেন আমাদের তৈরি করা তিনটি মেসেজ সুন্দর ভাবে শো করছেঃ



Hello World!

Hi there! How are You!

Have a nice day boy!

Hey! Are you Crazy !

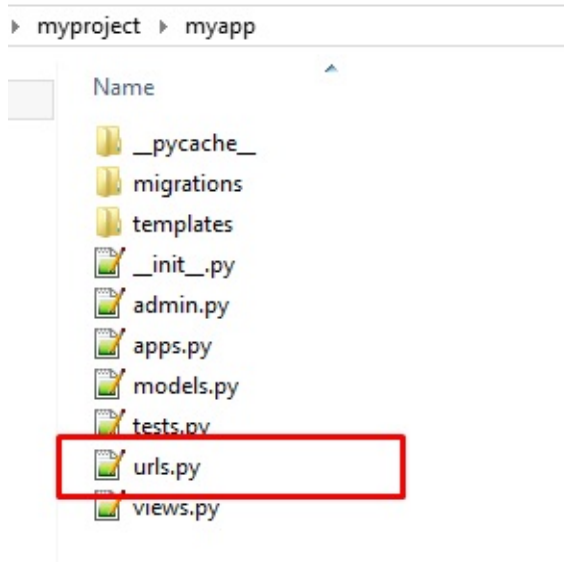
এক নজরেঃ যখনই আমরা ব্রাউজারে `http://127.0.0.1:8000/myview/` ইউআরএল লিখে এন্টার চাপছি তখন ব্রাউজার আমাদের সার্ভারে একটা রিকুয়েস্ট পাঠাচ্ছে, তারপর আমাদের সার্ভারে থাকা জ্যাক্সো সেই ইউআরএল টাকে `urls.py` ফাইলে থাকা ইউআরএল গুলোর সাথে মিলিয়ে দেখছে যে এই ইউআরএল এর সাথে সংযুক্ত কোন ভিউ আছে কিনা! যখন সে এটার সাথে `index()` ভিউ এর ম্যাচ পেল সাথে সাথে `index()` ভিউকে কল করল। `index()` ভিউ প্রথমে `Message` মডেলের অবজেক্টগুলোকে নিয়ে আসল এবং সেটাকে ট্যামপ্লেট এর মাধ্যমে রেন্ডার করতে দিল, ট্যামপ্লেটে অবজেক্টগুলো ঠিক মত রেন্ডার হয়ে গেলে `index()` ভিউটি সেটাকে রেসপন্স হিসেবে ব্রাউজারে পাঠিয়ে দিল আর আমরা ব্রাউজার আমাদের কাঙ্ক্ষিত ফলাফল দেখতে পেলাম!

ইউআরএল কনফিগারেশন ইন ডেপথ

আমাদের `myapp` এর ডিউগুলো এক্সেস করতে যে ইউআরএল ডিজিট করতে হয় সেগুলো আমরা `myproject` প্রোজেক্ট ফোল্ডার এর ডিতরের `urls.py` তে রেখেছি! এবং আমরা আগেই জেনেছি যে জ্যাক্সোতে প্রোজেক্টগুলো ছোট ছোট এপ এ ভাগ করা থাকে, আমাদের প্রোজেক্টে যদিও এখন পর্যন্ত একটা মাত্র এপ আছে, কিন্তু প্রোজেক্ট বড় হলে সেখানে অনেকগুলো এপ তৈরি করতে হয়।

এখন সবগুলো এপস এর ইউআরএল গুলো যদি সেই মেইন প্রোজেক্ট ফোল্ডারের `urls.py` তে রাখি তাহলে ফাইলটা অনেক বড় এবং বিদ্যুটে হয়ে যাবে! এটাই কি ভাল না যে আমরা প্রতিটা এপ এর ইউআরএল কনফিগারেশন সেই এপ ফোল্ডারের এর ডিতরেই রাখব!? তাহলে কোডগুলো আরো অর্গানাইজড এবং রিইউজএবল হবে!

একাজটি করা খুবই সোজা, প্রথমে এপ ফোল্ডারের ডিতর `urls.py` নামে একটা ফাইল তৈরি করুনঃ



তারপর সেটা টেক্সট এডিটরে ওপেন করুন, এবং নিচের কোডগুলো লিখুনঃ

```
from django.conf.urls import url

from myapp.views import index

urlpatterns = [
    url(r'^$', index),
]
```

- প্রথমে আমরা `url` ফাংশনটি ইম্পোর্ট করেছি।
- তারপর আমাদের ডিউ ফাইল থেকে `index()` ডিউ ফাংশন ইম্পোর্ট করেছি।
- তারপর `urlpatterns` নামের একটা লিস্টে `url` ফাংশনটি কল করেছি, যার প্রথম প্যারামিটার হিসেবে আছে `r'^$',` বেগুলার এক্সপ্রেশন, যেটা আসলে ফাকা স্ট্রিং এর সাথে ম্যাচ করবে (অর্থাৎ ইউআরএল এ কোন স্ট্রিং বা ভ্যালু থাকলে সেটা ম্যাচ করবেনা)

এখন `myproject` প্রোজেক্ট ফোল্ডারে থাকা `urls.py` ফাইল ওপেন করুন, সেখানে এরকম কোড থাকবেঃ

```
# উপরের কমেন্টগুলো বাদ দেয়া হয়েছে
from django.conf.urls import url
from django.contrib import admin

from myapp.views import index

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^myview/', index),
]
```

এটাকে মডিফাই করে এরকম করুনঃ

```
from django.conf.urls import url, include
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^myview/', include('myapp.urls')),
]
```

- প্রথম লাইনে আমরা `url` এর সাথে `include` নামে একটা ফাংশন ইম্পোর্ট করেছি।
- `myapp.views` থেকে যে `index()` ডিউ ইম্পোর্ট করা ছিল সেটা রিমোভ করেছি, সেটা এখানে আর দরকার নেই।
- `urlpatterns` লিস্টের দ্বিতীয় আইটেমটিতে `url` ফাংশনে আমরা `include('myapp.urls')` দিয়ে এটা বুঝিয়েছি যে যখনই কোনো রিকুয়েস্টকৃত ইউআরএল এর প্রথম অংশ `r'^myview/'` এর সাথে ম্যাচ করবে তখন সে ইউআরএল এর বাকি অংশ ম্যাচ করার জন্য `'myapp.urls'` ফাইলটা এখানে ইনক্লুড করে নিবে!

বিষয়টা একটু কঠিন হয়ে গেল!?! আবার বলি, ধরুন আমরা ব্রাউজারে <http://127.0.0.1:8000/myview/> লিখে এন্টার চাপলাম!

- ব্রাউজার থেকে সার্ভারে রিকুয়েস্ট আসল, জ্যাকসো উক্ত ইউআরএল টি `myproject` ফোল্ডারের `urls.py` তে থাকা `urlpatterns` লিস্টের ইউআরএল গুলোর সাথে মিলিয়ে দেখলো কোনটা মিলে!,
- দ্বিতীয় প্যাটার্নটা উক্ত ইউআরএল এর প্রথম অংশ `myview/` এর সাথে মিলল,
- জ্যাকসো এখন উক্ত ইউআরএল এর `myview/` এর পরের অংশ মিলানোর জন্য `myapp.urls` এ থাকা `urlpatterns` লিস্টের সাথে ম্যাচ করা শুরু করল!,
- যেহেতু `myview/` এর পরের অংশ ফাঁকা বা সেখানে যেহেতু কিছুই নেই তাই সেটা `r'^$'` রেজেক্স এর সাথে ম্যাচ করল এবং তার সাথে থাকা `index` ডিউ কল হল।
- অতঃপর ডিউ তার কাজ শেষে রেসপন্স পাঠিয়ে দিল!

মনে রাখবেনঃ জ্যাঙ্গো ইউআরএল ম্যাচ করে ডোমেইন নেম বাদ দিয়েঃ `http://127.0.0.1:8000/myview/` এর ক্ষেত্রে শুধু `myview/` টুকু ম্যাচ করা হবে `http://127.0.0.1:8000/` অংশটা বাদ দেয়া হবে। রিকুয়েস্ট করা যেকোন ইউআরএল এর শেষে যদি স্ল্যাশ `'/'` না থাকে তাহলে জ্যাঙ্গো অটোমেটিক সেখানে একটা স্ল্যাশ যুক্ত করে দেয়, তাই `http://127.0.0.1:8000/myview` এ রিকুয়েস্ট করলেও সেটা `http://127.0.0.1:8000/myview/` হিসেবে দেখা হবে।

জ্যাক্সো এডমিন পরিচিতি

জ্যাক্সোর অসাধারণ ফিচারগুলোর মধ্যে জ্যাক্সো এডমিন অন্যতম! কিন্তু এই এডমিন জিনিশটা আসলে কি?? ধরুন আপনি বিডিনিউজ বা কালের কণ্ঠের মত একটা নিউজ পাবলিশিং ওয়েবসাইট তৈরি করবেন, যেখানে পাঠক সংবাদ পড়তে পারবে। তো পাঠক কিন্তু শুধু সংবাদ পড়তে পারবে, সংবাদ প্রকাশ করতে পারবেনা! সংবাদ প্রকাশ করবে সেই নিউজ সাইটের সাংবাদিক বা এডিটরগন। সুতরাং সাইটের ডেভেলপার হিসেবে আপনাকে দুটি ইন্টারফেস তৈরি করতে হবে, একটি পাঠক বা ডিজিটরের জন্য যেখানে শুধু সংবাদ বা সাইটের কনটেন্ট দেখা যাবে, আরেকটি এডমিন/এডিটর বা সাইটের মালিকের জন্য, যেখানে সাইটের কনটেন্ট তৈরি বা পরিবর্তন বা ডিলেট করা যাবে!

আপনি যদি ব্লগস্পট, ওয়ার্ডপ্রেস বা অন্যকোন কনটেন্ট ম্যানেজমেন্ট সিস্টেমের সাথে পরিচিত হন তাহলে এটা আপনার কাছে আরো স্পষ্ট হবে। কনটেন্ট ম্যানেজমেন্ট সিস্টেমে পোস্ট করতে হলে আপনাকে এডমিন হিসেবে লগিন করতে হয়, এবং পোস্ট লেখার বা সাইটের অন্যান্য কনটেন্ট ম্যানেজ করার যে প্যানেলটা থাকে সেটাই এডমিন প্যানেল বা ড্যাশবোর্ড! এটা কিন্তু সাধারণ ডিজিটর ব্যবহার করতে পারেনা। শুধু এডমিন বা সাইটের মালিক ব্যবহার করতে পারে!

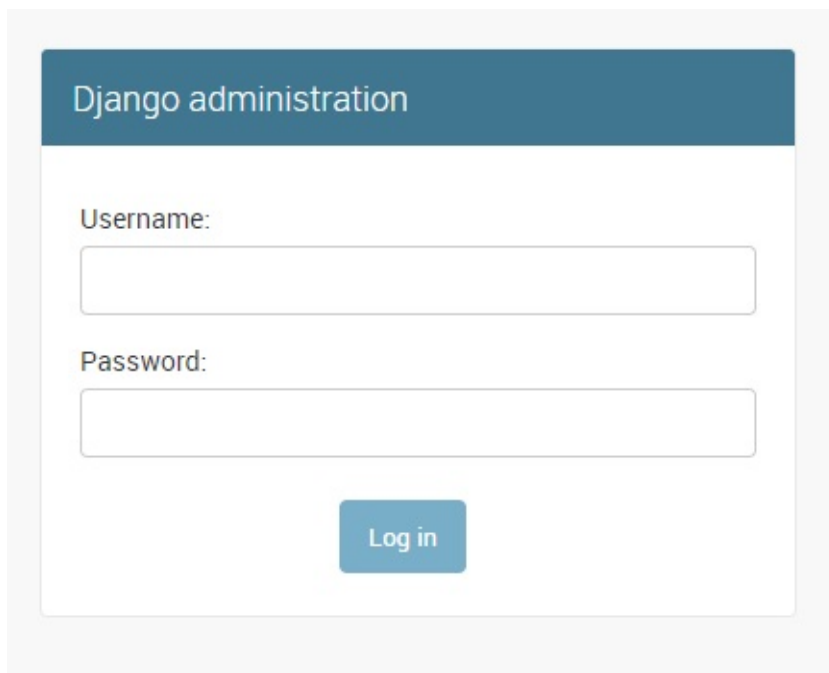
তো প্রায় প্রতিটি ওয়েবসাইটের বাইরের রূপ (যেটা সাধারণ ডিজিটর দেখতে পারে) এর সাথে একটা ভিতরের রূপ (যেটা শুধু এডমিনের জন্য) থাকে। ডেভেলপার হিসেবে আপনাকে এই দুটি রূপই ডেভেলপ বা ডিজাইন করতে হবে!

জ্যাক্সো আপনার এই কষ্ট অর্ধেক কমিয়ে দেয় তার এডমিন অ্যাপ এর মাধ্যমে! আপনি যেকোন সাইটই তৈরি করেননা কেন, সেটার জন্য রেডিমেড একটা এডমিন প্যানেল পেয়ে যাবেন। সব কিছু বিল্টইন সেট করাই থাকবে, আপনার কাজ হল সেটাকে নিজের মত করে কাস্টোমাইজ করে নেয়া!

জ্যাক্সো এডমিন ব্যবহার করতে প্রথমে প্রজেক্ট ডিরেকটরিতে টার্মিনাল ওপেন করে ভার্চুয়াল এনভায়রনমেন্ট একটিভ করুন, তারপর `python manage.py runserver` কমান্ড দিয়ে জ্যাক্সো ডেভলপমেন্ট সার্ভার চালু করুন।

ব্রাউজার ওপেন করে এই ঠিকানায় যান <http://127.0.0.1:8000/admin>

লগিন করার একটা অপশন দেখতে পাবেনঃ



জ্যাক্সো এডমিন প্যানেল আশা করে যে এডমিন (জ্যাক্সোর পরিভাষায় বলে সুপার ইউজার) তার ইউজারনেম এবং পাসওয়ার্ড দিয়ে লগিন করবে। কিন্তু আপনার কাছে তো ইউজারনেম/পাসওয়ার্ড নেই!!

সমস্যা নেই, আমরা ইউজারনেম/পাসওয়ার্ড বানিয়ে নিব, প্রথমে কমান্ড প্রম্পট (টার্মিনাল) এ যান। সেখানে দেখবেন এখনো সার্ভার চালু করা। কন্ট্রোল সি (Ctrl + c) বাটন দুটি চেপে সার্ভার বন্ধ করুন! এরপর সুপার ইউজার তৈরি করার নিচের কমান্ডটি দিনঃ

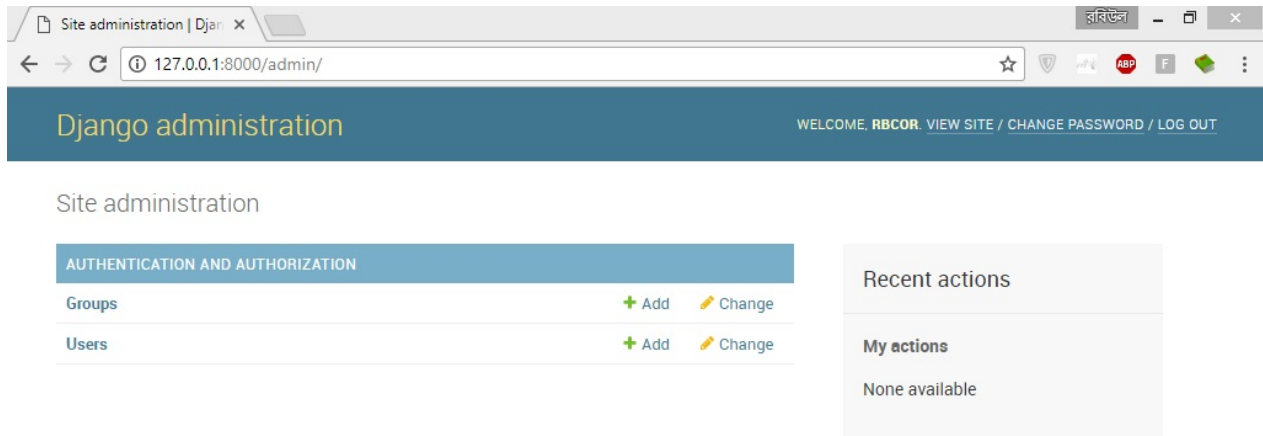
```
python manage.py createsuperuser
```

আপনার ইউজারনেম, ইমেইল এবং পাসওয়ার্ড চাইবে সেগুলো দিন। সঠিক ভাবে সব কিছু দিলে সেখানে সুপার ইউজার তৈরি হয়েছে বলে একটা মেসেজ শো করবে।

লক্ষ্য করুন, পাসওয়ার্ড কিন্তু দুইবার দিতে হবে, পাসওয়ার্ড কমান্ডপ্রম্পট বা টার্মিনালে শো করবেনা তাই পাসওয়ার্ড টাইপ করার সময় সেটা স্ক্রিনে না দেখে ঘাবড়াবেন না! পাসওয়ার্ড স্ট্রং এবং লম্বাচওড়া হতে হবে, ১২৩৪৫ টাইপের পাসওয়ার্ড দিলে কাজ হবেনা!

এখন আবার ডেভেলপমেন্ট সার্ভার চালু করুন (`python manage.py runserver` কমান্ড দিয়ে) ! এবং ব্রাউজারের এডমিন প্যানেলের ঠিকানায় (<http://127.0.0.1:8000/admin>) যেয়ে একটু আগে তৈরি করা সুপার ইউজারের ইউজারনেম/পাসওয়ার্ড দিয়ে লগিন করুন!

লগিন করলে আপনি এরকম একটা কিছু দেখতে পারবেনঃ



আপাতত সবকিছু বুঝে ফেলার দরকার নেই, পেইজটা ভালো করে দেখুন, উপরের ডান পাশের মেনুটা দেখুন, লগআউট, পাসওয়ার্ড চেঞ্জ, ডিউ সাইট! নাম দেখেই বোঝা যাচ্ছে কোনটা কি কাজের! যেটা বোঝা যাচ্ছেনা সেটা হল মারের অংশটা। Site administration লেখার নিচে Groups, Users লেখাগুলো আসলে ‘মডেল’! মডেল চিনেনতো!? এ দুটি মডেল হল জ্যান্সির অথেনটিকেশন এবং অথরাইজেশন অ্যাপ এর মডেল।

আমাদের তৈরি করা অ্যাপ এবং সেটার মডেল এখানে দেখা যাচ্ছেনা, কারন এটা অটোমেটিক ভাবে দেখা যায়না বরং এডমিনে মডেল এড করে নিতে হবে।

আমাদের myapp অ্যাপ এর ভিতর দেখবেন admin.py নামে একটা মডিউল আছে, সেটা ওপেন করুন! ভিতরে এরকম কোড দেখা যাবেঃ

```
from django.contrib import admin

# Register your models here.
```

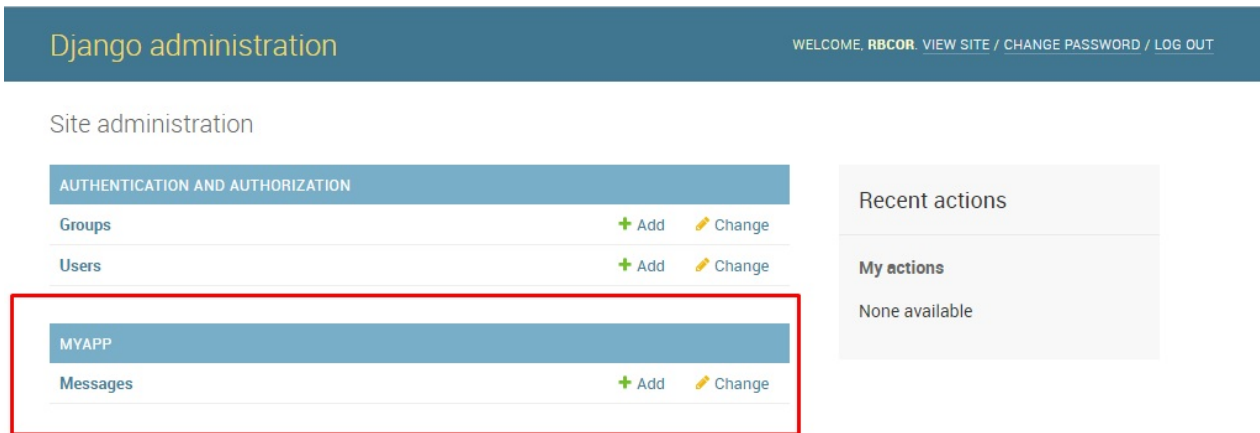
আপনি সেখানে আপনার আপনার মডেল ক্লাসটি ইম্পোর্ট করুন, অতপর সেটা রেজিস্টার করুন। আপনার admin.py এর সম্পূর্ণ কোড এরকম হবেঃ

```
from django.contrib import admin

# Register your models here.
from myapp.models import Message # মডেল ইমপোর্ট করা হন

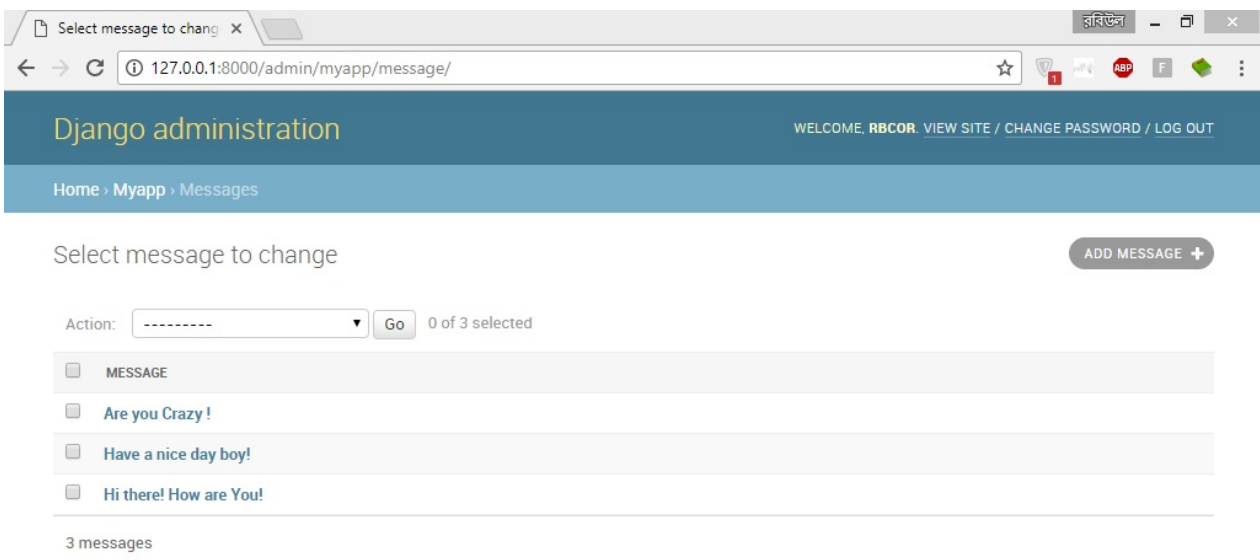
admin.site.register(Message) # মডেলকে এডমিন গ্রাইডে রেজিস্টার করা হন
```

এখন আবার ব্রাউজারে এডমিন প্যানেলে যান, এরকম দেখা যাবেঃ



আমাদের এপটি এখন এডমিন প্যানেলে দেখা যাচ্ছে, লক্ষ্য করুন, এপ এর নামটি বড় হাতের অক্ষরে লেখা MYAPP এবং তার নিচে সেই এপ এর Message মডেলটির নাম দেখা যাচ্ছে, তার পাশে Add এবং Change নামে দুটি বাটন দেখা যাচ্ছে!

এখন আপনি যদি Message লেখাটিতে ক্লিক করেন তাহলে মেসেজ মডেলের অবজেক্টগুলো দেখতে পাবেনঃ



আমাদের তৈরি তিনটি অবজেক্ট (জ্যাসো শেলে তৈরি করেছিলাম! মনে আছে?) সুন্দর ভাবে দেখা যাচ্ছে। এখন এই অবজেক্টগুলোর উপর ক্লিক করে আমরা কিন্তু এগুলো আমাদের মন মত এডিট করতে পারি, আবার ডিলেটও করতে পারি।

প্রথম থাকা অবজেক্ট অর্থাৎ ‘Are you Crazy !’ মেসেজটায় ক্লিক করে ডিতরে প্রবেশ করুন, এরকম একটা এইচটিএমএল টেমপ্লেট এরিয়া দেখতে পাবেন, যেখানে মেসেজটি এডিট করা যাবেঃ

The screenshot shows a web browser window with the URL `127.0.0.1:8000/admin/myapp/message/3/change/`. The page title is 'Django administration'. The breadcrumb trail is 'Home > Myapp > Messages > Are you Crazy !'. The main heading is 'Change message'. There is a 'HISTORY' button. The 'Text:' label is next to a text area containing 'Are you Crazy !'. At the bottom, there are four buttons: 'Delete' (red), 'Save and add another' (blue), 'Save and continue editing' (blue), and 'SAVE' (blue).

এটাকে এডিট করে যদি আমরা ‘Hey! Are you Crazy !’ করে সেভ বাটনে ক্লিক করিঃ

Change message

Text:

Buttons: Delete, Save and add another, Save and continue editing, **SAVE**

তাহলে আমাদের সেই মেসেজটি পরিবর্তন হয়ে যাবে! এভাবে আমরা প্রতিটা অবজেক্টই এডিট করতে পারি।

Select message to change

Message: The message "Hey! Are you Crazy !" was changed successfully.

Action: Go 0 of 3 selected

MESSAGE
<input checked="" type="checkbox"/> Hey! Are you Crazy !
<input type="checkbox"/> Have a nice day boy!
<input type="checkbox"/> Hi there! How are You!

3 messages

এডমিন প্যানেলের মাধ্যমে আমরা যেকোনো অবজেক্টকেই (উপরের উদাহরণের অবজেক্টে শুধু টেক্সট ছিল, কিন্তু আমরা আরো কমপ্লেক্স কোনো অবজেক্টকেও) ক্রিয়েট/রিড/আপডেট/ডিলেট করতে পারব!

আপনার কাজ হল এডমিন প্যানেলটা আরো ভালো করে ঘুরে দেখা, এই চ্যাপ্টারে আমরা শুধু অবজেক্ট রিড এবং আপডেট করা দেখেছি। আপনি নিজেই চেষ্টা করুন কিভাবে এডমিন প্যানেলের মাধ্যমে নতুন অবজেক্ট/ডাটা (আমাদের এপ এর ক্ষেত্রে 'মেসেজ') তৈরি করা যায়! এবং কিভাবে কোন অবজেক্ট/ডাটা ডিলেট করা যায়।

মনে রাখবেনঃ জ্যাঙ্গো ওআরএম এ প্রতিটা মডেল হল ডাটাবেইজের একেকটা টেবিল, মডেলের ফিল্ডগুলো হল ডাটাবেইজের টেবিলের কলাম, মডেলের প্রতিটা ইন্সট্যান্স (মডেল থেকে তৈরি করা প্রতিটা অবজেক্ট) হল টেবিলের ফিল্ড বা ডাটা বা 'রো' !

টিউটোরিয়াল - প্রথম অধ্যায়

নতুন প্রজেক্ট তৈরি করা

ইনস্টলেশন ধাপে আমরা দেখেছি ডার্সন নির্ণয় করতে django-admin.py টুলটির ব্যবহার। এখন আমরা এই টুলটি ব্যবহার করবো নতুন একটি প্রজেক্ট তৈরি করতে। টার্মিনালে এই কমান্ডটি ব্যবহার করুন -

```
django-admin.py startproject mysite
```

এটা mysite নামে একটা ডিরেক্টরী তৈরি করবে। চলুন দেখে নেই mysite ডিরেক্টরীর স্ট্রাকচার কেমন হলো -

```
mysite/
  manage.py
  mysite/
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

পুনশ্চ: আপনি যদি ভুলক্রমে জ্যাঙ্গোর অন্য কোন ডার্সন ইনস্টল করে থাকেন তবে এই স্ট্রাকচার ভিন্ন হতে পারে। যেহেতু এই বইটিতে আমরা এই স্ট্রাকচার অনুসরণ করবো তাই ভালো হয় এরকম স্ট্রাকচার তৈরি করলে।

এবার আসুন এই ডিরেক্টরী এবং ফাইলগুলোর পরিচিতি জেনে নেই -

- একেবারে বাইরের mysite ডিরেক্টরীটি আসলে আমাদের মূল প্রজেক্ট ডিরেক্টরী। এটার নাম আসলে কোন ব্যাপার না।
- manage.py : এটি হলো জ্যাঙ্গোর কমান্ড লাইন টুলগুলোর মধ্য একটি। এটি ব্যবহার করে আমরা প্রজেক্ট সম্পর্কিত নানাবিধ কাজ করতে পারি।
- ভিতরের mysite ডিরেক্টরীটি আমাদের মূল অ্যাপ্লিকেশন যেটি কিনা আসলে একটি পাইথন প্যাকেজ যার ভিতরে আমাদের অ্যাপ্লিকেশনের মৌলিক কিছু কনফিগারেশন আমরা সংরক্ষণ করবো।
- mysite/__init__.py : আমরা জানি __init__.py সাধারণত একটি ফাকা ফাইল হয় যেটা নির্দেশ করে যে এই ডিরেক্টরীটি আসলে একটি পাইথন প্যাকেজ। সাধারণ ডিরেক্টরি এবং পাইথন প্যাকেজের পার্থক্য করার জন্য এটি বেশ কাজের।
- mysite/settings.py : অ্যাপ্লিকেশনের সেটিংসগুলো এই ফাইলে রাখি আমরা।
- mysite/urls.py : এই ফাইলে থাকে এই অ্যাপ্লিকেশনের সকল URL এর সংজ্ঞা। এই ফাইলেই আমরা জ্যাঙ্গোকে বলে দেই কোন URL কিভাবে হ্যান্ডল করবো আমরা।
- mysite/wsgi.py : এই ফাইলটি WSGI কম্প্যাটিবল সার্ভার ব্যবহার করার সময় কাজে লাগে। এই ফাইলের কেরামতি আমরা যখন ডেপ্লয়মেন্ট নিয়ে আলোচনা করবো তখন দেখবো।

ডেভেলপমেন্ট সার্ভার চালু করা

সবকিছু ঠিকঠাক আছে কিনা দেখার জন্য এবার আমরা ডেভ সার্ভার চালু করবো । জ্যাঙ্গো অ্যাপ ডেভেলপ করার জন্য এর সাথেই চমৎকার একটি সার্ভার দেওয়া থাকে । ফলে সহজেই আমরা অন্য কোন সার্ভারের প্রয়োজন ছাড়াই অ্যাপ্লিকেশনটি চালু করতে পারি এই সার্ভার দিয়ে । এটি চালু করতে কমান্ড দিন -

```
python manage.py runserver
```

আউটপুট হবে নিচের মত -

```
Validating models...

0 errors found
May 24, 2013 - 15:50:53
Django version 1.5, using settings 'mysite.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

আমরা যদি `http://127.0.0.1:8000/` অ্যাড্রেসটি আমাদের ব্রাউজারে প্রবেশ করাই তবে আমরা জ্যাঙ্গোর হোম পেইজটি দেখতে পাবো ।

পুনশ্চ: আমরা চাইলে খুব সহজেই সার্ভারের পোর্ট বদল করতে পারি । যেমন আমরা যদি চাই সার্ভারটি ৪০৪০ পোর্টে রান করুক তবে আমরা `runserver` কমান্ডটি ব্যবহার করবো সাথে অতিরিক্ত পোর্ট নাম্বারটি দিয়ে দিবো -

```
python manage.py runserver 8080
```

ডাটাবেইজ সেটাপ করা

আমরা যদি `mysite/settings.py` ফাইলটি টেক্সট এডিটরে খুলি তবে দেখবো ডাটাবেইজ এর জন্য একটি আলাদা সেকশন আছে ।

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.', # Add 'postgresql_psycopg2', 'mysql', 'sqlite3' or 'oracle'.
        'NAME': '', # Or path to database file if using sqlite3.
        # The following settings are not used with sqlite3:
        'USER': '',
        'PASSWORD': '',
        'HOST': '', # Empty for localhost through domain sockets or '127.0.0.1' for localhost through TCP.
        'PORT': '', # Set to empty string for default.
    }
}
```

এখানে -

- `ENGINE` হলো ডাটাবেইজ ড্রাইভারের নাম ।
- `NAME` হলো ডাটাবেইজের নাম । SQLite এর বেলায় ফাইলের নাম ।
- `USER` ডাটাবেইজের ইউজার নেইম ।
- `PASSWORD` হলো পাসওয়ার্ড ।
- `HOST` হচ্ছে ডাটাবেইজ সার্ভার যে হোস্টে রান করছে তার নাম । লোকাল মেশিনের হোস্টনেম হবে `localhost` । উহ্য রাখলে জ্যাস্পো লোকালহোস্টই ব্যবহার করবে ।
- `PORT` ডাটাবেইজ সার্ভার যে পোর্টে চালু আছে । সাধারণত এটির কোন মান দেওয়া লাগে না ।

আমরা একটি পূর্ণাঙ্গ উদাহরণ দেখে নেই:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'student_database',
        'USER': 'root',
        'PASSWORD': 'Pa55w0Rd',
        'HOST': '',
        'PORT': '',
    }
}
```

syncdb - ডাটাবেইজ সিঙ্ক্রোনাইজেশন

ধরে নিচ্ছি আমরা যে ডাটাবেইজ ইনফরমেশন দিয়েছি `settings.py` এ তা সঠিক । এবার তাহলে ডাটাবেইজ সিঙ্ক্রোনাইজেশন এর পালা । একটি জ্যাস্পো সাইট বা প্রজেক্ট অনেকগুলো ছোট ছোট অ্যাপ্লিকেশনের সমন্বয়ে তৈরি হয় । এই অ্যাপ্লিকেশনগুলি `settings.py` এর `INSTALLED_APPS` সেকশনে যোগ করা হয় । বাই ডিফল্ট কতগুলো অ্যাপ্লিকেশন যোগ করাই থাকে এই লিস্টে । অনেক অ্যাপ্লিকেশনই ডাটাবেইজে তথ্য সংরক্ষণ করে । তাই এদের প্রয়োজন পড়ে টেবিল তৈরি করার । `manage.py` এর `syncdb` কমান্ডটি এই কাজটিই করে দেয় আমাদের জন্য । আমরা টার্মিনালে রান করবো -

```
python manage.py syncdb
```

জ্যাস্পো একে একে সব গুলো অ্যাপ্লিকেশন এর জন্য টেবিল তৈরি করবে । এরপর একটি সুপারইউজার তৈরি করতে চাইবে । সম্মতি দিয়ে পছন্দমত ইউজারনেইম, ইমেইল অ্যাড্রেস এবং পাসওয়ার্ড দিয়ে তৈরি করে ফেলুন সুপারইউজারটি । পরে কাজে লাগবে । প্রসেসটি বেশ সিম্পল । স্ক্রীনের ইনস্ট্রাকশন ফলো করলেই কাজটি সম্পন্ন করতে পারবেন বেশ করে ।

পুনশ্চ: জ্যাস্পো শুধু মাত্র `INSTALLED_APPS` তালিকায় থাকা অ্যাপ্লিকেশনগুলোর জন্যই টেবিল তৈরি করবে । যে সকল অ্যাপ্লিকেশন আপনার প্রয়োজন সেগুলো যোগ করে `syncdb` চালান । তবে একবার টেবিল তৈরি করে পরে অ্যাপ্লিকেশন তালিকা থেকে রিমুভ করে দিলেও জ্যাস্পো ঐ টেবিল ডাটাবেইজ থেকে রিমুভ করবে না । যেসব অ্যাপ্লিকেশন আপনার দরকার পড়বে না সেগুলো `syncdb` চালানোর আগেই এই তালিকা থেকে মুছে দেওয়া বা কমেন্ট আউট করাই শ্রেয় ।

প্রজেক্ট vs অ্যাপ্লিকেশন

আমরা শুরুতেই দেখেছি প্রজেক্ট তৈরি করা। এরপর `syncdb` সেকশনে বলেছি একটি প্রজেক্ট হলো এক বা একাধিক অ্যাপ্লিকেশনের সমষ্টি। একটি জ্যাকসো সাইট বা প্রজেক্ট এর ওভার অল ফাংশনালিটি আমরা ছোট ছোট ইউনিটে ভাগ করে ফেলি। এরপর প্রত্যেকটি ভিন্ন ভিন্ন ফিচারের জন্য আমরা আলাদা আলাদা অ্যাপ্লিকেশন ডেভেলপ করি। একটু ব্যাখ্যা করি, ধরুন আপনি একটি কমিউনিটির জন্য সাইট ডেভেলপ করছেন। আপনার প্রয়োজন পড়বে -

- ব্লগ
- ফোরাম
- ফটো বা ভিডিও গ্যালারি

জ্যাকসোতে আমরা প্রত্যেকটি ফাংশনের জন্য আলাদা আলাদা অ্যাপ্লিকেশন তৈরি করবো। এতে লাভ - একই ধরনের কাজ করে এমন সব কোড এক জায়গায় থাকছে, এই অ্যাপ্লিকেশনটি চাইলে আপনি অন্য আরেকটি প্রজেক্ট এও ব্যবহার করতে পারছেন খুব সহজে। জ্যাকসোর অ্যাপ্লিকেশনগুলো অনেকটা প্লাগ অ্যান্ড প্লে ধাঁচের। আপনি শুধু মূল অ্যাপ্লিকেশনের `settings.py` এর `INSTALLED_APPS` সেকশনে যোগ করে দিবেন, একবার `syncdb` করবেন এবং তারপর `urls.py` তে রাউট যোগ বা ইম্পোর্ট করবেন। ব্যাস, কাজ শেষ! এই ধরনের ফ্লেক্সিবিলিটির কারণে জ্যাকসোতে কাজ করতে অনেক মজা এবং সময়ও কম লাগে। বিশাল কমিউনিটির সবাই মিলে এমন হাজার হাজার অ্যাপ্লিকেশন ডেভেলপ করে রেখেছে। আপনার যা প্রয়োজন, সেটি ইনস্টল করে নিন অথবা হালকা পরিবর্তন পরিবর্তন করে নিজের মত করে ব্যবহার করুন।

এবার তৈরি করি নিজেদের অ্যাপ্লিকেশন

জ্যাকসো শিখতে এবার আমরা নিজেরাই একটি অ্যাপ্লিকেশন তৈরি করবো। জ্যাকসোর অফিশিয়াল টিউটোরিয়ালের সাথে মিল রেখে ওদের অ্যাপ্লিকেশনটিই তৈরি করবো আমরা। অ্যাপ্লিকেশনের কনসেপ্ট বেশ সহজ। একটা "পোল" অ্যাপ। অনলাইনে জন্মদাত জরিপ করার জন্য বিভিন্ন দৈনিক পত্রিকাগুলো যেমনটি করে, ঠিক তেমন একটি অ্যাপ। এখানে কতগুলো প্রশ্ন থাকবে। প্রত্যেক প্রশ্নের কতগুলো নির্দিষ্ট উত্তর থাকবে। ডিজিটররা ইচ্ছামত ভোট দিতে পারবে।

একটা অ্যাপ্লিকেশনের স্ট্রাকচার তৈরি করার জন্য `manage.py` এর `startapp` কমান্ডটি ব্যবহার করবো এভাবে -

```
python manage.py startapp polls
```

তাহলে আমাদের প্রজেক্ট রুটে আরেকটি ফোল্ডার (আসলে পাইথন প্যাকেজ) তৈরি হবে `polls` নামে। এবার এটির স্ট্রাকচার দেখে নেই -

```
polls/
  __init__.py
  models.py
  tests.py
  views.py
```

এখানে -

- `__init__.py` নির্দেশ করে যে ডিরেক্টরীটি একটি পাইথন প্যাকেজ
- `models.py` এ আমরা আমাদের প্রয়োজনীয় ডাটা মডেল তৈরি করবো
- `tests.py` এ আমরা ইউনিট টেস্টিং এর জন্য টেস্ট কেইসগুলো রাখবো
- `views.py` গতানুগতিক MVC ফ্রেমওয়ার্ক থেকে জ্যঙ্গো একটু ভিন্ন । জ্যঙ্গোতে Controller এর পরিবর্তে আমরা View ব্যবহার করি । এই ফাইলে আমরা সেই ভিউগুলো তৈরি করবো ।

MVC vs MTV

MVC ফ্রেমওয়ার্কের কম্পোনেন্টগুলো হয় এমন -

- `Models` : ডাটা মডেল
- `View` : ব্যবহারকারীরা যা দেখে । সাধারণত HTML, CSS, JS থাকে ।
- `Controller` : বিজনেস লজিক থাকে । কন্ট্রোলার মডেল থেকে ডাটা নিয়ে ভিউ এর সাথে সমন্বয় করে ।

জ্যঙ্গোতে আমরা MTV (Model, Template, View) প্যাটার্ন ফলো করবো । যদি MVC থেকে সরাসরি অনুবাদ করি তবে -

- `Model` এর জায়গায় `Model` ই থাকছে
- `View` এর কাজটা জ্যঙ্গোতে করছে `Template`
- `Controller` এর জন্য আছে `View`

জ্যঙ্গোর `View` নিয়ে ইনশাআল্লাহ পরবর্তিতে আরো বিস্তারিত আলোচনা থাকবে ।

ডাটা মডেলিং - মডেল তৈরি করা

জ্যঙ্গোর দর্শন হচ্ছে অ্যাপ্লিকেশনের ডাটা একটি সেন্ট্রাল মডেলে রাখা । এরপর প্রয়োজন মারফিক এই মডেলের উপর ভিত্তি করে অন্যান্য জিনিসগুলো অটোমেটিক্যালি জেনারেট করা । এই বিষয়ে আরো বিস্তারিত আমরা পরে দেখবো । আপাতত আমাদের ভোটিং অ্যাপ্লিকেশনের জন্য মডেল তৈরি করে নেই । আগেই বলেছি সব মডেল থাকবে অ্যাপ্লিকেশনের `models.py` তে । সে হিসাবে আমরা `polls/models.py` ফাইলটি খুলে নিচের কোড টাইপ করি -

```
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

আপনি যেহেতু পাইথনের পূর্ব ধারণা রাখেন সেহেতু এই কোড বেশ সহজবোধ্যই মনে হবে । মূলকথা হলো জ্যঙ্গোর একেকটি মডেল হলো একটি ক্লাস যা `models.Model` ক্লাসের সাবক্লাস । মডেলগুলো ডাটাবেইজের টেবিলকে রিপ্রেজেন্ট করে । ক্লাসের ভ্যারিয়েবল টেবিলের ফিল্ড বা কলামকে নির্দেশ করে । এই ভ্যারিয়েবলগুলোর টাইপ নির্ধারণ

করে টেবিলের কলাম বা ফিল্ডের ডাটা টাইপ কি হবে। এই উদাহরণে আমরা দেখলাম CharField যেটা অল্পকিছু ক্যারেक्टर (কিংবা ছোট খাটো স্ট্রিং) ধারণ করে। IntegerField ইন্টিজার (পূর্ণ সংখ্যা) এবং DateTimeField দিন তারিখের জন্য ব্যবহার করা হয়। এরকম অনেকগুলো ডিফল্ট ফিল্ড টাইপ জ্যাকসের models মডিউলে পাওয়া যায়। তবে সবগুলো আমাদের আপাতত না জানলেও চলবে।

মডেল এর ব্যবহার

মডেল তৈরি করেছি আমরা, এবার এটাকে বাস্তবে ব্যবহার করার পালা। মডেলটি ব্যবহার করতে প্রথমেই আমাদের প্রয়োজন হবে ডাটাবেইজ সিস্টেম করা। এতে জ্যাকসো আমাদের জন্য প্রয়োজনীয় টেবিল তৈরি করে দিবে। তবে ডাটাবেইজ সিস্টেম করার জন্য প্রথমেই আমাদের এ্যাপটিকে settings.py এর INSTALLED_APPS এ যোগ করে নিতে হবে:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'polls',
)
```

এবার রান করুন -

```
python manage.py syncdb
```

মডেল এর বাস্তবিক ব্যবহার বোঝার জন্য আমরা ইন্টারঅ্যাকটিভ শেল ব্যবহার করবো। পাইথনের ইন্টারঅ্যাক্টিভ প্রম্পট এর মতই এটি জ্যাকসো এপিআই ব্যবহার করার সুযোগ করে দেয়। এটি চালানোর জন্য আমরা এই কমান্ডটি ব্যবহার করবো -

```
python manage.py shell
```

যেই শেলটি চালু হবে সেটিতে আমরা কোড লাইন বাই লাইন কোড লিখে আউটপুট পরখ করবো -

প্রথমেই আমাদের মডেল ক্লাসটি ইম্পোর্ট করে নেই -

```
>>> from polls.models import Poll, Choice
```

পেতে চাই সবগুলি Poll -

```
>>> Poll.objects.all()
[]
```

কিন্তু আমরাতো কোন Poll যোগ করিনি এখনো তাই একটি ফাকা লিস্ট পেলাম ।

তাহলে তৈরি করি প্রথম Poll টি -

```
>>> from django.utils import timezone
>>> p = Poll(question="What's new?", pub_date=timezone.now())
>>> p.save()
```

জ্যাস্কে অবজেক্টটি ডাটাবেইজে সেইভ করবে । এবার দেখি অবজেক্টটির নানা এ্যাট্রিবিউটগুলো -

```
>>> p.id
1
>>> p.question
"What's new?"
>>> p.pub_date
datetime.datetime(2012, 2, 26, 13, 0, 0, 775217, tzinfo=<UTC>)
```

এবার দেখি কিভাবে খুব সহজেই এটিকে পরিবর্তন করা যায় -

```
>>> p.question = "What's up?"
>>> p.save()
```

এবার আবার সবগুলো Poll অবজেক্ট এর তালিকাটি দেখে নেই -

```
>>> Poll.objects.all()
[<Poll: Poll object>]
```

এভাবেই আমরা খুব সহজে মডেল তৈরি এবং ব্যবহার করতে পারি । ভবিষ্যতে আমরা মডেলের উপর আরো বিস্তারিত দেখবো ।

শুরুর আগে

ওয়েব ফ্রেমওয়ার্ক পরিচিতি

আমি ধরে নিচ্ছি যে, ওয়েব ফ্রেমওয়ার্ক জিনিসটির সঙ্গে আপনি পরিচিত নন এবং কোনওদিনও পাইথন বা অন্য কোন ল্যান্গুয়েজে ওয়েব ফ্রেমওয়ার্ক ব্যবহার করেননি। ওয়েব ফ্রেমওয়ার্ককে ডিফাইন করার সবচেয়ে সহজ উপায় হল বলা যে, এটি আপনার ওয়েব ডেভেলপমেন্টের কাজকে সহজ করে দেয়। একটু ইতিহাস টানছি (বেশি না)।

জ্যাক্সের ডেভেলপাররা আগে বড় বড় নিউজমিডিয়া বা পত্রিকার জন্য ওয়েবসাইট তৈরি করে দিতো। একদিন কাজ করতে গিয়ে তারা হঠাৎ করে খেয়াল করলো যে, "আরে! দৈনিক অমুক পত্রিকার ওয়েবসাইট আর দৈনিক তমুক পত্রিকার ওয়েবসাইটের কোডের প্রায় ৯০%-ই একইরকম!" অর্থাৎ, তারা নতুন নতুন ওয়েবসাইট তৈরি করতে গিয়ে বোকার মত একই কোড বার বার লিখছিল।

তখন তাদের মাথায় একটা সাংঘাতিক আইডিয়া এসে গেল! "আচ্ছা, আমরা যদি এমন একটা "জিনিস" বানাই যাতে বারবার আমাদের গাধার মত একই কোড লিখতে না হয়, একবার লিখে ফেলবো এবং দরকারমত বিভিন্ন প্রজেক্টে ওই জিনিসটা ব্যবহার করবো!" ইউরেকা! এরকম চিন্তাভাবনা থেকেই জন্ম নেয় জ্যাক্স ওয়েব ফ্রেমওয়ার্কের।

ফ্রেমওয়ার্ক নিলে আমার লাভ কি?

আপনি জিজ্ঞেস করতে পারেন, ফ্রেমওয়ার্ক কেন ব্যবহার করবো? কয়েকটি সোজা উদাহরণ দিয়ে বুঝিয়ে দিচ্ছি! যেমন ধরুন আপনি একটি ব্লগ বানাবেন, নিজে কোড করে ব্লগ বানাবেন ডাব মারার পাশাপাশি আপনার বদের হাড্ডি ছোট ভাইটা ওয়ার্ডপ্রেস বা অন্য কোন সিএমএসের নিরাপত্তার দুর্বলতা জেনে ব্লগের সামনে বিশাল "হ্যাক করছি" নোটিশ যাতে না দিতে পারে।

ধরি আপনি প্লেইন পাইথন বা প্লেইন পিএইচপিতে বা অন্য কোন ভাষার কোন ওয়েব ফ্রেমওয়ার্ক ব্যবহার না করে লিখবেন। তাহলে আপনার কাজের লিস্ট মোটামুটি...

1. ডাটাবেজের সঙ্গে সংযোগ দেয়া
2. ডাটাবেজে SQL লিখে টেবিল তৈরি করা
3. টেবিলে ডাটা INSERT ও UPDATE করানোর জন্য অ্যাডমিন ইন্টারফেস (যেখানে ব্লগ পোস্ট লিখবেন, আপডেট করবেন) তৈরি করা
4. সাইট সুরক্ষিত রাখতে সিকিউরিটি নিশ্চিত করা; CSRF , XSS ... প্রটেকশন দেয়া।
5. ডাটাবেজের ডাটা SELECT করে সাইটের ডিজিটরদের জন্য সুন্দর ফ্রন্টএন্ড (সাইটের যে অংশ ডিজিটররা দেখবে) তৈরি করা

ঘামতে শুরু করেছেন? ঘামা ভালো, ঘামের সঙ্গে নাকি শরীরের বর্জ্য বেড়িয়ে যায়। তবে, অতিরিক্ত ঘামা শরীরের জন্য খারাপ! তাই এখনই বলে ফেলছি, ওপরের ছয়টি কাজের প্রত্যেকটিই জ্যাক্স স্বয়ংক্রিয়ভাবে করতে আপনাকে যথেষ্ট পরিমাণে সহায়তা করবে। জ্যাক্সকে বলে দিন, ডাটাবেজের নাম, ঠিকানা, পোর্ট নম্বর, ইউজারনেম আর পাসওয়ার্ড - সে নিজেই কানেক্ট করে নিবে যখন প্রয়োজন।

আপনার দেয়া নির্দেশনা বা প্ল্যান অনুযায়ী নিজেই ডাটাবেজের সঙ্গে যোগাযোগ করে কাজ করে টেবিল তৈরি, যাবতীয় তথ্য লেখা, পড়া বা আপডেট করার কাজ করে নেবে, স্বয়ংক্রিয়ভাবে নিরাপদ অ্যাডমিন ইন্টারফেস তৈরি করে দেবে! রক্তচাপ বেড়ে গেছে? আর সাসপেন্স রাখবো না, পরের চ্যাপ্টার থেকেই শুরু করবো কি করে এই অসাধারণ ফ্রেমওয়ার্কটি ব্যবহার করে একদম কম সময়ের মধ্যে কি করে সহজেই সাংঘাতিক সব ওয়েব অ্যাপ্লিকেশন তৈরি করতে পারবেন!

ইনস্টলেশন

কি কি লাগবে?

- পাইথন (আমরা Python 2.7.4 ব্যবহার করছি, Python 3 ব্যবহার করবো না)
- টেক্সট এডিটর বা আইডিই (আমি Sublime Text 2 রিকমেন্ড করি)
- ডাটাবেইজ সার্ভার (MySQL হলেই চলবে)
- pip (পাইথন প্যাকেজ ইন্সটল করার জন্য এটি লাগবে । [ইনস্টলেশন গাইড](#))

প্রথম ধাপ - জ্যাঙ্গো ইনস্টল করা

```
pip install Django==1.5
```

এই কমান্ডটি জ্যাঙ্গো ফ্রেমওয়ার্ক এবং তার প্রয়োজনীয় টুলগুলো ইনস্টল করে নিবে ।

দ্বিতীয় ধাপ - মাইসিকুয়েল বাইন্ডিং ইনস্টল করা

যেহেতু আমরা জ্যাঙ্গোর সাথে মাইসিকুয়েল ব্যবহার করবো সেহেতু আমাদের পাইথনের জন্য মাইসিকুয়েল লাইব্রেরী ইনস্টল করে নিতে হবে ।

```
pip install MySQL-python
```

এই প্যাকেজটি সি থেকে কম্পাইল করা হয় তাই ইন্সটল করার জন্য বেশ কিছু ডিপেন্ডেন্সীর প্রয়োজন পড়ে (যেমন: পাইথন এবং মাইসিকুয়েলের সি হেডার) । আপনার সিস্টেমে যদি সব গুলো ডিপেন্ডেন্সী না থাকে তবে ইরর মেসেজ দেখে দেখে সেগুলো ইনস্টল করে নিতে হবে ।

যারা ডেবিয়ান বা উবুন্টুতে আছেন, তাদের জন্য অবশ্য ইনস্টলেশন বেশ সহজ -

```
sudo apt-get install python-mysqldb
```

এই এক কমান্ডেই প্যাকেজ ম্যানেজার যা যা লাগবে সব ইনস্টল করে তারপর মাইসিকুয়েল বাইন্ডিং ইনস্টল করে দিবে ।

জ্যাঙ্গোর ভার্সন মিলিয়ে দেখা

আমরা ভার্সন 1.5 ব্যবহার করবো এই বইতে । তাই আসুন, জ্যাঙ্গোর ভার্সন দেখে নেই -

```
django-admin.py --version
```

এই কমান্ডটি রান করলে আমরা জ্যাক্সের ভার্সনটি জানতে পারবো । যদি আউটপুট আসে 1.5.* ফর্ম্যাটে তাহলে আমরা সঠিক ভার্সনই ইনস্টল করেছি ।

টিউটোরিয়াল - দ্বিতীয় অধ্যায়

জ্যাঙ্গো এ্যাডমিন - পরিচিতি

জ্যাঙ্গোর জনপ্রিয় ফিচারগুলোর মধ্যে এর এ্যাডমিন এ্যাপ্লিকেশনটি বেশ জনপ্রিয়। এই এ্যাপ্লিকেশনটি জ্যাঙ্গোর সাথেই আসে। এটি যেকোন সাইট এ্যাডমিনিস্ট্রেশন অনেক সহজ করে দেয়। তো আসলে এটা করে টা কি? এই এ্যাপটিকে আপনার মডেল গুলোর নাম জানিয়ে দিলে সে আপনার জন্য সকল ডাটা অপারেশন (CRUD == Create Read Update Delete) করার জন্য বেশ চমৎকার ইন্টারফেইস তৈরি করে দিবে যেমন কোন কোডিং ছাড়াই। আপনি সেই ইন্টারফেইস ব্যবহার করে ডাটাবেইজে নতুন ডাটা ইনপুট করতে পারবেন, পুরাতন ডাটা দেখা, পরিবর্তন করা কিংবা মুছে ফেলতেও পারবেন বেশ সহজে। কষ্ট করে আর এ্যাডমিন প্যানেল বানানোর প্রয়োজন নেই।

এ্যাডমিন এ্যাপ্লিকেশন চালু করা

জ্যাঙ্গো এ্যাডমিন এ্যাপ্লিকেশনটি পাওয়া যাবে `django.contrib.admin` প্যাকেজে। তাই আমাদেরকে `settings.py` এর `INSTALLED_APPS` সেকশনটিতে এই এ্যাপ্লিকেশন টি যোগ করে নিতে হবে। এই কাজটি কিভাবে করতে হবে তার উদাহরণ আমরা প্রথম অধ্যায়ে দেখেছি।

এ্যাডমিন এ্যাপ্লিকেশনটি তার ব্যবহারের জন্য বেশ কিছু ডাটা মডেল সরবরাহ করে। তাই এ্যাপ্লিকেশনটি ব্যবহার করার আগে আমাদের `syncdb` কমান্ডটিও চালানো প্রয়োজন হবে -

```
python manage.py syncdb
```

এই প্রসেসের শেষে একটি সুপার ইউজার তৈরি করার প্রম্পট পাওয়া যাবে। এ ব্যাপারে আগের অধ্যায়ে আলোচনা করা হয়েছে। এই কমান্ড চালানোর ফলে জ্যাঙ্গো এ্যাডমিন এ্যাপের জন্য প্রয়োজনীয় সব টেবিল তৈরি করে নিবে।

এবার আমাদেরকে বলে দিতে হবে কোন URL ডিজিট করলে এ্যাডমিন এ্যাপটি পাওয়া যাবে। জ্যাঙ্গোর এই ডেফিনিশনগুলো থাকে `mysite/urls.py` ফাইলে।

আমরা ফাইলটি এডিট করে এরকম রূপ দেই -

```
from django.conf.urls import patterns, include, url

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    url(r'^admin/', include(admin.site.urls)),
)
```

জ্যাক্সের `urls.py` তে মূলত একটি ড্যারিয়েবল থাকে `urlpatterns` নামে যেটিতে আমরা সব URL এর জন্য ডেফিনিশন প্রদান করি। এটি কিভাবে কাজ করে তা আমরা জ্যাক্সো এপিআই থেকে আরো বিস্তারিত দেখে নিবো পরবর্তীতে কোন এক সময়।

আমরা আপাতত এই `urlpatterns` এ আমাদের ডেফিনিশন যোগ করে দিলাম।

```
url(r'^admin/', include(admin.site.urls))
```

এটির মানে হলো যদি রিকুয়েস্ট করা URL এর প্রথমে `admin` থাকে তবে যেন জ্যাক্সো `admin.site.urls` থেকে ডেফিনিশন ইম্পোর্ট করে নেয়। `include` ব্যবহার করে আমরা আমাদের `urlpatterns` এর মধ্যে প্রয়োজনীয় url ইম্পোর্ট করে নিতে পারি সহজেই।

চালু করি ডেভ সার্ভার

এ্যাডমিন এ্যাপ্লিকেশনটি চেখে দেখার জন্য আমাদের এবার ডেভেলপমেন্ট সার্ভারটি চালু করে নিতে হবে। কমান্ডটি যদিও এর আগে দেখানো হয়েছে তবু আরেকবার দেখে নেই -

```
python manage.py runserver
```

পুনশ্চ: এই কমান্ডটি জ্যাক্সো এ্যাপ্লিকেশন ডেভেলপমেন্টে বোধহয় সবচাইতে বেশী ব্যবহৃত কমান্ড। তাই এটি আশ্বস্ত করা অতীব জরুরী।

ডেভ সার্ভার রান করলে এবার চটপট ব্রাউজারে এই URL টা আমরা ভিজিট করি -

```
http://127.0.0.1:8000/admin/
```

যদি আপনি সব কিছু ঠিক ঠাক মতো করে থাকেন তবে এরকম একটি ইন্টারফেইস পাবেন -



আর যদি এরকম স্ক্রীন না এসে কোন ইরর মেসেজ দেখায় তবে ধাপগুলো আবার চেক করুন, খুঁজে বের করুন কোথায় ভুল হয়েছে।

এ্যাডমিন প্যানেলে লগিন করা

জ্যাক্সো এ্যাডমিন এ্যাপটি যোগ করে প্রথমবার `syncdb` রান করার পর একটি সুপার ইউজার তৈরি করেছিলাম আমরা। এ্যাডমিন প্যানেলে প্রবেশ করার জন্য সেই এ্যাকাউন্টটির ইউজারনেইম এবং পাসওয়ার্ড ব্যবহার করুন। যদি সেই সময় সুপার ইউজার তৈরি করা না হয়ে থাকে বা কোন কারণে নতুন আরেকটি এ্যাকাউন্ট প্রয়োজন হয় তবে নতুন সুপার ইউজার তৈরি করার জন্য এই কমান্ডটি রান করুন -

```
python manage.py createsuperuser
```


লগিন করার পর আমরা এরকম একটি স্ক্রীন দেখতে পাবো -



এখান থেকে আমরা বিভিন্ন এ্যাপ্লিকেশন এর মডেলগুলো এ্যাক্সেস করতে পারি । উদাহরণ স্বরূপ জ্যাক্সোর `Auth` এবং `Sites` এ্যাপ্লিকেশনের মডেলগুলো আমরা এই স্ক্রীনশটে দেখতে পারছি ।

নিজেদের মডেল এ্যাডমিনে যোগ করা

এ্যাডমিন এ্যাপ্লিকেশন সংক্রান্ত সকল কোড রাখতে হবে আমাদের এ্যাপ্লিকেশন প্যাকেজের ভিতর `admin.py` নামক একটি ফাইলে । জ্যাক্সো এ্যাডমিন লোড হওয়ার সময় সকল ইনস্টল্ড এ্যাপ্লিকেশন এর ভিতর `admin.py` ফাইলগুলোকে রান করে । ফলে এই ফাইলে আমরা এ্যাডমিন এ্যাপ্লিকেশনকে বলে দিতে পারি আমরা এ্যাডমিন প্যানেলে আমাদের এ্যাপ্লিকেশন থেকে কি কি করতে দিতে চাই ।

আপাতত আমাদের চাওয়া খুব সিম্পল । আমাদের ভোটাবোটির জন্য `Poll` মডেলটি দরকার এ্যাডমিন ইন্টারফেইসে । তাই আমরা `polls/admin.py` ফাইলটি তৈরি করে সেখানে এই কোড ব্যবহার করবো -

```
from django.contrib import admin
from polls.models import Poll

admin.site.register(Poll)
```

এই কোডে আসলে আমরা জ্যাক্সো এ্যাডমিনকে বলে দিচ্ছি আমাদের `Poll` মডেলটিকে এ্যাডমিন সাইটে রেজিস্টার করে নিতে । অর্থাৎ আমরা এ্যাডমিন এ্যাপকে বলে দিলাম আমাদের মডেলটির কথা যেন এ্যাডমিন এ্যাপ আমাদের সুযোগ করে দেয় সহজে এই মডেলটি ব্যবহার করার ।

ফাইলটি এ্যাড করার পর ব্রাউজারে আবার এ্যাডমিন সাইটে প্রবেশ করুন । এবার চিত্রটি হবে অনেকটা এরকম -



`Polls` লিংক এ ক্লিক করলে এরকম একটি স্ক্রীন পাবেন -



আমরা চাইলে আমাদের আগেই সেইভ করা এন্ট্রিটি পরিবর্তন করতে পারি -



জ্যাক্সো এ্যাডমিন - সারমর্ম

এতক্ষণ ধরে আমরা যা দেখলাম তা হলো কত সহজে আমাদের ডাটা মডেলকে ব্যবহার করা যায় জ্যাক্সো এ্যাডমিন থেকে । জ্যাক্সো এ্যাডমিন বেশ কাস্টোমাইজেবলও বটে । নানা ধরনের অপশন ব্যবহার করে এটিকে সাজিয়ে নেওয়া যায় নিজের ইচ্ছামত । এ্যাডমিন সাইট কাস্টোমাইজেশন নিয়ে পরবর্তীতে বিস্তারিত আলোচনা থাকবে ইনশাআল্লাহ!

টিউটোরিয়াল - তৃতীয় অধ্যায়

ডিউ - ব্যাসিক

আমরা আগের আলোচনায় দেখেছি জ্যাঙ্গো একটি MVT (Model View Template) ফ্রেমওয়ার্ক। এই পর্বে আমরা ডিউ নিয়ে কাজ করবো। জ্যাঙ্গোতে একেকটি ডিউ হলো একেকটি পেইজের মত। প্রত্যেকটি ডিউ একটি নির্দিষ্ট কাজ করে এবং সাধারণত কাজ শেষে একটি টেম্পেট রেন্ডার করে।

যেমন আমরা যদি আমাদের অনলাইন পোল এ্যাপটির কথাই চিন্তা করি তবে আমাদের কতগুলো ডিউ এর দরকার পড়বে -

- একটি পেইজে থাকবে সবগুলো পোল এর তালিকা। যেখান থেকে লিংক এ ক্লিক করে আমি যে কোন একটি নির্বাচন করতে পারবো।
- একটি পেইজে থাকবে নির্বাচিত পোল এর ডিটেইলস। এখানে থাকবে প্রশ্ন এবং ভোট করার জন্য অপশনগুলো।
- রেজাল্ট পেইজ থাকবে যেখানে একটি পোলের রেজাল্ট প্রদর্শন করা হবে
- আরেকটি পেইজ ব্যবহারকারীর সাবমিট করা ভোটটি প্রসেস করবে। প্রসেসিং শেষে অন্য কোথাও রিডিরেক্ট করে দিবে।

আমাদের প্রথম ডিউ

তাহলে এবার আমাদের প্রথম ডিউ লিখে দেখা যাক বাস্তবে এটা কিভাবে কাজ করে। ঝটপট `polls\views.py` খুলে চটপট করে লিখে ফেলুন নিচের কোডটুকু -

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello, world. You're at the poll index.")
```

ব্যাস, আমাদের ডিউ তৈরি করা শেষ। এবার পালা এই ডিউটিকে একটি URL এর সাথে সংযুক্ত করার। এজন্য আমরা একটি `URLconf` তৈরি করবো। `URLconf` নামটি জটিল হলেও এর কাজ কিন্তু বেশ সহজ সরল। কোড দেখলেই বুঝতে পারবো। আমরা `urls.py` নামে একটি ফাইল তৈরি করবো `polls` ডিরেক্টরীর মধ্য এবং এই কোড যোগ করবো -

```
from django.conf.urls import patterns, url

from polls import views

urlpatterns = patterns('',
    url(r'^$', views.index, name='index')
)
```

আসলে `URLconf` হলো `URL configuration` যেখানে আমরা বলে দেই কোন URL এর জন্য কোন ডিউ ব্যবহার করতে হবে। ফিক্সড URL এর পরিবর্তে একটি বিশেষ প্যাটার্ন এর URL বোঝানোর জন্য রেগুলার এক্সপ্রেশন ব্যবহার করারও সুযোগ রয়েছে।

আমরা আমাদের নিজেদের এ্যাপ্লিকেশন এর জন্য `urls.py` ফাইলে `URLconf` যোগ করলাম। এবার পালা এটাকে মূল `URLconf` এর সাথে যোগ করার। আমরা দেখেছিলাম `mysite/urls.py` ফাইলে প্রজেক্ট এর সব `URLconf` ছিলো। সেখানে আমরা এডমিন এ্যাপ্লিকেশন এর জন্য `URLconf` ইম্পোর্ট করেছিলাম। এবার আমরা আমাদের এ্যাপ্লিকেশনের জন্যও `URLconf` ইম্পোর্ট করবো যাতে জ্যাকসো রান করার সময় আমাদের `URLconf` লোড করে নেয়। আর সেটি করলে প্রকারান্তরে জ্যাকসো জানবে ঠিক কোন URL এর জন্য আমাদের এ্যাপ্লিকেশন এর কোন ডিউটি রান করতে হবে।

এবার তাহলে `mysite/urls.py` ফাইলটি খুলে নিচের কোড যোগ করে দেই -

```
from django.conf.urls import patterns, include, url

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    url(r'^polls/', include('polls.urls')),
    url(r'^admin/', include(admin.site.urls)),
)
```

এখানে আমরা জ্যাকসোকে জানিয়ে দিচ্ছি URL এর শুরুতে `polls` থাকলে `polls.urls` মডিউল থেকে `URLconf` লোড করে নিতে।

যোগ করি আরো কিছু ডিউ

এতক্ষণে আমরা কিভাবে ডিউ যোগ করতে হয় তা দেখলাম। এবার আমাদের এ্যাপ্লিকেশন এর জন্য প্রয়োজনীয় আরো কয়েকটি ডিউ যোগ করে নেই। বরাবরের মত `polls/views.py` ফাইলটি খুলে নিচের লাইনগুলো যোগ করে দেই -

```
def detail(request, poll_id):
    return HttpResponse("You're looking at poll %s." % poll_id)

def results(request, poll_id):
    return HttpResponse("You're looking at the results of poll %s." % poll_id)

def vote(request, poll_id):
    return HttpResponse("You're voting on poll %s." % poll_id)
```

এখানে আমরা ৩টি নতুন ডিউ যোগ করলাম - `detail`, `results`, `vote` - এবার এগুলোকে আমরা `URLconf` এ যোগ করবো। `polls/urls.py` ফাইলে প্রয়োজনীয় পরিবর্তন করে নেই যেন ফাইলটির কনটেন্ট নিম্নরূপ হয় -

```

from django.conf.urls import patterns, url

from polls import views

urlpatterns = patterns('',
    # ex: /polls/
    url(r'^$', views.index, name='index'),
    # ex: /polls/5/
    url(r'^(?P<poll_id>\d+)/$', views.detail, name='detail'),
    # ex: /polls/5/results/
    url(r'^(?P<poll_id>\d+)/results/$', views.results, name='results'),
    # ex: /polls/5/vote/
    url(r'^(?P<poll_id>\d+)/vote/$', views.vote, name='vote'),
)

```

এবার আপনার ব্রাউজারে ভিজিট করুন - `http://localhost:8000/polls/34` - দেখবেন `detail()` ভিউ রান করেছে। এমনি করে `http://localhost:8000/polls/34/results/` কিংবা `http://localhost:8000/polls/34/vote/` এ্যাড্রেস গুলো ভিজিট করে দেখুন। যথাক্রমে `results()` এবং `vote()` ভিউ দুটি থেকে আউটপুট পাবো আমরা।

এখানে উল্লেখ্য আমরা URL প্যাটার্ন নির্দেশ করার জন্য রেগুলার এক্সপ্রেশন ব্যবহার করেছি। রেগুলার এক্সপ্রেশন সম্পর্কে আরো বিস্তারিত জানতে চাইলে ঘুরে আসতে পারেন উইকিপিডিয়া থেকে -

http://en.wikipedia.org/wiki/Regular_expression। সহজ কথায় ব্যখ্যা করতে গেলে `^(?P<poll_id>\d+)/$` এই অংশটুকু নির্দেশ করে `/polls/` এর পর যদি কোন সংখ্যা থাকে URL এ তবে যেন

সেটির জন্য `detail()` ভিউটি ব্যবহার করা হয়। হয়তো ভাবছেন, রেগুলার এক্সপ্রেশনে তো `polls` এর কোন নাম বা চিহ্নও নাই তবে `/polls/` কোথা হতে আসলো? এটা আসলে এসেছে আমাদের মূল `URLconf` (যেটি কিনা `mysite/urls.py` ফাইলে আছে) হতে। ওখানে আমরা URL এর শুরুতে `/polls/` ম্যাচ করেছি। বাকিটা ম্যাচ করা হয়েছে এখানে।

কার্যকর ভিউ তৈরি করা

ভিউ নিয়ে অনেক কিছুই দেখলাম এতক্ষণ। এবার আসুন ভিউগুলোকে কাজের উপযোগী করে নেই। আগের ভিউগুলো আসলে কিছু করছিলো না। শুধু এক লাইন টেক্সট আউটপুট করছিলো। এবার আমরা এগুলোকে কাজের কাজী বানাবো। শুরু করি `index()` ভিউ দিয়ে। নিচের মত করে কোড লিখে ফেলি -

```

from django.http import HttpResponse

from polls.models import Poll

def index(request):
    latest_poll_list = Poll.objects.all().order_by('-pub_date')[:5]
    output = ', '.join([p.question for p in latest_poll_list])
    return HttpResponse(output)

```

ভিউগুলো আসলে একেকটি ফাংশন যা হয় `HttpResponse` অথবা কোনো এক্সেপশন (যেমন: `Http404`) রিটার্ন করবে। ভিউ থেকে রিটার্ন করা রেসপন্স জ্যাঙ্গো সরাসরি ব্রাউজারে আউটপুট করে।

এখন সমস্যা হলো আমাদের আউটপুট হার্ডকোড করা। অর্থাৎ আউটপুট পরিবর্তন করতে হলে আমাদেরকে ভিউ এর পাইথন কোড পরিবর্তন করার প্রয়োজন পড়বে। এই সমস্যার সমাধান করতে পারে টেম্প্লেটস!

টেম্প্লেটস - বেসিক

আসলে টেম্প্লেটস কি জিনিস? মা খালারা পিঠা বানানোর সময় এক ধরণের ছাচ ব্যবহার করেন। এর ভিতরে নকশা করাই থাকে। এই ছাচে পিঠা তৈরির উপকরণ বসিয়ে দিয়ে একটু কসরত করলেই পিঠার গায়ে তৈরি হয় নানা প্রকারের নকশা। টেম্প্লেট ও অনেকটা সেরকম। টেম্প্লেট এর কন্টেন্টগুলো বেশীরভাগই হয় স্ট্যাটিক যা পরিবর্তন হয় না। আর কিছু ডাইনামিক পার্ট থাকে যেগুলো প্রোগ্রামাবল। যখন টেম্প্লেট লোড করা হয় তখন টেম্প্লেট ইন্জিনকে এই ডাইনামিক পার্টগুলোর জন্য তথ্য সরবরাহ করা হয়। টেম্প্লেট ইন্জিন সেই তথ্যের উপর ভিত্তি করে পুরো কন্টেন্ট রেন্ডার করে।

যেমন ধরুন, একটা পেইজে আমাদের ব্যবহারকারীর নাম সহ একটা মেসেজ দেখানো প্রয়োজন। সেক্ষেত্রে নামটাই বারবার পরিবর্তন হবে, বাকি মেসেজটা একই থাকছে। আমরা কাজটি খুব সহজেই টেম্প্লেট দিয়ে করতে পারি। টেম্প্লেটটি হতে পারে এমন -

```
Hello {{ username }}, thank you for visiting!
```

এখানে `{{ username }}` এই অংশটি ডাইনামিক। এবার ধরুন একজন ব্যবহারকারীর নাম 'masnun', আমরা টেম্প্লেটকে বলে দিবে `username` এর ভ্যালু হবে 'masnun', তখন আমরা আউটপুট পাবো এরকম -

```
Hello masnun, thank you for visiting!
```

টেম্প্লেট ব্যবহার করলে আমাদেরকে পাইথন কোড নিয়ে ঘাটতে হচ্ছে না। ব্যাকেন্ড এর কোড আর ফ্রন্টএন্ড এর ডিজাইন খুব সহজেই পৃথক রাখতে পারছি। সেই সাথে টেম্প্লেট ইনহেরিট্যান্স, ব্লক প্রভৃতির সুবিধা নিয়ে আমরা খুব সহজে কোড মেইনটেইন করতে পারি।

জ্যাঙ্গোয় টেম্প্লেট

জ্যাঙ্গোতে টেম্প্লেটস থাকে এ্যাপ্লিকেশনের `templates` ডিরেক্টরীতে। আমরা `polls` ডিরেক্টরীতে `templates` নামে একটি ফোল্ডার তৈরি করি। এই ডিরেক্টরীতে আবার `polls` নামে আরেকটি ডিরেক্টরী তৈরি করি আমাদের টেম্প্লেটগুলোর জন্য। এখানে আমরা `index.html` নামে একটি ফাইল তৈরি করি। ফুল পথ হবে -

```
polls/templates/polls/index.html
```

। কন্টেন্ট হবে এরকম -

```
{% if latest_poll_list %}
    <ul>
    {% for poll in latest_poll_list %}
        <li><a href="/polls/{{ poll.id }}">{{ poll.question }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

এখানে `{{ poll.id }}` এবং `{{ poll.question }}` হলো ডাইনামিক কন্টেন্ট। এই টেম্পেটটির জন্য ভিউ টি মডিফাই করে নেই -

```
from django.http import HttpResponse
from django.template import Context, loader

from polls.models import Poll

def index(request):
    latest_poll_list = Poll.objects.all().order_by('-pub_date')[:5]
    template = loader.get_template('polls/index.html')
    context = Context({
        'latest_poll_list': latest_poll_list,
    })
    return HttpResponse(template.render(context))
```

এবার আমরা ব্রাউজারে আউটপুট দেখে নেই।

এই উদাহরণে টেম্পেট রেন্ডার করার জন্য বেশ খানিকটা কোড লিখতে হয়েছে। এই কাজটা আমরা সহজে করতে পারি জ্যাঙ্গোর একটা শর্টকাট ব্যবহার করে।

```
from django.shortcuts import render

from polls.models import Poll

def index(request):
    latest_poll_list = Poll.objects.all().order_by('-pub_date')[:5]
    context = {'latest_poll_list': latest_poll_list}
    return render(request, 'polls/index.html', context)
```

`render()` হচ্ছে আলোচ্য শর্টকাট। এরকম অনেক শর্টকাট আছে যেগুলো আমাদের নিত্য নৈমিত্তিক কাজগুলোকে সহজ করে দেয়।

404 ইরর পেইজ

ওয়েবের খুব কমন একটা http error হলো 404 । এটির মানে কন্টেন্ট এ্যাভেইলেবল না । এই ইরর থ্রো করার জন্য আমাদেরকে ভিউ থেকে আমাদের একটি `Http404` এক্সেপশন রেইজ করলেই চলবে । যেমন `detail()` ভিউটি হতে পারে এমন:

```
from django.http import Http404

def detail(request, poll_id):
    try:
        poll = Poll.objects.get(pk=poll_id)
    except Poll.DoesNotExist:
        raise Http404
    return render(request, 'polls/detail.html', {'poll': poll})
```

এই ভিউটির জন্য টেম্পেট আমরা পরে দেখবো তবে টেস্ট করার জন্য `polls/details.html` ফাইলে নিম্নোক্ত কন্টেন্ট যোগ করা যায়:

এবার ব্রাউজারে `http://localhost:8000/polls/34` ভিজিট করলে 404 ইরর মেসেজ পাওয়া যাবে ।

টিউটোরিয়াল - চতুর্থ অধ্যায়

তৃতীয় অধ্যায়ে আমরা যেখানে শেষ করেছি সেখান থেকেই আমরা চতুর্থ অধ্যায় শুরু করবো। আগের অধ্যায়গুলোর বিষয়বস্তু চোখ বুলিয়ে নিলে এই অধ্যায়টি আশ্বস্ত করতে সুবিধা হবে।

জ্যাক্সোয় ফর্ম তৈরি করা

ওয়েব এ্যাপ্লিকেশন ডেভেলপ করতে গেলে আমাদেরকে কম বেশি ফর্ম নিয়ে কাজ করতে হয়। ফর্মের মাধ্যমে ব্যবহারকারী ইনপুট প্রদান করতে পারেন। যাদের HTML এর জ্ঞান আছে তারা ইতোমধ্যে `<form>` এর সাথে পরিচিত। জ্যাক্সোয় আমরা কিভাবে ফর্ম ডাটা হ্যান্ডল করতে পারি সেটা দেখবো। আসুন `poll/detail.html` টেমপ্লেটটিতে নিচের মত করে কোড যোগ করে নেই -

```
<h1>{{ poll.question }}</h1>

{ % if error_message %}<p><strong>{{ error_message }}</strong></p>{ % endif %}

<form action="{ % url 'polls:vote' poll.id %}" method="post">
{ % csrf_token %}
{ % for choice in poll.choice_set.all %}
    <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}" />
    <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br />
{ % endfor %}
<input type="submit" value="Vote" />
</form>
```

এখানে আসলে কি করলাম আমরা?

- প্রত্যেকটা `choice` এর জন্য আমরা একটি করে রেডিও বাটন বসালাম। যখন ব্যবহারকারী যে কোন অপশন নির্বাচন করে ফর্মটি সাবমিট করবে তখন আমরা `POST` ডাটা হিসেবে পাবো এমন কিছু একটা - `choice=3`। (বিষয়টি আরো বিশদভাবে বোঝার জন্য HTML এবং `<form>` সম্পর্কে ভালো করে জানা প্রয়োজন)
- ফর্ম এর `action` হিসেবে আমরা ব্যবহার করেছি `{ % url 'polls:vote' poll.id %}` এবং `method` হিসেবে `post`। `url` একটি টেমপ্লেট ট্যাগ যেটি `polls:vote` এবং `poll.id` এর মান ব্যবহার করে সঠিক `url` টি আউটপুট দিবে।
- `forloop.counter` দিয়ে আমরা বুঝতে পারি `for` লুপ এ ঠিক কতবার লুপ ঘটেছে
- জ্যাক্সো স্বাভাবিকভাবে Cross Site Request Forgery ঠেকানোর চেষ্টা করে। এজন্য প্রতিটি ফর্মে `{ % csrf_token %}` থাকা অত্যাবশ্যক। এটি না থাকলে জ্যাক্সো ধরে নেয় রিকুয়েস্টটিতে কোন ঘাপলা আছে এবং সে আর এটা প্রসেস করে না। যদিও এটা ডিজএ্যাবল করা যায় কিন্তু এটা ব্যবহার করা উচিত।

আমরা আগের অধ্যায়ে নিচের `URLconf` টি তৈরি করেছিলাম -


```
url(r'^(?P<poll_id>\d+)/vote/$', views.vote, name='vote'),
```

সাথে সাথে আমরা একটি ডামি ভিউও তৈরি করেছিলাম যেটা আসলে তেমন কোন কাজই করতো না । এবার আমরা এই ভিউটিকে পরিবর্তন করে নিবো নিচের মত করে -

```
from django.shortcuts import get_object_or_404, render
from django.http import HttpResponseRedirect, HttpResponse
from django.core.urlresolvers import reverse
from polls.models import Choice, Poll
# ...
def vote(request, poll_id):
    p = get_object_or_404(Poll, pk=poll_id)
    try:
        selected_choice = p.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):
        # Redisplay the poll voting form.
        return render(request, 'polls/detail.html', {
            'poll': p,
            'error_message': "You didn't select a choice.",
        })
    else:
        selected_choice.votes += 1
        selected_choice.save()
        # Always return an HttpResponseRedirect after successfully dealing
        # with POST data. This prevents data from being posted twice if a
        # user hits the Back button.
        return HttpResponseRedirect(reverse('polls:results', args=(p.id,)))
```

এখানে কি করছি আমরা?

- `request.POST` হচ্ছে পাইথন ডিকশনারী । এখানে সব ডাটা কি-ভ্যালু জোড় হিসেবে থাকে । যেমন:
`request.POST['choice']` থেকে আমরা `choice` এর মান পেতে পারি । যদি POST ডাটায় `choice` না থাকে তবে জ্যাকসো `KeyError` রেইজ করবে । একারণেই আমরা `try...except` ব্লক ব্যবহার করেছি ।
- এখানে আমরা `HttpResponse` না পাঠিয়ে `HttpResponseRedirect` রিটার্ন করছি । এটার মাধ্যমে কোন আউটপুট না দেখিয়ে আমরা ব্যবহারকারীকে অন্য ঠিকানায় পাঠিয়ে দিতে পারি ।
- `reverse()` ফাংশনটি `{ % url %}` টেমপ্লেট ট্যাগের মত করেই ভিউ এর নাম থেকে পূর্ণ URL রিটার্ন করে ।

এখন এ্যাপ্লিকেশনে কেউ ভোট দিলে আমরা `results()` ভিউতে রিডিরেক্ট করে দিবে । আসুন সেই ভিউটি তৈরি করে নেই এবার -

```
def results(request, poll_id):
    poll = get_object_or_404(Poll, pk=poll_id)
    return render(request, 'polls/results.html', {'poll': poll})
```

এবার ডিউটির জন্য টেম্পেট তৈরি করে নেই -

```
<h1>{{ poll.question }}</h1>

<ul>
{ % for choice in poll.choice_set.all %}
  <li>{{ choice.choice_text }} -- {{ choice.votes }} vote{{ choice.votes|pluralize }}
</li>
{ % endfor %}
</ul>

<a href="{ % url 'polls:detail' poll.id %}">Vote again?</a>
```

এবার এ্যাপ্লিকেশনটি রান করুন এবং ভোট দিয়ে দেখুন কেমন কাজ করে ।

শেষ কথা

এই ৪টি অধ্যায় ভালো করে রপ্ত করতে পারলে জ্যাক্সো ব্যবহার করে সাধারণ ওয়েবসাইট খুব সহজেই তৈরি করতে পারবেন । তবে এখানেই থেমে থাকলে চলবে না । এই টিউটোরিয়ালটি নবীনদের কথা চিন্তা করে অফিশিয়াল টিউটোরিয়াল থেকে বেশ পরিবর্তন পরিবর্ধন করে লেখা হয়েছে । তাই অধিকাংশ সময়েই বেস্ট প্র্যাক্টিসের চাইতে সহজবোধ্যতার উপরে জোর দেওয়া হয়েছে । জ্যাক্সোর এ্যাডভান্সড কনসেপ্টগুলো আয়ত্ত্ব করতে পারলে এই কাজগুলো করা যায় আরো সহজে, আরো সুন্দর করে ।

জ্যাক্সোর নানা ফিচার, বেস্ট প্র্যাকটিস কিংবা এ্যাডভান্সড অনেক কনসেপ্ট নিয়ে আলাদা আলাদা ভাবে আলোচনা করা হবে এই বইয়ের অন্য অংশে ।

এই টিউটোরিয়াল ফলো করে তৈরি করা একটি স্যাম্পল এ্যাপ্লিকেশন পাওয়া যাবে শিবলি ভাইয়ের গিট রিপোজে - <https://github.com/Shibly/Poll> :)

হ্যালো ওয়ার্ল্ড

অনিরুদ্ধ অধিকারী

প্রথম অ্যাপ্লিকেশন

এবার আমরা প্রথমবারের মত জ্যাক্সো ব্যবহার করে "কিছু একটা" তৈরি করবো। (আবারও) প্রথমে কিছু জ্ঞানগর্ভ বাণী দেওয়া হবে (@%\$&*) এবং তার পরেই আমরা ঝাপিয়ে পড়বো কাজে!

প্রজেক্ট আর অ্যাপ

জ্যাক্সো এমনভাবে তৈরি হয়েছে যেন তা দ্বারা খুব সহজ থেকে শুরু করে খুবই জটিল ধরনের ওয়েবসাইট তৈরি করা যায়। সাজানো-গোছানো ভাবটা বজার রাখার স্বার্থে জ্যাক্সোতে প্রজেক্ট ও অ্যাপ নামে দু'টি ধারণা প্রবর্তন করা হয়েছে। প্রতিটি প্রজেক্টের মধ্যে এক বা একাধিক অ্যাপ থাকতে পারে।

জ্যাক্সো নিয়ে কাজ করা শুরু করলে ক্রমে এই প্রজেক্ট-অ্যাপ ব্যাপারটা আয়ত্ত্ব করে ফেলতে পারবেন। আপাতত একটা উদাহরণ দিই। ধারণ আপনাকে একটি পত্রিকার জন্য সাইট ডিজাইন করতে হবে। সেখানে নিয়মিত খবর প্রকাশিত হবার পাশাপাশি ব্লগ থাকবে, ম্যাক্সো পিপল সেই ব্লগে লেখালেখি করতে পারে। সেক্ষেত্রে একটি প্রজেক্ট হতে পারে omuk-potrika , সেই প্রজেক্টের মধ্যে থাকতে পারে কয়েকটি অ্যাপ news , blog ... ইত্যাদি।

```
omuk-potrika
-- news
-- blog
...
```

জ্যাক্সোর প্রজেক্টকে আপনি একটি কম্পিউটারের কেসিংয়ের সঙ্গে তুলনা করতে পারেন। কেসিংটি নিজে খুব একটা কাজ না করলেও, কেসিংয়ের মধ্যে যন্ত্রপাতিগুলোই "কাজ" করে। "কাজ" করা যন্ত্রপাতিগুলোকে আমরা অ্যাপের সঙ্গে তুলনা করতে পারি। অর্থাৎ, অ্যাপগুলো নিয়েই আমরা প্রকৃতপক্ষে কাজ করবো আর প্রজেক্টটি হল এর ধারক/বাহক বা কন্ট্রোলার মত কিছু একটা! তাহলে ধরে নিলাম, প্রজেক্ট আর অ্যাপের ধারণা কিছুটা ঝাপসা হয়ে হলেও আয়ত্ত্ব হয়েছে - এবার কাজ শুরু হাতে-কীবোর্ডে!

আসলে কি করবো?

আমরা প্রথমে এমন একটি ওয়েবসাইট তৈরি করবো, যা "Hello World!" লেখাটি ব্রাউজারে প্রদর্শন করবে। জ্যাক্সোর জগতের সঙ্গে পরিচিত হবার জন্য এই অতিরিক্ত রকমের সিম্পল অ্যাপটি আমাদের অনেক সাহায্য করবে। এইবার নিশ্চিত হয়ে নিই বিদ্যুৎ আছে? কম্পিউটার কাজ করছে? নিশ্চাপ অনুভব করলে ছোটকাজ সেরে আসুন। সব ঠিকঠাক? জোশ! ৩-২-১-গো!

নতুন প্রজেক্ট তৈরি

প্রথমে আপনার টার্মিনাল চালু করুন। এবার প্রজেক্টের জন্য মনের পছন্দমত একটি নাম বেছে নিই (নামে কোন স্পেস থাকতে পারবে না)। আমি আমার প্রজেক্টের নাম ঠিক করলাম `helloworldproject`। এই নামের একটি প্রজেক্ট তৈরি করার জন্য টার্মিনালে কমান্ড দিই,

```
django-admin.py startproject helloworldproject
```

এবার দেখবো `helloworldproject` নামের একটি সুন্দর ফোল্ডার তৈরি হয়ে গেছে এবং তার ভেতর কতগুলো ফাইল ও ফোল্ডার রয়েছে। ফোল্ডারটির ভেতরে থাকা জিনিসপত্রের লিস্ট খানিকটা এরকম:

```
helloworldproject/
  manage.py
  helloworldproject/
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

হুম, তাহলে ওই কমান্ডটি দেওয়ার পর `helloworldproject` নামের একটি ফোল্ডার তৈরি হল, তার মধ্যে `manage.py` নামের একটি ফাইল আর আবার আরেকটি `helloworldproject` ফোল্ডার রয়েছে। ওই ফোল্ডারের মধ্যে কতকগুলো অল্প নামের ফাইল রয়েছে। ফাইলগুলোর কাজ আমরা ধীরে ধীরে জানবো। যাই হোক, আমরা তৈরি করে ফেললাম আমাদের প্রথম জ্যাকো প্রজেক্ট! কিন্তু, এতো কেবল স্কেলেটন মানে আমাদের কাঙ্ক্ষিত জিনিসের কঙ্কাল। এবার রক্তমাংস বসানোর পালা!

প্রজেক্টের মধ্যে অ্যাপ তৈরি

`cd helloworldproject` কমান্ডটির মাধ্যমে উপরের কমান্ড দেওয়ার পর তৈরি নতুন ফোল্ডারটির মধ্যে প্রবেশ করি। এখন আমরা আমাদের প্রজেক্টের ফোল্ডারের মধ্যে অবস্থান করছি। এইবার রক্তমাংস (অ্যাপ) তৈরির পালা। ধরি আমরা `helloworldapp` নামের একটি অ্যাপ তৈরি করবো। এজন্য কমান্ড দিই,

```
django-admin.py startapp helloworldapp
```

ইউরেকা! আমরা আমাদের প্রথম অ্যাপ তৈরি করে ফেললাম! এই কমান্ডটি দেয়ার ফলে `helloworldapp` নামে একটি কিউট ফোল্ডার তৈরি হবে। এই ফোল্ডারের মধ্যে যা আছে...

```
helloworldapp
  __init__.py
  models.py
  tests.py
  views.py
```

মোটমোট চারটি ফাইল! অর্থাৎ আমরা শুরুতে যে helloworldproject ফোল্ডারে কাজ শুরু করেছিলাম, তার চেহারা মোটামুটি এরকম:

```
helloworldproject/
  manage.py
  helloworldapp
    __init__.py
    models.py
    tests.py
    views.py
  helloworldproject/
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

ডেভেলপমেন্ট সার্ভার চালিয়ে দেখা

এবার একটু আমাদের ডেভেলপমেন্ট সার্ভারের সঙ্গে পরিচিত হই। আপাতত ডেভেলপমেন্ট চলাকালীন আমাদের আলাদা কোন সার্ভার ইন্সটল করার দরকার নেই, জ্যঙ্গোর নিজস্ব সার্ভারই যথেষ্ট হবে। চোখ জুড়ানোর জন্য আসুন একবার সার্ভারটি চালিয়ে দেখি। এজন্য আমরা শুরুতে যে helloworldproject ফোল্ডারের মধ্যে কাজ শুরু করেছিলাম, সেখানে ফিরে যাই। কমান্ড দেই,

```
python manage.py runserver
```

এই কমান্ডটি দেওয়ার পর টার্মিনালে কিছু আউটপুট আসবে।

```
Validating models...

0 errors found
May 27, 2013 - 00:29:33
Django version 1.5.1, using settings 'helloworldproject.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

এর থেকে বোঝা গেল আমাদের সিস্টেমের 8000 নম্বর পোর্টে জ্যঙ্গোর সার্ভার চলছে। আমাদের ব্রাউজারের সাহায্যে <http://localhost:8000> ঠিকানায় ব্রাউজ করলে আমরা জ্যঙ্গোর ডিফল্ট হোমপেজটি দেখতে পাবো।

It worked!

Congratulations on your first Django-powered page.

Of course, you haven't actually done any work yet. Here's what to do next:

- If you plan to use a database, edit the DATABASES setting in a/settings.py.
- Start your first app by running `python manage.py startapp [appname]`.

You're seeing this message because you have `DEBUG = True` in your Django settings file and you haven't configured any URLs. Get to work!

আমরা যেহেতু এখনো কোড কোডই লিখিনি, তাই এই অবস্থা আরকি! চোখ জুড়ালো? আমরা প্রতিবার আমাদের অ্যাপ্লিকেশন ডেভেলপমেন্ট চলাকালীন টেস্ট করার জন্য উপরের কমান্ডটি দিয়ে ডেভেলপমেন্ট সার্ভার চালিয়ে দেখব। আপাতত Ctrl+C চেপে ডেভেলপমেন্ট সার্ভার বন্ধ করে দিই।

স্বস্তি প্রথম কোড

ব্রাউজারে তো অনেক সুন্দর একটা Welcome মেসেজ দেখলাম, কিন্তু নিজেরা তো কিছুই কোড করিনি! এইবার নিজের হাতে কোড করে দেখবো। বাবুই পাখি তো বলেই গিয়েছে, "নিজ হাতে গড়া মোর কাঁচা ঘর খাসা!" আসুন নেমে পড়ি।

প্রথম ভিউ

প্রথমবারের জন্য আমরা কোন ধরনের ঝামেলায় না গিয়ে শুধুমাত্র একটি ভিউ লিখবো। টেমপ্লেট ও মডেল সম্পর্কে ক্রমে ক্রমে পরে জানবো। আচ্ছা এইবার ছোট্ট একটা কুইজ, বলুন তো, আপনি কি কোথাও `views` বা এর কাছাকাছি নামের কোন ফাইল দেখেছেন? অবশ্যই দেখেছেন! `helloworldproject > helloworldapp` ফোল্ডারের মধ্যেই রয়েছে `views.py` !

এইবার আপনার পছন্দের টেক্সট এডিটরে ফাইলটি চটপট খুলে ফেলুন। ফাইলে যা লেখা আছে মুছে ফেলুন। প্রথমে আমাদের কিছু জিনিসপত্র জরুরী ভিত্তিতে আমদানী করতে হবে। `django.http` থেকে `HttpResponse` কে আমদানি (`import`) করে ফেলুন।

```
from django.http import HttpResponse
```

আমদানীর কাজ শেষ। এইবার আমরা একটি ছোট্ট ফাংশন লিখবো। ফাংশনটির নাম হবে `index` বা আপনার পছন্দমত যেকোন নাম (`gittu` , `lutuputu` , `bashay_jane` ...)। ফাংশনটি একটি আর্গুমেন্ট বা প্যারামিটার গ্রহণ করবে। জ্যাকসো ডেভেলপারদের মধ্যে এক ধরনের অলিখিত নিয়ম হল এই আর্গুমেন্ট ড্যারিয়েবলটির নাম `request` রাখা। আপনি চাইলে অন্য নামও রাখতে পারেন কিন্তু `request` রাখাটাই যুক্তিসঙ্গত কেননা... থাক পরে বুঝিয়ে বলবো।

যাক, আর কনফিউজ না হয়ে এইবার একটি `HttpResponse` অবজেক্ট রিটার্ন করবো আমাদের ফাংশন থেকে। `HttpResponse` এর প্রথম আর্গুমেন্টটি হবে যা আপনি ডিজিটরের ব্রাউজারে দেখাতে চান, ঠিক তাই!

```
from django.http import HttpResponse

def index(request):
    return HttpResponse('Hello World!')
```

এইবার ডেভেলপমেন্ট সার্ভার চালিয়ে দেখুন তো। ইউরেকা! কিছুই আসেনি! (আগের মত জ্যাক্সের ডিফল্ট ওয়েলকাম মেসেজ ছাড়া)। তাহলে আমরা কি ভুল করলাম? হুম, ছোট্ট একটা জিনিস মিসিং রয়ে গেছে। আমরা জ্যাক্সকে কিন্তু বলিনি কোন URL প্যাটার্ন বা URL এর জন্য আমাদের `index` নামের ভিউটি কার্যকর হবে। (হ্যাঁ, ঠিক ধরেছেন। `views.py` ফাইলের মধ্যে প্রতিটি ফাংশনই "সাধারণত" এক একটি ভিউ।) এইবার URL নিয়ে কাজকারবার।

প্রথম urlpattern

আপনি মনে করার চেষ্টা করুন তো, কোথাও কি `url` বা এই ধরনের নামের কোন ফাইল দেখেছেন? দেখেছেন বলে আমার বিশ্বাস, `helloworldproject > helloworldproject` ফোল্ডারের মধ্যেই কিন্তু `urls.py` নামের একটি ফাইল রয়েছে! এই ফাইলের মধ্যেই আমরা URL ম্যাচিংয়ের কাজ করবো। আসুন, আমাদের পছন্দের টেক্সট এডিটরে ফাইলটি খুলে ফেলি। সবার আগে আবারও কিছু আমদানীর কাজ করতে হবে। চটপট `helloworldapp.views` আমদানী(`import`) করে ফেলুন। কি ইমপোর্ট করলেন বুঝতে পেরেছেন তো? না বুঝলে কিছুক্ষণ ভাবুন।

```
import helloworldapp.views
```

এইবার খেয়াল করে দেখুন `urlpatterns = patterns('',` একটি লাইন আছে, সেই লাইনটি পর আমাদের `urlpatterns` লিখে ফেলবো। আমাদের লেখা প্যাটার্নটির উপর ভিত্তি করে জ্যাক্সো সিদ্ধান্ত নেবে কোন URL এর জন্য কোন ভিউ ব্যবহার করবে। নিচের লাইনটি আমাদের `urlpatterns`।

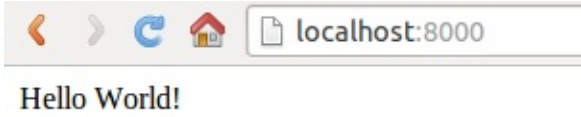
```
url(r'^$', helloworldapp.views.index)
```

তো আমাদের প্যাটার্ন হল মূলত একটি `url` অবজেক্ট। এর প্রথম আর্গুমেন্টটি হল একটি রেগুলার এক্সপ্রেসন। রেগুলার এক্সপ্রেসন সম্পর্কে জানার জন্য অনেক ব্লগ পোস্ট, বই, ভিডিও টিউটোরিয়াল আছে (আপনার প্রিয় সার্চ ইঞ্জিন ব্যবহার করুন), রেগুলার এক্সপ্রেসন শিখানো এই বইয়ের পরিধির বাইরে। যাই হোক, এরপরে যে ভিউটি কার্যকর করা হবে, সেই ভিউটিকে (ফাংশনটি) উল্লেখ করতে হবে। এইবার ইমপোর্টের কারণটি বুঝতে পেরেছেন আশা করি! আমরা যে অ্যাপ থেকে কোন ভিউ আনতে চাই, সেটির `views.py` কে আমরা ইমপোর্ট করে ওপরের মত ব্যবহার করবো। তাহলে, `urls.py` এর চেহারা দাঁড়ালো...

```
from django.conf.urls import patterns, include, url
import helloworldapp.views

urlpatterns = patterns('',
    url(r'^$', helloworldapp.views.index),
)
```

সাধারণত আপনার `urls.py` তে অনেকগুলো কমেণ্ট থাকবে। ওগুলো মোছার দরকার নেই। ওরা ওদের মত থাক, কেননা আমাদের টিউটোরিয়ালের পরবর্তী অংশে যখন আমরা স্বয়ংক্রিয় অ্যাডমিন ইন্টারফেস ব্যবহার করা শিখবো, তখন ওই কমেণ্টগুলোকে সাংঘাতিক রকমের প্রয়োজন হবে! যাই হোক, এইবার আবার আমাদের মূল প্রজেক্টের ফোল্ডারে ফিরে যাই এবং ডেভেলপমেন্ট সার্ভার চালু করি। চালু করে ব্রাউজারে <http://localhost:8000> তথা আমাদের ডেভেলপমেন্ট সার্ভারে প্রবেশ করি। অভিনন্দন! আপনি আপনার প্রথম জ্যাকো অ্যাপ তৈরি করে ফেলেছেন!



আনন্দে লাফ দেওয়া হয়ে গেছে? ব্যাস, হোমওয়ার্ক টাইম! এটি আমাদের প্রথম হোমওয়ার্ক। তাই, কোন রকমের ফাঁকি চলবে না!

হোমওয়ার্ক / বাড়ির কাজ

1. "Hello World!" লেখাটির বদলে নিজের ইচ্ছেমত কোন লেখা দেখান।
2. প্লেইন টেক্সটের বদলে HTML ব্যবহারের চেষ্টা করুন।
3. `index` ভিউটির নাম পরিবর্তন করে আপনার পছন্দমত নাম দিন, সেই অনুসারে `urls.py` পরিবর্তন করুন।
4. আমাদের বর্তমানে `urlpatterns` পরিবর্তন করার চেষ্টা করুন। (এই কাজে সফল হলে আপনাকে 4x অভিনন্দন)